COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Predictive model of stock market values using Deep Learning and Sentiment Analysis

Autor: Ignacio Sagardoy Valverde

Director: Marco Lippi

Madrid

Agosto de 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Predictive model of stock market values using Deep Learning and Sentiment Analysis

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el curso académico 2020/21 es de mi autoría, original e inédito y no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido tomada de otros documentos está debidamente referenciada.

[illegible]

Fdo.: Ignacio Sagardoy Valverde Fecha: 21/08/2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: [illegible] Fecha: 21/08/2021

AUTHORIZATION FOR DIGITALIZATION, STORAGE AND DISSEMINATION IN THE NETWORK OF THE END-OF-DEGREE PROJECTS, MASTER PROJECTS DISSERTATIONS OR BACHILLERATO REPORTS

1. Declaration of authorship and accreditation thereof.
The author Mr. /Ms. IGNACIO SAGARDOY VALVERDE

HEREBY DECLARES that he/she owns the intellectual property rights regarding the piece of work: PREDICTIVE MODEL OF STOCK MARKET VALUES USING DEEP LEARNING AND SENTIMENT ANALYSIS that this is an original piece of work, and that he/she holds the status of author, in the sense granted by the Intellectual Property Law.

2. Subject matter and purpose of this assignment.

With the aim of disseminating the aforementioned piece of work as widely as possible using the University's Institutional Repository the author hereby **GRANTS** Comillas Pontifical University, on a royalty-free and non-exclusive basis, for the maximum legal term and with universal scope, the digitization, archiving, reproduction, distribution and public communication rights, including the right to make it electronically available, as described in the Intellectual Property Law. Transformation rights are assigned solely for the purposes described in a) of the following section.

3. Transfer and access terms

Without prejudice to the ownership of the work, which remains with its author, the transfer of rights covered by this license enables:

a) Transform it in order to adapt it to any technology suitable for sharing it online, as well as including metadata to register the piece of work and include "watermarks" or any other security or protection system.

b) Reproduce it in any digital medium in order to be included on an electronic database, including the right to reproduce and store the work on servers for the purposes of guaranteeing its security, maintaining it and preserving its format.

c) Communicate it, by default, by means of an institutional open archive, which has open and cost-free online access.

d) Any other way of access (restricted, embargoed, closed) shall be explicitly requested and requires that good cause be demonstrated.

e) Assign these pieces of work a Creative Commons license by default.

f) Assign these pieces of work a HANDLE (*persistent* URL). by default.

4. Copyright.

The author, as the owner of a piece of work, has the right to:

a) Have his/her name clearly identified by the University as the author
b) Communicate and publish the work in the version assigned and in other subsequent versions using any medium.
c) Request that the work be withdrawn from the repository for just cause.
d) Receive reliable communication of any claims third parties may make in relation to the work and, in particular, any claims relating to its intellectual property rights.

5. Duties of the author.

The author agrees to:

a) Guarantee that the commitment undertaken by means of this official document does not infringe any third party rights, regardless of whether they relate to industrial or intellectual property or any other type.

b) Guarantee that the content of the work does not infringe any third party honor, privacy or image rights.
c) Take responsibility for all claims and liability, including compensation for any damages, which may be brought against the University by third parties who believe that their rights and interests have been infringed by the assignment.
d) Take responsibility in the event that the institutions are found guilty of a rights infringement regarding the work subject to assignment.

6. Institutional Repository purposes and functioning.

The work shall be made available to the users so that they may use it in a fair and respectful way with regards to the copyright, according to the allowances given in the relevant legislation, and for study or research purposes, or any other legal use. With this aim in mind, the University undertakes the following duties and reserves the following powers:

a) The University shall inform the archive users of the permitted uses; however, it shall not guarantee or take any responsibility for any other subsequent ways the work may be used by users, which are non-compliant with the legislation in force. Any subsequent use, beyond private copying, shall require the source to be cited and authorship to be recognized, as well as the guarantee not to use it to gain commercial profit or carry out any derivative works.
b) The University shall not review the content of the works, which shall at all times fall under the exclusive responsibility of the author and it shall not be obligated to take part in lawsuits on behalf of the author in the event of any infringement of intellectual property rights deriving from storing and archiving the works. The author hereby waives any claim against the University due to any way the users may use the works that is not in keeping with the legislation in force.
c) The University shall adopt the necessary measures to safeguard the work in the future.
d) The University reserves the right to withdraw the work, after notifying the author, in sufficiently justified cases, or in the event of third party claims.

Madrid, on 21 of August , 2021

Chapter 1. HEREBY ACCEPTS

Signed IGNACIO SAGARDOY [illegible]

Reasons for requesting the restricted, closed or embargoed access to the work in the Institution's Repository

COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

GRADO EN INGENIERÍA EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

Predictive model of stock market values using Deep Learning and Sentiment Analysis

Autor: Ignacio Sagardoy Valverde

Director: Marco Lippi

Madrid

Agosto de 2021

Acknowledgements

After the completion of this study, I would like to express my gratitude and appreciation to those whose support and help provided contributed to the culmination of my Undergraduate Thesis Project:

To my family for being constantly supportive during the long hours of hard work put to this study.

To my friends for being able to put up for my unavailability during several months, while being supportive to bring the best of me.

To the Lord Almighty for giving me the chance to revel in the culmination of this project.

PREDICTIVE MODEL OF STOCK MARKET VALUES USING DEEP LEARNING AND SENTIMENT ANALYSIS

Author: Sagardoy Valverde, Ignacio.
Supervisor: Lippi, Marco.

ABSTRACT

Making use of simulations in Python, this paper concludes that the model that most effectively forecasts the value of a stock is a Deep Neural Network that employs two layers and two hidden units in each layer, combining both technical data and the Twitter's publications that debated the situation of the stock analyzed. The simulation of the model was validated using actual stock records.

Keywords: Deep Neural Network, hyperparameter tuning, stock value, overfitting, test set, training set.

1. Introduction

For many years now, economists, mathematicians and stock market experts have devoted much time and resources to stock value prediction [1]. The key to success lies in understanding what are the most relevant elements that influence the stock price and how they affect it.

Traditionally, data analysis methods such as decision trees, logistic regression or S.V.M. have been used to understand the behavioural patterns of stock values; however, thanks to the digitalization of society, new tools have appeared that threaten to render these obsolete [2]–[4].

The aim of this project is to develop an algorithm that maximizes the use of new technologies and incorporates those non-quantitative features that influence the value of the share, but are not usually taken into account in technical analysis.

2. State of the Art

Throughout history, many models have been developed to advise buying or selling in the stock market, but two have stood out above the rest and are still being used today: fundamental analysis and technical analysis [5], [6]. These models analyze the value and evolution of the stock prices using very different approaches and therefore different tools.

Fundamental analysis is defined as a financial and accounting methodology that studies the profitability of an investment through financial statements [7]. The main objective of the analysis is to generate an accurate estimate of the value that the entity should have through a valuation of different parameters. Among many elements, some of the factors analyzed are company financial metrics (assets, liabilities, earnings, etc.), management effectiveness, and broader indicators such as GDP forecasts, interest rate evolution, debt spread and country political stability. The ultimate goal is to determine the fair value of the company and whether it is over or undervalued.

On the other hand, technical analysis is based on the review of historical data to predict future market behavior [2]. It is based on the notion that, in similar circumstances, history repeats itself and does so in recognizable patterns. This approach has been strongly reinforced in recent years with the advent of new technologies as they improved the ability to identify behavioral indicators and thus increase forecasting skills [8].

With the proliferation of new technologies, numerous models for asset prediction have been developed [9], [10]. These new models can be classified into three groups: linear models, non-linear models and hybrid models. All these different methods allow the computational skills of the models to be improved, yet the step change they brough was the possibility to enhance performance through the use of more complex input variables such as macroeconomic factors or other indicators related to the market or sector in which the company operates [11], [12].

In addition to the above, nowadays also the impact of social networks on stock performance is beginning to be taken into account. The reason for this is because it has been evidenced that events happening in these media platforms (influencers' publications, activity in chat forums, etc.) can trigger changes in the stock market prices [3], [13]–[15].

3. Methodology

The process followed for the execution of this project, had three main phases. The first stage consisted in a training process in which the student attended several courses to increase his knowledge in relation to the area of work. These courses were imparted by the universities of Michigan, Modena and Reggio Emilia and the company DeepLearning.ai and it took the student approximately 3 months to fulfill them all.

Once the courses had been completed, the first thing to do was to establish the requirements to be met by the databases (quantity, variety, typology, etc.). This is a key decision since, depending on the output desired and the available data, the code has to be programmed in one way or another. Having already established the criteria, the next step was to download the data using an opensource platform and begin the processes that regarded the cleansing of the information downloaded.

Finally, the last stage of this process consisted in the creation and validation of three different models: a Logistic Regression Model, a Monolayered Neural Network and a Deep Neural Network. These three models differed from each other on the complexity of the data computation. The reason for developing different algorithms was to validate which one best interpreted the data to generate the most accurate predictions. During the building of the models, it was necessary to tune the different settings (hyperparameter tuning) to ensure the best results from each one.

In order to facilitate the understanding of the methodology followed throughout the project, the below flowchart shows the key steps undergone. It will help to have a general overview of the course of the study.

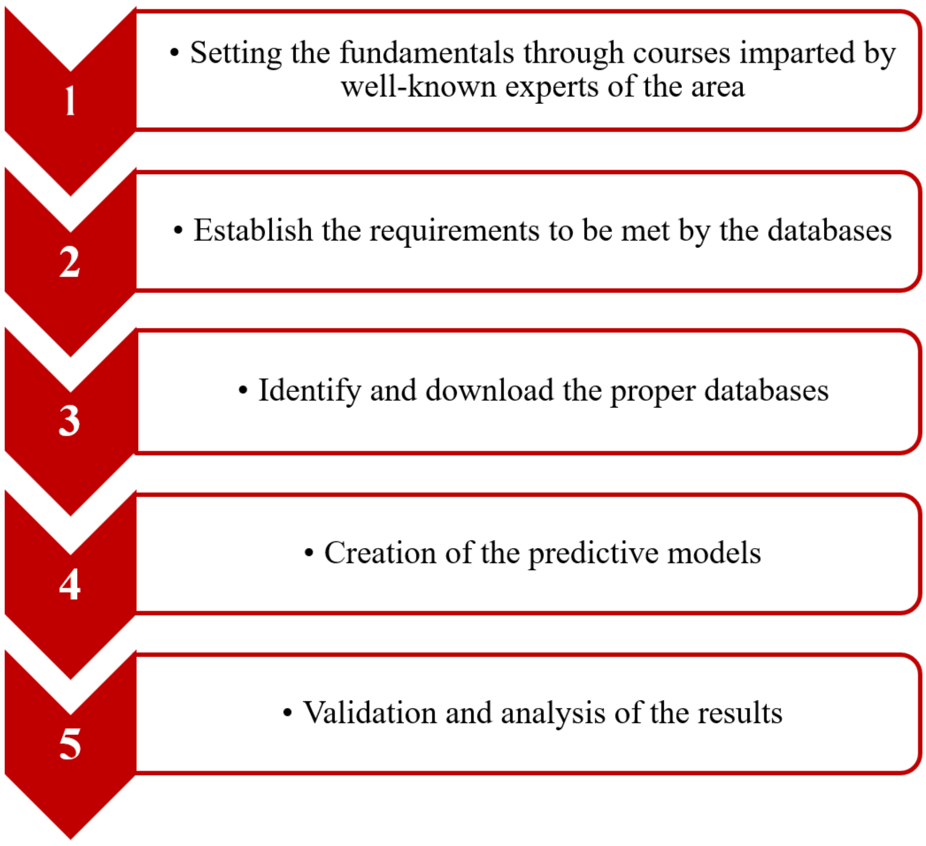

1
• Setting the fundamentals through courses imparted by well-known experts of the area
2
• Establish the requirements to be met by the databases
3
• Identify and download the proper databases
4
• Creation of the predictive models
5
• Validation and analysis of the results

Figure 1-1: summary of methodology

4. Results

The behavior of the models at different stages of the optimization process shows the following results:

		Logistic Regression	One Layered Neural Network	Deep Layered Neural Network
Stock values	Initial Computation	72.7/73	74/70.8	54.1/61
	Number of reference days	81.8/77.6	85.5/82.3	92.2/89.6
	Number of iterations	81.8/80.6	87/83.8	92.2/89.6
	Final Computation	81.8/80.6	90.8/88.3	92.2/89.6
Sentiment scores	Initial Computation	54.4/61	57/60	54/62
	Number of reference days	67.8/70.1	88/87	92.9/91
	Number of iterations	68/70.2	88/87	92.9/91
	Final Computation	69.1/68.4	88/87	92.9/91
Stock values Sentiment scores	Initial Computation	56.4/54	71/54	53.8/60.4
	Number of reference days	80.6/79.5	92/88	93.5/92
	Number of iterations	80.6/79.5	92/90	93.5/92
	Final Computation	80.6/79.5	92/90	93.5/92

Figure 1-2: table of results

As showed on the chart, for each stage two values are presented: the first one represents the percentage of efficiency the model achieves when tested with the data with which it has been trained (training set performance), while the second one represents the model percentage of efficiency when tested with new data that it has not seen before (test set performance). Through the combined analysis of both values, the efficiency of the different models can be determined.

In order to choose the best model, two fundamental criteria are used: on one hand, the aim is to achieve efficiencies close to 100%; on the other, it is also necessary that both values are very high. If the training set performance is 98% but the test set performance is 80%, this means that the model is memorizing the values with which it was trained but then it is useless with data that it has not seen before. This problem is known as overfitting and it must always be reduced in order to ensure the most efficient forecasting.

5. Conclusions

Taking into account all the results obtained, it can be concluded that the model that best predicts the future value of a stock is the Deep Neural Network with two layers and two hidden units on each layer, combining numerical and non-numerical data.

In addition to determining which model was the most recommendable, another purpose of the work was to determine the best data source to generate the best predictions for each system. From the results obtained, it can be concluded that the most elaborate data source (stock values together with sentiment ratings) delivers the best results. What is interesting though is that the second most useful data source was not the sentiment ratings but the stock values; the reason for this is that being simpler information, it was easier for the different models to understand the relationship between the input matrices and therefore predict future behaviors more accurately. These findings are very interesting because they reinforce the theoretical basis of the technical analysis, which suggest that stock markets move in recognizable patterns and therefore are not only driven by external actions such as social media postings [6].

Finally, the last relevant observation is the impact of the different hyperparameters and how optimizing them helps to improve the model considerably. After analyzing the effect of each of the hyperparameters separately, it is concluded that the element that most influences the performance of the different models is clearly the number of previous days to be taken into account. This makes sense, since in many studies regarding model performance through hyperparameter tuning it is stated that the amount of input data is one of the most important hyperparameters - if not the most important [16]–[18]. It is proven that the increase of the number pairs of data for each of the training examples promotes and facilitates pattern recognition.

6. References

[1] M. R. Vargas, C. E. M. Dos Anjos, G. L. G. Bichara, and A. G. Evsukoff, “Deep Leaming for Stock Market Prediction Using Technical Indicators and Financial News Articles,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2018-July, Oct. 2018, doi: 10.1109/IJCNN.2018.8489208.

[2] A. Lo and J. Hasanhodzic, “The evolution of technical analysis: financial prediction from Babylonian tablets to Bloomberg terminals,” 2010, Accessed: Aug. 17, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=aJv9DwAAQBAJ&oi=fnd&pg=PR7&dq=technical+analysis+munehisa+homma&ots=crkJzWRQwU&sig=GR_Z-fHu7JjQY5VwsK3ObH8oJ0c.

[3] W. Jiang, “Applications of deep learning in stock market prediction: Recent progress,” *Expert Syst. Appl.*, vol. 184, p. 115537, Dec. 2021, doi: 10.1016/J.ESWA.2021.115537.

[4] A. E. de Oliveira Carosia, G. P. Coelho, and A. E. A. da Silva, “Investment strategies applied to the Brazilian stock market: A methodology based on Sentiment Analysis with deep learning,” *Expert Syst. Appl.*, vol. 184, p. 115470, Dec. 2021, doi: 10.1016/J.ESWA.2021.115470.

[5] B. G. Inchausti, C. R. Maya, and M. A. Gisbert, “El papel del análisis fundamental en la investigación del mercado de capitales: Análisis crítico de su evolución,” *Rev. Esp. Financ. y Contab.*, vol. 31, no. 114, pp. 1111–1150, 2002, doi: 10.1080/02102412.2002.10779470.

[6] J. Chen, “Essentials of technical analysis for financial markets,” 2010, Accessed: Aug. 17, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=202I0ZWA6tIC&oi=fnd&pg=PR13&dq=technical+analysis+of+the+financial+markets&ots=kj5ZIs13C4&sig=x4plLnpCU4MvDr0cl5OCecYVuK8.

[7] B. Graham, D. Dodd, and S. Cottle, “Security analysis,” 1934, Accessed: Aug. 17, 2021. [Online]. Available: https://bridgecollege.net/download/75/accounting-and-finance/6485/00899.pdf.

[8] H. Liu, J. H.-E. S. with Applications, and undefined 2010, “Forecasting S&P-100 stock index volatility: The role of volatility asymmetry and distributional assumption in GARCH models,” *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417409010689.

[9] Y. Zhang, L. W.-E. systems with applications, and undefined 2009, “Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network,” *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800852X.

[10] S. Chaigusin, … C. C.-2008 I., and undefined 2008, “The use of neural networks in the prediction of the stock exchange of Thailand (SET) Index,” *ieeexplore.ieee.org*, pp. 670–673, 2008, doi: 10.1109/CIMCA.2008.83.

[11] D. P.-T. and E. D. of and undefined 2010, “Macroeconomic indicators and their impact on stock market performance in the short and long run: the case of the Baltic States,” *ceeol.com*, vol. 16, no. 2, pp. 291–304, 2010, doi: 10.3846/tede.2010.19.

[12] K. Hussainey and L. Khanh Ngoc, “The impact of macroeconomic indicators on Vietnamese stock prices,” *J. Risk Financ.*, vol. 10, no. 4, pp. 321–332, Aug. 2009, doi: 10.1108/15265940910980632/FULL/HTML.

[13] A. Anta, L. Chiroque, P. M.-… del lenguaje natural, and undefined 2013, “Sentiment analysis and topic detection of Spanish tweets: A comparative study of of NLP techniques,” *journal.sepln.org*, 2013, Accessed: Aug. 11, 2021. [Online]. Available: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/download/4658/2760.

[14] D. Valle-Cruz, V. Fernandez-Cortez, A. López-Chau, and R. Sandoval-Almazán, “Does Twitter Affect Stock Market Decisions? Financial Sentiment Analysis During Pandemics: A Comparative Study of the H1N1 and the COVID-19 Periods,” *Cogn. Comput. 2021*, vol. 1, pp. 1–16, Jan. 2021, doi: 10.1007/S12559-021-09819-8.

[15] Y. Xu and S. B. Cohen, “Stock Movement Prediction from Tweets and Historical Prices,” *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 1970–1979, 2018, doi: 10.18653/V1/P18-1183.

[16] “Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization | Coursera.” https://www.coursera.org/learn/deep-neural-network?specialization=deep-learning (accessed Aug. 14, 2021).

[17] M. Feurer, F. H.-A. machine learning, and undefined 2019, “Hyperparameter optimization,” *library.oapen.org*, Accessed: Aug. 19, 2021. [Online]. Available: https://library.oapen.org/bitstream/handle/20.500.12657/23012/1007149.pdf?sequence=1#page=15.

[18] M. Parsa, J. Mitchell, … C. S.-F. in, and undefined 2020, “Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design,” *frontiersin.org*, Accessed: Aug. 19, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00667/full?report=reader.

MODELO PREDICTIVO DEL VALOR DE ACCIONES EN BOLSA EMPLEANDO DEEP LEARNING Y SENTIMENT ANALYSIS

Autor: Sagardoy Valverde, Ignacio.
Supervisor: Lippi, Marco.

ABSTRACT

Haciendo uso de simulaciones en Python, este trabajo concluye que el modelo que pronostica con mayor eficacia el valor de una acción es una Red Neuronal Profunda que emplea dos capas y dos unidades ocultas en cada capa. Para esta predicción se combinan tanto datos técnicos como las publicaciones de Twitter que debaten la situación del activo analizado. La simulación del modelo se validó utilizando registros reales de acciones.

Palabras clave: Red Neuronal Profunda, optimización de hiperparámetros, valor de la acción, overfitting, test set, training set.

1. Introducción

Desde hace muchos años, economistas, matemáticos y expertos en bolsa han dedicado mucho tiempo y recursos a la predicción del valor de las acciones [1]. La clave del éxito reside en comprender cuáles son los elementos más relevantes que influyen en el precio de las acciones y cómo lo hacen.

Tradicionalmente, se han empleado métodos de análisis de datos como los árboles de decisión, la regresión logística o los S.V.M. para entender los patrones de comportamiento de los valores bursátiles; sin embargo, gracias a la digitalización de la sociedad, han aparecido nuevas herramientas que amenazan con dejar a las anteriores obsoletas [2]-[4].

El objetivo de este proyecto es desarrollar un algoritmo que maximice el uso de las nuevas tecnologías e incorpore aquellas características no cuantitativas que influyen en el valor de la acción, pero que no suelen ser tenidas en cuenta en el análisis técnico.

2. Estado de la técnica

A lo largo de la historia se han desarrollado muchos modelos para aconsejar la compra o la venta en el mercado de valores, pero hay dos que han destacado por encima del resto y que se siguen utilizando en la actualidad: el análisis fundamental y el análisis técnico [5], [6]. Estos modelos analizan el valor y la evolución de las cotizaciones bursátiles utilizando enfoques muy diferentes y, por tanto, herramientas distintas.

El análisis fundamental se define como una metodología financiera y contable que estudia la rentabilidad de una inversión a través de los estados financieros [7]. El objetivo principal del análisis es generar una estimación precisa del valor que debería tener la entidad a través de una valoración de diferentes parámetros. Entre muchos elementos, algunos de los factores que se analizan son las métricas financieras de la empresa (activos, pasivos, beneficios, etc.), la eficacia de la gestión e indicadores más amplios como las previsiones del PIB, la evolución de los tipos de interés, el diferencial de la deuda y la estabilidad política del país. El objetivo final es determinar el valor razonable de la empresa y si está sobrevalorada o infravalorada.

Por otro lado, el análisis técnico se basa en la revisión de datos históricos para predecir el comportamiento futuro del mercado [2]. Se basa en la noción de que, en circunstancias similares, la historia se repite y lo hace en patrones reconocibles. Este enfoque se ha visto fuertemente reforzado en los últimos años con la llegada de las nuevas tecnologías, ya que han mejorado la capacidad de identificar indicadores de comportamiento y, por tanto, de aumentar la capacidad de previsión [8].

Con la proliferación de las nuevas tecnologías, se han desarrollado numerosos modelos de predicción de activos [9], [10]. Estos nuevos modelos pueden clasificarse en tres grupos: modelos lineales, modelos no lineales y modelos híbridos. Todos estos métodos diferentes permiten mejorar las capacidades computacionales de los modelos, pero el avance que aportaron fue la posibilidad de mejorar el rendimiento mediante el uso de variables de entrada más complejas, como factores macroeconómicos u otros indicadores relacionados con el mercado o el sector en el que opera la empresa [11], [12].

Además de lo anterior, hoy en día también se está empezando a tener en cuenta el impacto de las redes sociales en el rendimiento de las acciones. Esto se debe a que se ha evidenciado que los eventos que ocurren en estas plataformas mediáticas (publicaciones de influencers, actividad en foros de chat, etc.) pueden desencadenar cambios en las cotizaciones bursátiles [3], [13]-[15].

3. Metodología

El proceso seguido para la ejecución de este proyecto, tuvo tres fases principales. La primera fase consistió en un proceso de formación en el que el alumno asistió a varios cursos para aumentar sus conocimientos en relación con el área de trabajo. Estos cursos fueron impartidos por las universidades de Michigan y UNIMORE (Módena y Reggio Emilia) y la empresa DeepLaerning.ai. El alumno tardó aproximadamente 3 meses en realizarlos todos.

Una vez completados los cursos, lo primero fue establecer los requisitos que debían cumplir las bases de datos (cantidad, variedad, tipología, etc.). Se trata de una decisión clave, ya que, en función de la salida deseada y de los datos disponibles, hay que programar el código de una manera u otra. Una vez establecidos los criterios, el siguiente paso fue descargar los datos utilizando una plataforma de código abierto e iniciar los procesos relativos a la depuración de la información descargada.

Finalmente, la última etapa de este proceso consistió en la creación y validación de tres modelos diferentes: un Modelo de Regresión Logística, una Red Neuronal Monocapa y una Red Neuronal Profunda. Estos tres modelos se diferenciaban entre sí por la complejidad del cálculo de los datos. La razón para desarrollar diferentes algoritmos era validar cuál de ellos interpretaba mejor los datos para generar las predicciones más precisas. Durante la construcción de los modelos, fue necesario ajustar las diferentes configuraciones (optimización de hiperparámetros) para garantizar los mejores resultados de cada uno.

Para facilitar la comprensión de la metodología seguida a lo largo del proyecto, el siguiente diagrama de flujo muestra los principales pasos seguidos. Ayudará a tener una visión general del curso del estudio.

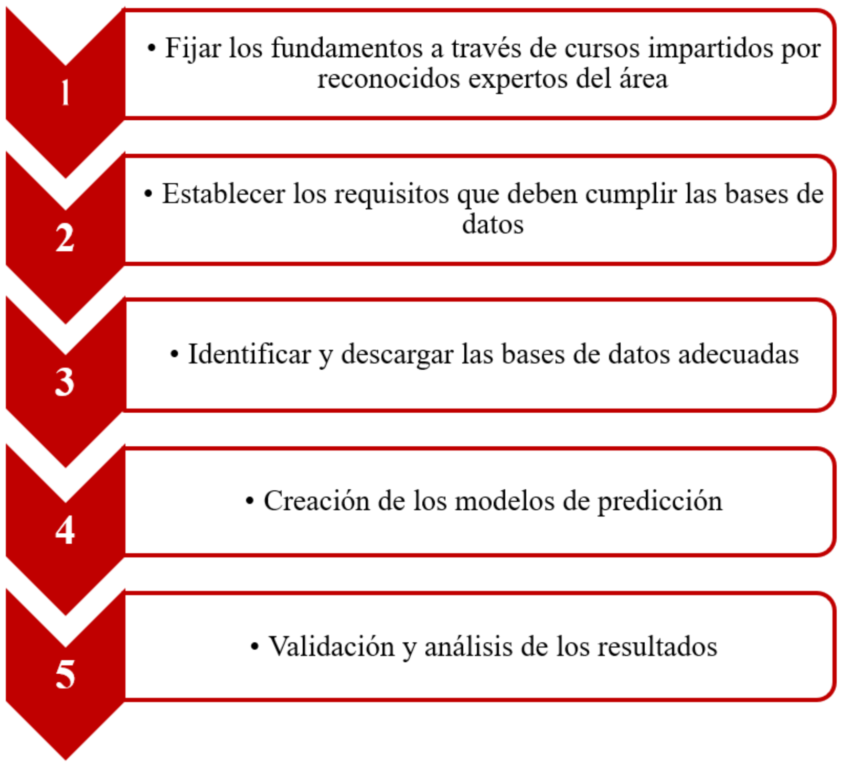

1
• Fijar los fundamentos a través de cursos impartidos por reconocidos expertos del área
2
• Establecer los requisitos que deben cumplir las bases de datos
3
• Identificar y descargar las bases de datos adecuadas
4
• Creación de los modelos de predicción
5
• Validación y análisis de los resultados

Ilustración 1: Resumen de la metodología

4. Resultados

El comportamiento de los modelos en las diferentes etapas del proceso de optimización se muestra en los siguientes resultados:

		Logistic Regression	**One Layered Neural Network**	**Deep Layered Neural Network**
Stock values	Initial Computation	72.7/73	74/70.8	54.1/61
	Number of reference days	81.8/77.6	85.5/82.3	92.2/89.6
	Number of iterations	81.8/80.6	87/83.8	92.2/89.6
	Final Computation	81.8/80.6	90.8/88.3	92.2/89.6
Sentiment scores	Initial Computation	54.4/61	57/60	54/62
	Number of reference days	67.8/70.1	88/87	92.9/91
	Number of iterations	68/70.2	88/87	92.9/91
	Final Computation	69.1/68.4	88/87	92.9/91
Stock values Sentiment scores	Initial Computation	56.4/54	71/54	53.8/60.4
	Number of reference days	80.6/79.5	92/88	93.5/92
	Number of iterations	80.6/79.5	92/90	93.5/92
	Final Computation	80.6/79.5	92/90	93.5/92

Ilustración 2: tabla de resultados

Como se muestra en el gráfico, para cada etapa se presentan dos valores: el primero representa el porcentaje de eficiencia que alcanza el modelo cuando se pone a prueba con los datos con los que ha sido entrenado (rendimiento del training set), mientras que el segundo representa el porcentaje de eficiencia del modelo cuando se pone a prueba con datos nuevos que no ha visto antes (rendimiento del test set). Mediante el análisis combinado de ambos valores, se puede determinar la eficacia de los distintos modelos.

Para elegir el mejor modelo, se utilizan dos criterios fundamentales: por un lado, se trata de conseguir eficacias cercanas al 100%; por otro, también es necesario que ambos valores sean muy altos. Si el rendimiento del conjunto de entrenamiento es del 98% pero el del conjunto de prueba es del 80%, significa que el modelo está memorizando los valores con los que se ha entrenado, pero luego es inútil con datos que no ha visto antes. Este problema se conoce como overfitting y debe reducirse siempre para garantizar la mayor eficacia de los modelos.

5. Conclusiones

Teniendo en cuenta todos los resultados obtenidos, se puede concluir que el modelo que mejor predice el valor futuro de una acción es la Red Neuronal Profunda con dos capas y dos unidades ocultas en cada capa, combinando datos numéricos y no numéricos.

Además de determinar qué modelo era el más recomendable, otro de los objetivos del trabajo era determinar la mejor fuente de datos para generar las mejores predicciones de cada sistema. A partir de los resultados obtenidos, se puede concluir que la fuente de datos más elaborada (valores bursátiles junto con calificaciones de sentimiento) ofrece los mejores resultados. Sin embargo, lo interesante es que la segunda fuente de datos más útil no fueron las calificaciones de sentimiento, sino los valores bursátiles; la razón de ello es que, al ser una información más sencilla, a los distintos modelos les resultó más sencillo comprender la relación entre las matrices de entrada y, por tanto, predecir los comportamientos futuros con mayor precisión. Estos resultados son muy interesantes porque refuerzan la base teórica del análisis técnico, que sugiere que los mercados bursátiles se mueven según patrones reconocibles y, por lo tanto, no sólo son impulsados por acciones externas como las publicaciones en las redes sociales [6].

Finalmente, la última observación relevante es el impacto de los diferentes hiperparámetros y cómo su optimización ayuda a mejorar considerablemente el modelo. Tras analizar el efecto de cada uno de los hiperparámetros por separado, se concluye que el elemento que más influye en el rendimiento de los diferentes modelos es claramente el número de días previos a tener en cuenta. Esto tiene sentido, ya que en muchos estudios sobre el rendimiento de los modelos mediante el ajuste de hiperparámetros se afirma que la cantidad de datos de entrada es uno de los hiperparámetros más importantes, si no el más importante [16]-[18]. Se corrobora entonces que el aumento del número de pares de datos para cada uno de los ejemplos de entrenamiento promueve y facilita el reconocimiento de patrones.

6. Referencias

[1] M. R. Vargas, C. E. M. Dos Anjos, G. L. G. Bichara, and A. G. Evsukoff, "Deep Leaming for Stock Market Prediction Using Technical Indicators and Financial News Articles," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2018-July, Oct. 2018, doi: 10.1109/IJCNN.2018.8489208.

[2] A. Lo and J. Hasanhodzic, "The evolution of technical analysis: financial prediction from Babylonian tablets to Bloomberg terminals," 2010, Accessed: Aug. 17, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=aJv9DwAAQBAJ&oi=fnd&pg=PR7&dq=technical+analysis+munehisa+homma&ots=crkJzWRQwU&sig=GR_Z-fHu7JjQY5VwsK3ObH8oJ0c.

[3] W. Jiang, "Applications of deep learning in stock market prediction: Recent progress," *Expert Syst. Appl.*, vol. 184, p. 115537, Dec. 2021, doi: 10.1016/J.ESWA.2021.115537.

[4] A. E. de Oliveira Carosia, G. P. Coelho, and A. E. A. da Silva, "Investment strategies applied to the Brazilian stock market: A methodology based on Sentiment Analysis with deep learning," *Expert Syst. Appl.*, vol. 184, p. 115470, Dec. 2021, doi: 10.1016/J.ESWA.2021.115470.

[5] B. G. Inchausti, C. R. Maya, and M. A. Gisbert, "El papel del análisis fundamental en la investigación del mercado de capitales: Análisis crítico de su evolución," *Rev. Esp. Financ. y Contab.*, vol. 31, no. 114, pp. 1111–1150, 2002, doi: 10.1080/02102412.2002.10779470.

[6] J. Chen, "Essentials of technical analysis for financial markets," 2010, Accessed: Aug. 17, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=202I0ZWA6tIC&oi=fnd&pg=PR13&dq=technical+analysis+of+the+financial+markets&ots=kj5ZIs13C4&sig=x4plLnpCU4MvDr0cl5OCecYVuK8.

[7] B. Graham, D. Dodd, and S. Cottle, "Security analysis," 1934, Accessed: Aug. 17, 2021. [Online]. Available: https://bridgecollege.net/download/75/accounting-and-finance/6485/00899.pdf.

[8] H. Liu, J. H.-E. S. with Applications, and undefined 2010, "Forecasting S&P-100 stock index volatility: The role of volatility asymmetry and distributional assumption in GARCH models," *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417409010689.

[9] Y. Zhang, L. W.-E. systems with applications, and undefined 2009, "Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network," *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800852X.

[10] S. Chaigusin, … C. C.-2008 I., and undefined 2008, "The use of neural networks in the prediction of the stock exchange of Thailand (SET) Index," *ieeexplore.ieee.org*, pp. 670–673, 2008, doi: 10.1109/CIMCA.2008.83.

[11] D. P.-T. and E. D. of and undefined 2010, "Macroeconomic indicators and their impact on stock market performance in the short and long run: the case of the Baltic States," *ceeol.com*, vol. 16, no. 2, pp. 291–304, 2010, doi: 10.3846/tede.2010.19.

[12] K. Hussainey and L. Khanh Ngoc, "The impact of macroeconomic indicators on Vietnamese stock prices," *J. Risk Financ.*, vol. 10, no. 4, pp. 321–332, Aug. 2009, doi: 10.1108/15265940910980632/FULL/HTML.

[13] A. Anta, L. Chiroque, P. M.-… del lenguaje natural, and undefined 2013, "Sentiment analysis and topic detection of Spanish tweets: A comparative study of of NLP techniques," *journal.sepln.org*, 2013, Accessed: Aug. 11, 2021. [Online]. Available: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/download/4658/2760.

[14] D. Valle-Cruz, V. Fernandez-Cortez, A. López-Chau, and R. Sandoval-Almazán, "Does Twitter Affect Stock Market Decisions? Financial Sentiment Analysis During Pandemics: A Comparative Study of the H1N1 and the COVID-19 Periods," *Cogn. Comput. 2021*, vol. 1, pp. 1–16, Jan. 2021, doi: 10.1007/S12559-021-09819-8.

[15] Y. Xu and S. B. Cohen, "Stock Movement Prediction from Tweets and Historical Prices," *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 1970–1979, 2018, doi: 10.18653/V1/P18-1183.

[16] "Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization | Coursera." https://www.coursera.org/learn/deep-neural-network?specialization=deep-learning (accessed Aug. 14, 2021).

[17] M. Feurer, F. H.-A. machine learning, and undefined 2019, "Hyperparameter optimization,"

library.oapen.org, Accessed: Aug. 19, 2021. [Online]. Available: https://library.oapen.org/bitstream/handle/20.500.12657/23012/1007149.pdf?sequence=1#page=15.

[18] M. Parsa, J. Mitchell, … C. S.-F. in, and undefined 2020, “Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design,” *frontiersin.org*, Accessed: Aug. 19, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00667/full?report=reader.

Contents

List of Figures

List of Tables

Chapter 1. INTRODUCTION

For many years now, economists, mathematicians and stock market experts have devoted much time and resources to stock value prediction. The key to success lies in understanding what are the key elements that influence the stock and how they affect it.

Traditionally, data analysis methods such as decision trees, logistic regression or S.V.M. have been used to understand the behavioural patterns of information, however, thanks to the digitalization of society, new tools have appeared that threaten to render these obsolete.

The aim of this project is to develop an algorithm that maximizes the use of new technologies and incorporates non-quantitative features that influence the value of the share, but are not usually taken into account in technical analysis.

1.1 MOTIVATION

When talking about stock analysis, there are two principal techniques: fundamental analysis and technical analysis. Both of them help traders and investors make buying and selling decisions, but each one takes a different approach.

The fundamental analysis is a long-term approach that focuses on the determination of the "fair market" value of the share based on company's financial statements, any report released by the enterprise as well as on micro and macro indicators. By doing so, it can be concluded that a specific stock is undervalued (and therefore stock should be bought) or overvalued (hence stock should be sold). This type of analysis requires a great deal of knowledge of the sector in which the company operates as well as a tremendous analytical skill to determine which elements are most significant and which indicators should be given more relevance.

On the other hand, the technical analysis is a short-term approach that looks at the past and present price of the share and its negotiated volume. The main objective is to predict future values taking into account its track record. This approach focuses less on economic theories and more on the mathematical analysis of stock behavior patterns.

Both techniques seek to provide a recommendation based on sound criteria, however, the main difference is that one will indicate whether or not a stock should be bought (or sold) whereas the other will indicate the value it would have and let the investor decide whether or not to invest.

Although both techniques are highly recommended and useful, technical analysis is currently being used more frequently. The reasons for this are mainly based on the fact that not everyone has the ability to properly interpret the documents issued by the companies, this mainly has the consequence that risks can be overlooked and this can lead to large losses of money. In addition, the algorithms used today for technical analysis are becoming more and more accurate and take into consideration a greater number of parameters that allow for more robust predictions.

Moreover, as markets change, so do investment strategies. Investors are increasingly betting on faster capital inflows and outflows in order to generate profits in less time by taking advantage of stock market fluctuations. This shift towards more aggressive strategies is resulting in an increased demand for technical analysis as it focuses on short-term stock analysis. As a consequence of these changes, the need arises to cover the interests of investors and, together with the digitalization process, this has led to a significant development of the tools used for technical analysis. Among the major advances is the use of Deep Learning and more specifically the use of Deep and Bidirectional Neural Networks for the prediction of securities.

However, there is still much debate about the effectiveness of these new tools. Experts explain that as technology keeps on developing in this society (due to digitalization), there is more data available and therefore a greater chance of improving the efficiency of the forecasting models. The problem is that traditional methods (independently of the quantity of data) can only reach a certain performance; in small training sets they are very efficient, sometimes even better than Deep Neural Networks, however, when large sets of data are available, they are unable to take advantage of the opportunity to improve performance.

Be that as it may, this is not the case of the Neural Networks, since there is a proportional relationship between the quantity of data and the performance; the more data you have, the better the algorithm will do. Even so, it is important to bear in mind that not any Neural Network can withstand any amount of data, depending on the characteristics of the algorithm a certain volume of data will be recommended.

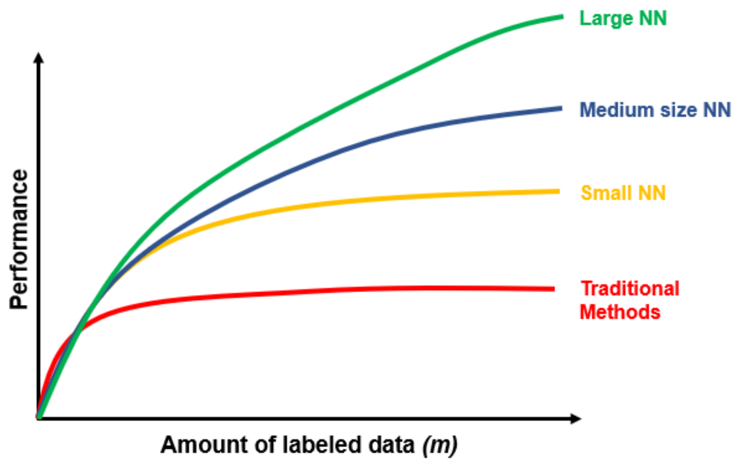


Figure 1-1: performance of different forecasting methods based on the quantity of data.

Finally, there is one more notion that has to be discussed. Although the amount of data is a major element in determining the effectiveness of the prediction, there are others that influence it as well. Among these is the use of non-numerical data. Recent studies [1]–[3] suggest that news published in newspapers or social media should be taken into account. The investigations conclude that negative (or positive) news regarding the performance of a company will incite people to sell (or buy) accordingly and therefore, will definitively impact its stock price.

Moreover, this project will be done using only opensource databases, to prove that in order to achieve good performance in the algorithm it is not necessary to use private databases that only those who pay have access to. The project wants to inspire and incite others to start similar projects without the fear of having insufficient data available.

1.2 *PROJECT'S OBJECTIVE*

In this project, these previous notions will be put to the test. The aim is to determine the influence of different types of data (such as the information provided by mass media or stock market values) on the price of market stock and consequently decide whether, and how much, they should be considered when forecasting the prices.

Through this project the goal is to provide a better understanding of fluctuations of the stock market and help creating more accurate algorithms by proposing new types of inputs. Another objective is to raise awareness of the importance of mass media and remind their ethic responsibility, as they can easily unbalance the market resulting this in negative repercussions to the companies affected and ultimately to the country's economy.

1.3 *SUMMARY OF METHODOLOGY*

It is necessary to take into account that the level of knowledge on behalf of the student with respect to the world of Deep Learning and programming in the Python language before the beginning of this project was completely null. For this reason, the first step, before the creation of the different models that will be explained later, was to receive professional training in the technological areas covered by the project.

The first step of the student's training consisted of attending the course "Introduction to Machine Learning" imparted by the tutor of this project, Marco Lippi. This course was part of the curriculum for students studying for a PhD in Industrial Innovation Engineering and consisted of a brief introduction to the world of Artificial Intelligence as well as an overview of the most frequently used prediction models and the state-of-the-art ones nowadays.

After this first course, two other specialized programs would follow in which helped the student to acquire the skills that would be used to program and execute the different algorithms used in this project. The first specialized program consisted in 3 courses offered by the University of Michigan (via Coursera) and they were focused on the development of skills related to the use of Python and data structures as well as the use of Python to access web data. The second specialized program was the most important one, it was imparted by the company Deeplearning.ai (via Coursera) and it consisted on 5 courses that analysed in depth all the concepts connected with Deep Learning, the structuring of Machine Learning projects, Deep and Convolutional Neural Networks as well as hyperparameter tuning and model optimization. This learning process of the student required approximately 3 months (November-January).

Once the courses had been completed, the first thing to do was to establish the requirements to be met by the databases (quantity, variety, typology, etc.). This decision is very important because, depending on the available data, the code has to be programmed in one way or another. In the case of this project, there was particular interest in obtaining two different types of data (but in the same time interval); on the one hand there was interest in the numerical evolution of the share value, while on the other hand the plan was to collect a fraction of the Twitter posts (tweets) that were published commenting on the situation of the company under study and that of its direct competitors.

Once the databases had been downloaded and the database information met the requirements previously established, the creation of the predictive models could begin.

The intention of the project was to analyse the predictive capacity of the different algorithms by varying the amount and complexity of the data as well as the sophistication of the models themselves. With these variations it was sought to understand what were the optimal performance conditions for each model. For this, the methodology would be based on a process in which initially an algorithm is chosen and its performance is studied when working with a single type of parameters (e.g., numerical), in this analysis it will also be modified hyperparameters such as the learning rate, the number of training epochs or whether or not to take into account the competitors; these modifications will be very useful to see how the model responds (efficiency-wise). Next, the process is repeated but now with the other database (if one continued with the example, it will now be the tweets) and finally it is repeated again but now with the two types of data simultaneously. This process will be repeated for all the different algorithms and the results will be analysed to determine which algorithm works best in each case. The following table will summarize all the different analysis carried out:

Model	**Data type**	**Hyperparameter tuning**
Logistic Regression	Numerical	Yes
Logistic Regression	Text	Yes
Logistic Regression	Numerical & Text	Yes
One-layered Neural Network	Numerical	Yes
One-layered Neural Network	Text	Yes
One-layered Neural Network	Numerical & Text	Yes
Two-layered Neural Network	Numerical	Yes
Two-layered Neural Network	Text	Yes
Two-layered Neural Network	Numerical & Text	Yes
Deep-layered Neural Network	Numerical	Yes
Deep-layered Neural Network	Text	Yes

Deep-layered Neural Network	Numerical & Text	Yes

Table 1-1: summary of process

In order to facilitate the understanding of the methodology followed throughout the project, the following flowchart with the key steps is added. It will be helpful to have a general overview of the course of the study.

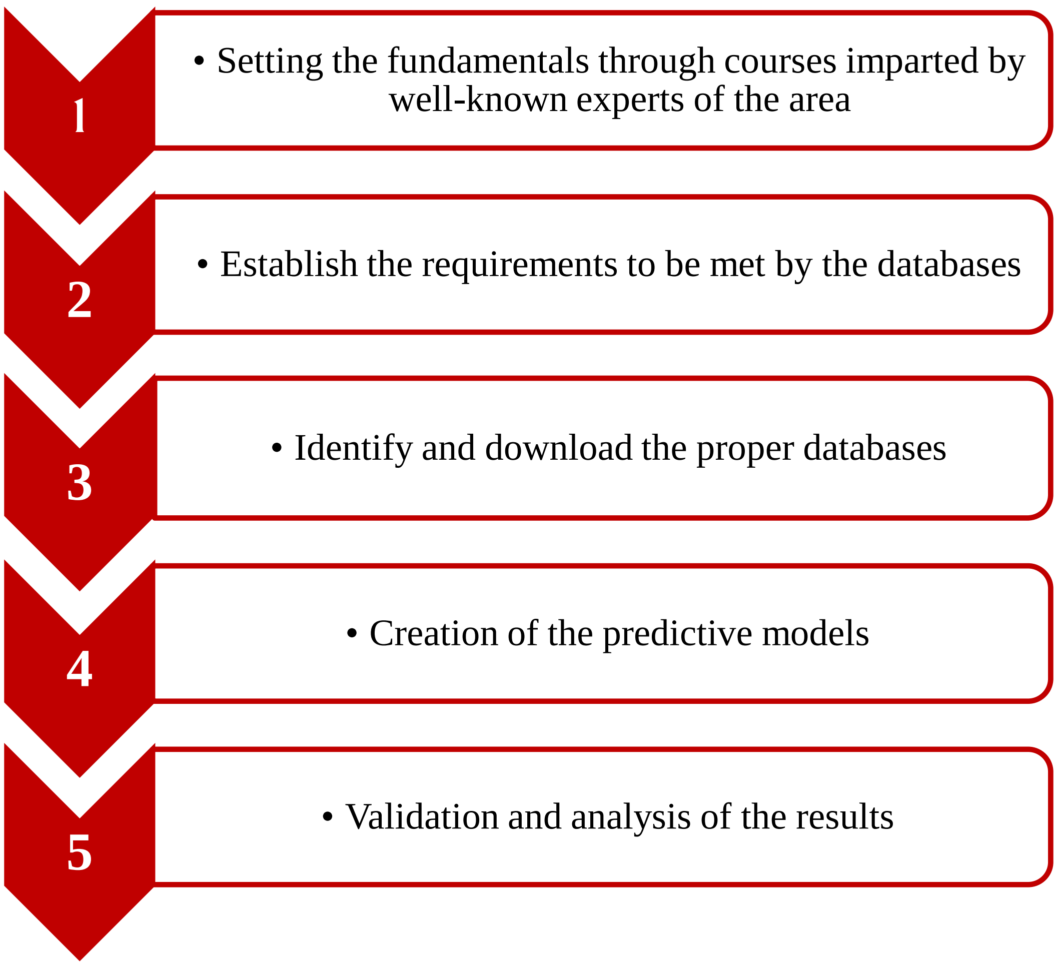

1
Setting the fundamentals through courses imparted by well-known experts of the area
2
Establish the requirements to be met by the databases
3
Identify and download the proper databases
4
Creation of the predictive models
5
Validation and analysis of the results

Figure 1-2 - Summary flowchart of methodology followed during project

Chapter 2. DESCRIPTION OF TECHNOLOGIES

To fully understand the process followed to complete the project, it is critical to know in advance the technologies used in the project. As it has been previously explained in Chapter 1.3, the computational models have been written using the Python programming language. However, it is not only the use of Python that adds value to the developed algorithms, but the combined use of this programming language together with other tools such as Neural Networks or Sentiment Analysis models. They are the ones that allow better conclusions to be drawn from the available information. The following is a brief outline of the theoretical foundations of the tools just mentioned.

Although it has been said before, it is important to remember that the underlying principle of all these algorithms is Machine Learning, therefore this concept will be used as a starting point.

The concept of Machine Learning (sometimes also referred to as ML) is nowadays among the most popular topics of discussion due to its wide range of action; the applications of Machine Learning go from image recognition to virtual personal assistants. The purpose of this chapter is to introduce and explain what Machine Learning is and what are its key elements. The concepts will be explained in a progressive order of complexity, beginning with the introduction to Computer Science and Artificial intelligence, continuing with Neural Networks and finishing with Deep Neural Networks and Recurrent Neural Networks. For a detailed and more extensive explanation of the concepts see [4][5].

2.1 MACHINE LEARNING

Computer Science is the discipline that focuses on the study of computers and the act of computing. It analyzes both theoretical and algorithmic foundations; it includes the study of, among many other areas, algorithms and data structures, the processing and storage of information, modeling data, information processes and the development of Artificial Intelligence (often referred to as AI). It is on this last one that the attention will be placed in as it is the area that focuses on the creation of "smart" machines that are able to perform activities and tasks that generally would depend on human intelligence.

Once the notion of AI has been explained, it becomes easier to understand the role of ML and its connection to Artificial Intelligence. Machine Learning comprises all the computational algorithms that seek to extract patterns from the input data, through complex statistical analysis, in order to forecast (within a specific range) the output of future inputs.

It is important to notice that ML differs from traditional computational approaches as in the last one it is the programmer the one that has to decide the "rules" or boundaries for the algorithm to follow; in other words, the algorithm simply executes the instructions. On the other hand, the ML algorithms are the ones that create the rules; the programmer inputs the data and it is the algorithm the one that decides how the data inserted is interrelated.

Inside ML, there are three main learning paradigms: Supervised Learning, Unsupervised Learning and Reinforcement Learning.

The Supervised Learning subcategory is characterized by the use of inputs that have previously undergone a classification procedure (what is known as "labeled datasets"). Once the data has been inserted, the algorithm will focus on adjusting the weights associated to each input until the model has been fitted appropriately. The information provided (during the training and test phases) consists in inputs and its associated output, this makes possible the determination of the patterns over time. Algorithms typically used in this process are, among many others, Neural Networks, Logistic Regression, Support Vector Machines (SVM) or K-nearest neighbor.

Another subdivision of ML is the Unsupervised Learning, this one is known for using "unlabeled datasets", this means that it is the own algorithm the one that decides what are the patterns and the groupings of the data (there is no need for human intervention). This characteristic makes these models ideal for tasks like clustering, association, dimensionality reduction or anomaly detection.

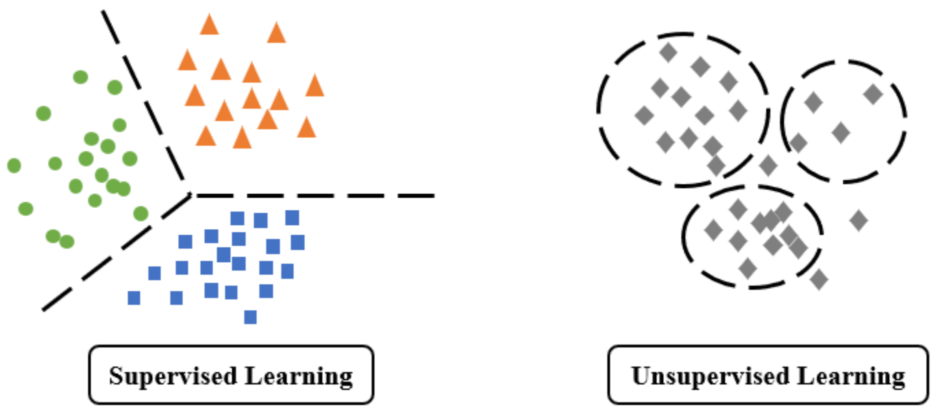


Figure 2-1: supervised vs. unsupervised learning

As one can see in Figure 2-1, in the Supervised Learning model the data is perfectly divided into the different subgroups. However, the Unsupervised Learning approach does its own clustering and this results in the possibility of not recognizing a group for all the data.

Finally, the Reinforcement Learning is a model that aims to teach algorithms to make decisions or their own, focusing on the development of intelligent solutions to complex control problems. It is similar to Supervised Learning but the algorithm does not use sample data. Instead of feeding data to the algorithm, the information is generated by the algorithm in a trial-and-error process where the results are constantly being labelled. A very common area where this technique is used is in videogames, as it is a simulated environment. Other areas where this model is used are those regarding autonomous driving of vehicles or robotic process automation.

In this project, the focus will be placed on the Supervised Learning subgroup. It will now be explained in major depth and the different algorithms that can be used with this model depending on the problem will be explained.

2.1.1 COST AND LOSS FUNCTIONS

Focusing now on the Supervised Learning area, one of the most important concepts to be explained is the *cost function*. As mentioned above, in this type of learning there is an already classified database in which one knows the output that corresponds to each input. From now on, the algorithm will begin looking for patterns that explain the relationship between the pairs of data imputed in order to be able to, using the conclusions reached earlier in the processing, forecast output values for unclassified datasets. In order to do so, many different algorithms can be used, some of them are: linear regressions, logistic regressions, decision trees or Support Vector Machine (SVM).

In order to make the explanation of the function as easy to understand as possible, the algorithm that will be used will be that of a linear regression that depends only on one variable *x*.

With this formula the objective is to predict the outputs from a linear equation relating the input as follows:

$$\hat{y} = (\theta_0 \ + \ \theta_1 \cdot x) \tag{2.1}$$

Both parameters θ_0 *and* θ_1, also known as “weights”, will be estimated in order to formulate the line that best explains the reality of the data imputed. For explaining purposes, the following picture will be used to simplify the explanation.

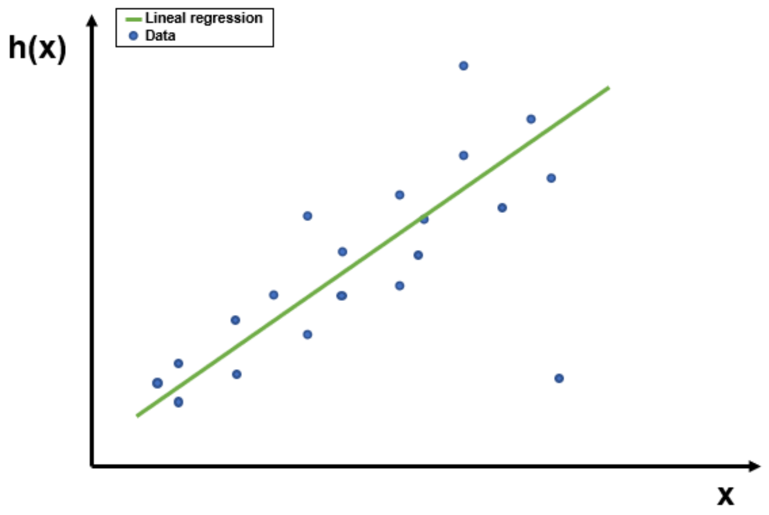


Figure 2-2: linear regression

In Figure 2-2 there are *m* points, for each one of them there will be a value of *x*, a value of *y* and a value of ŷ. It is important to notice that the values of *y* and ŷ are both represented in the *y* axis, the reason for this is that ŷ represents the estimated value for all the different values of *x*; however, *y* represents the real value of each of the *m* points.

The main purpose of the cost function can be to either:

- **Minimize** the difference that exists between *y* and ŷ for the given values of *x*. The returned value will be called *cost.*
- **Maximize** the difference that exists between *y* and ŷ for the given values of *x*. In this case the output will be called *reward*.

For this project the method used will be the minimization procedure.

Before the implementation of the cost function, it is necessary to explain what the loss, or error, function is. It is the key element of the cost function as it is the one that indicates how well the algorithm is performing for each data input. Once the loss has been calculated for each input, the cost function is simply the average of all the losses. In other words, for a single cycle, the loss function is calculated numerous times whereas the cost function is only calculated once.

There are many equations that can be used for the loss function (MSE, RMSE, MAD, etc.). However, depending on the problem at hand, some of them can be more effective than others. In current situation (linear regression), the equation used is the Mean Squared Error (MSE) as it is the one that best represents the errors for this kind of regressions. The MSE equation is the following:

$$L(\hat{y}, y) = \frac{1}{2} \cdot (\hat{y} - y)^2 \tag{2.2}$$

Therefore, the cost function for this situation will be:

$$J(\theta_0,\, \theta_1) = \frac{1}{2 \cdot m} \cdot \sum_{i=1}^{m} \left(\hat{y}\left(x^{(i)}\right) - y^{(i)}\right)^2 \tag{2.3}$$

It is important to point out that the superscript *i* is not an exponent, it is the notation used to indicate that, on each step of the addition, the squared difference computed refers to the same training example; they are the estimated and real values of the i^{th} element of the set.

The main objective of the algorithm will be to minimize the cost function *J*. In order to do so, the value of the weights will be continuously modified until the optimum ones are reached. The optimum parameters will provide the line that best explains the pattern of the inputs; this will then enable a more efficient forecasting of future inputs.

If the cost function was plotted as function of θ_0 and θ_1, one would obtain a 3-Dimnesional graph. In order to visualize this in two dimensions, contour plots will be used; to make it more visual, refer to Figure 2-3. The advantage of this method is that all the points with the same height (same cost function) will be represented with ellipses, this will make easy to determine what pair of weights should be used to obtain the minimum value of *J*.

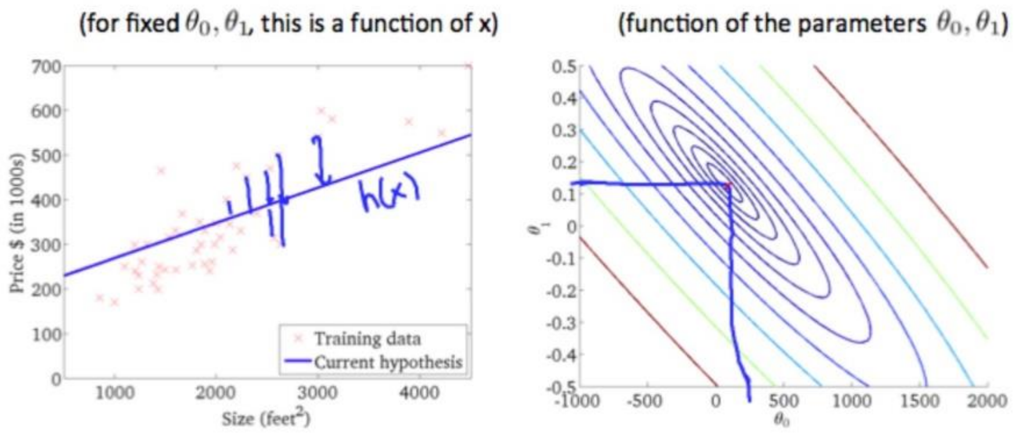


Figure 2-3: 2D representation of the cost function [14]

Normally, the number of input variables for each dataset will be much greater, therefore it will not be possible to represent the data in the 2D graph and the determination of the minimum value for J will be much harder. In order to solve this problem, an iterative procedure will be used to look for the global minimum of the function, this method will be known as *gradient descent*.

Before discussing the gradient descent, there is one last element that has to be introduced; the *Bayes error*. When optimizing the cost function, one will never be able to reach the value of zero. The reason for this is that there is always "noise" in the distribution and this makes even the perfect algorithm fail. For this reason, experts created a reference value that enabled others to measure the performance of their algorithm using a realistic reference; this reference is called the *Bayes error*, and it is lowest possible prediction error (the irreducible one).

In order to further understand this last concept, one could think of the following example; when flipping a coin everybody knows the process that generates the outcome. However, if one was to predict the outcome of a sequence, mistakes will still be made, this is due to the randomness of the process (i.e., stochastic).

2.1.1.1 Gradient descent

The gradient descent is one of the most used iterative algorithms in Machine Learning. It is a method that makes it possible to, in unbounded optimization problems (like the ones that will be seen in this project), train or learn the values of the weights in order to find the minimum value of the cost function *J*. Graphically, the Gradient Descent could be represented like this:

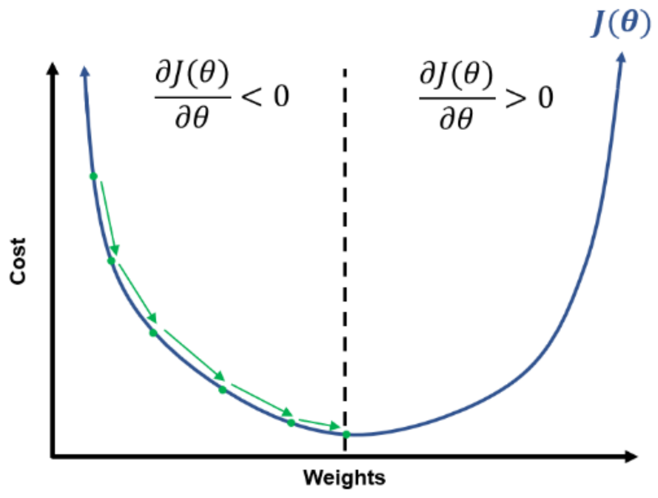

$J(\theta)$
$\frac{\partial J(\theta)}{\partial \theta} < 0$
$\frac{\partial J(\theta)}{\partial \theta} > 0$
Cost
Weights

Figure 2-4: gradient descent optimization

This algorithm consists in a series of iterations in which the value of the weights (θ) is being modified aiming to reduce the value of *J*. The gradient descent will start at a point A and will move in the steepest downhill direction until the global minimum is reached. The equation that represents the descent is the following:

$$\theta_j := \theta_j - \alpha \cdot \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_j}, \qquad j = 0, 1 \tag{2.4}$$

The main idea of the equation is that, for a certain value of the variables to be optimized, there is a movement towards another value of these variables in the direction of maximum change of the cost function, i.e., the direction of the gradient.

In this particular case, the only parameters present are θ_0 and θ_1. Therefore, the values for j will only be 0 and 1.

Moreover, another element that must not go unnoticed is the learning rate α; this is the hyperparameter that determines the "size" of the update. In other words, small values of α will involve a slow gradient descent (less aggressive), resulting this in the need of many iterations in order to get to the minimum value of the cost function. On the other hand, very high values of α may trigger problems such as non-convergence. For this reason, it will be necessary to find an intermediate value of the learning rate that enables a reach the optimum value of the weights (θ_0 and θ_1). This technique is known as Hyperparameter Tuning, and will be explained in more detail later on this document.

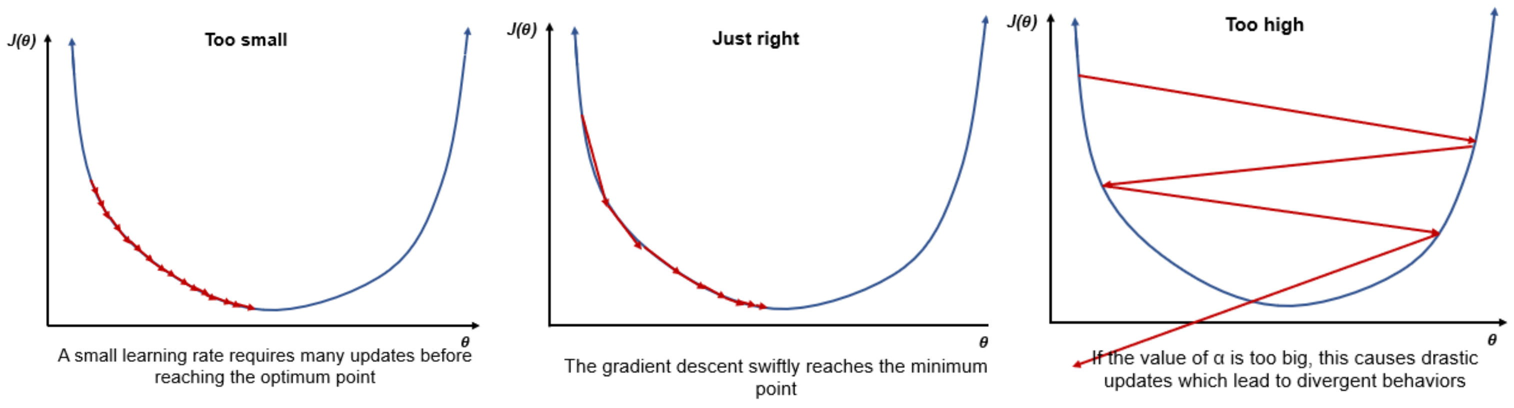

J(θ)
Too small
θ
A small learning rate requires many updates before reaching the optimum point
J(θ)
Just right
θ
The gradient descent swiftly reaches the minimum point
J(θ)
Too high
θ
If the value of α is too big, this causes drastic updates which lead to divergent behaviors

Figure 2-5: Gradient descent for different values of α

Going back to the gradient descent equation (2.4), one can see that, as the equation of the cost function $J(\theta_0, \theta_1)$ is known, the partial derivative can now be developed. From now on, the equation for the cost function will be generalized; this means that, instead of solving the equation for 2 weights, the gradient descent will now consider n of them. In addition, it is important to mention that for each θ_j, there will be the corresponding x_j and, by convention, x_0 will always be equal to 1. With all this, after several operations, one obtains the following general equation of gradient descent:

$$\theta_j := \theta_j - \alpha \cdot \frac{1}{m} \cdot \sum_{i=1}^{m} \left(\hat{y}\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j{}^{(i)} \qquad \forall j \tag{2.5}$$

In practice, when n (the number of weights) is greater than two, it means that, for each data, there are n characteristics that will help to classify it.

A very common problem with the gradient descent is that if the inputs have been measured in different units (kilograms and millimeters for example), the computation of the gradient descent becomes too complex and this often results in poor performances in the optimization of the values of the weights. It is for this reason that the data is usually normalized.

The normalization of data is a very important technique and, as one can imagine, there are different ways to normalize the data. One way to do so is to scale each value by dividing it by the maximum of the total range $x_1' = \frac{x_1}{max}$. However, the procedure that will be used is the mean normalization:

$$x_j' = \frac{x_j - \mu_j}{s_j} \tag{2.6}$$

Where μ_j is the mean of all the data for the corresponding input and s_j is the one between the maximum and the minimum value of the data (the range of the data).

Apart from all the things explained previously, there is another challenge that one usually has to deal with whenever working with gradient descent; the possibility of using a local minimum instead of the global minimum. The problem lies in the fact that, if one falls into a local minimum, the algorithm will no longer have any path by which to minimize the function and will consider the point where it stands as optimal. To solve this problem, the algorithm will be initialized randomly several times. The reason to do this several times is that, since it is possible that the algorithm may fall into a local minimum, starting from another path in the next execution reduces the chances of happening again.

2.1.2 LOGISTIC REGRESSION

Until now, the cases discussed had as inputs values in a continuous range. However, for the interest of this paper, a new type of problems will be considered; the binary classification problems. The main difference is that the output from the network will be either discrete or even values. An example of this could be an algorithm that predicted whether an email should be classified as spam or not. For this, the output will be 1 in the case of spam and 0 in the opposite case.

To solve this kind of problems one of the best alternatives is the use logistic regression algorithms. For this reason, ŷ will be defined as:

$$\hat{y} = P(y = 1|x) \qquad\qquad 0 < \hat{y} < 1 \tag{2.7}$$

This means that ŷ is the probability of *y* being 1 given the input data *x*. Therefore, it is a value that can range from 0 to 1.

Given that *x* is an n_x dimensional vector, the parameters used will be an n_x dimensional vector of weights (*w*), together with a real number called the bias (*b*). Having this said, the most intuitive formulation of ŷ would be:

$$\hat{y}(x) = w^T \cdot x + b \tag{2.8}$$

This is the formula that one would use for linear regression. However, it cannot be used for logistic regression as, depending on the values inputted the output can be very big, very small or even negative. This not acceptable in the logistic regression as the values of the prediction can only go from 0 to 1. The best way to solve this problem is to use the sigmoid function:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.9}$$

Being *z*:

$$z = w^T \cdot x + b \tag{2.10}$$

This is a very effective algorithm as:

- If z is a large value, then: $\sigma(z) \approx \frac{1}{1+0} = 1$
- If z is a small value, then: $\sigma(z) \approx \frac{1}{1+Big\ Number} \approx 0$

The graphical representation of the sigmoid unction would be:

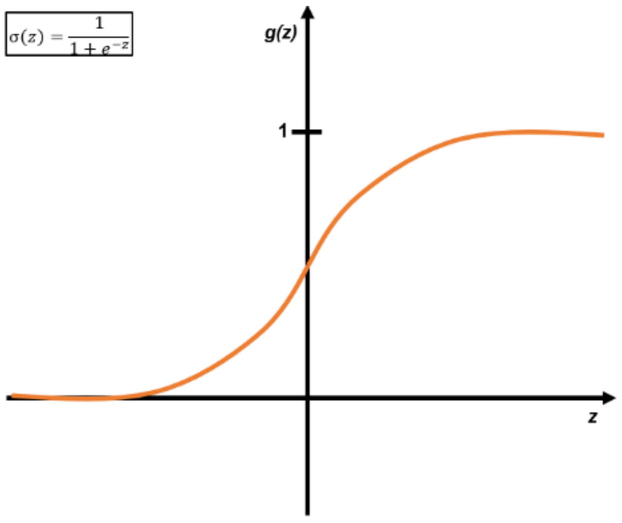


Figure 2-6: Sigmoid function representation

Another matter that must be addressed whenever working with binary classification problems is the cost function. When working with linear regression, the equation to be used is (2.3). However, if instead of imputing the linear equation, one inputted the sigmoid function of it in the Error Mean Squared, the cost function would not converge and therefore the gradient descent would not work. The solution for this problem will be to use another formula to compute the loss and cost function. The equation used is known as the *cross-entropy function* and the cost function would be:

$$J(w,b) = \frac{-1}{m} \cdot \sum_{i=1}^{m} \left[y^{(i)} \cdot \log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) \cdot \log\left(1 - \hat{y}^{(i)}\right) \right] \tag{2.11}$$

This method is used whenever there is a need to classify variables, as it enables to verify which of the actual values is closest to the value predicted. Thus, the output of this function gives values between 0 and 1, being 1 when the case when the preceded value is equal to the real one and 0 when there is no similarity between them.

The gradient descent, however, is exactly the same as the one previously (2.4), only modifying the linear ŷ(x), by the $\hat{y}(x)$ formed with the sigmoid.

After explaining the process for tackling binary classification problems, the next step is to present the multiclass classification problem. These two types pf classification, have numerous similarities and differ mainly in the output of our algorithm. In this case, instead of deciding between two classes, as in the case of an algorithm that classifies emails as spam or non-spam, there will now be more. Therefore, representing it graphically, it would look like the following image:

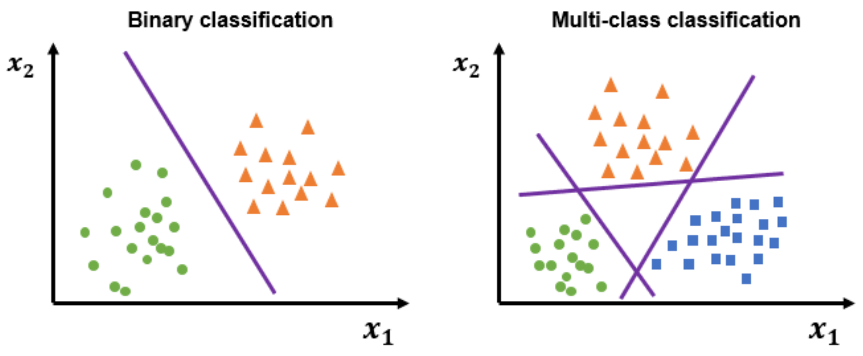


Figure 2-7: binary and multi-class classification

There are many ways to solve this problem. One of the most effective and used is the "one vs. all". This consists of selecting one of the classes and "confronting" it with the others; in other words, the multi-class classification will be transformed into several binary classifications. For example, if there are three classes (1, 2 and 3), one will first compare the first class with the rest. To do this, one will use a binary classifier that gives an output of 1 (in case it identifies and element of the first class) or 0 (in case it predicts another class). This process will be repeated for classes 2 and 3.

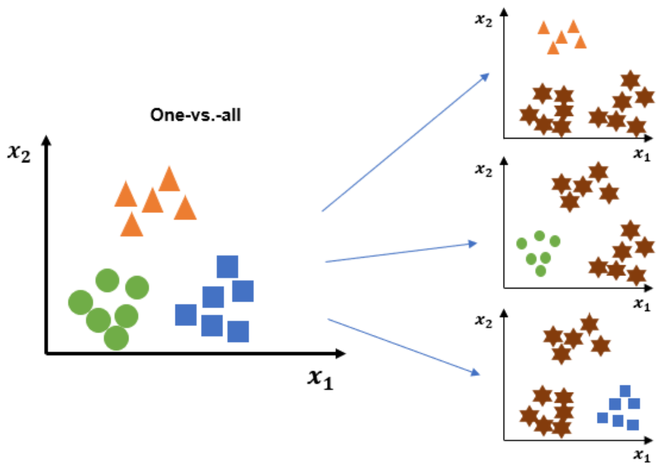


Figure 2-8: multi-class classification using "one vs. all" technique

2.1.3 FITTING PROBLEMS

To achieve good performance after running a Machine Learning algorithm, it is very important to find a suitable structure that is well-balanced with the amount of data available. In many situations, when the program does not work properly, it is due to a very complex architecture for a small amount of data or, on the contrary, an architecture that is too simple for a large amount of data. This gives rise to two of the main challenges one has to face when working with ML algorithms: overfitting and underfitting.

Overfitting: It is also known as high variance. It is one of the most common problems when it comes to the training of the data. What happens is that the machine over-adjusts to the input; this means that it directly memorizes the particular data inputted and is incapable of learning the patron of the inputs. If this is not taken into account, it is easy to fall into thinking that the algorithm is working perfectly, as it recognizes the input data perfectly and the cost function is minimum (exactly zero, because the prediction will not deviate from the input), but in reality, it is useless. An example where this happens is in a regression problem, when a regression line with a high degree polynomial function is used; what will happen is that the algorithm will fit the data as it is and will not adapt to new cases. The following image will allow to illustrate the concept just explained:

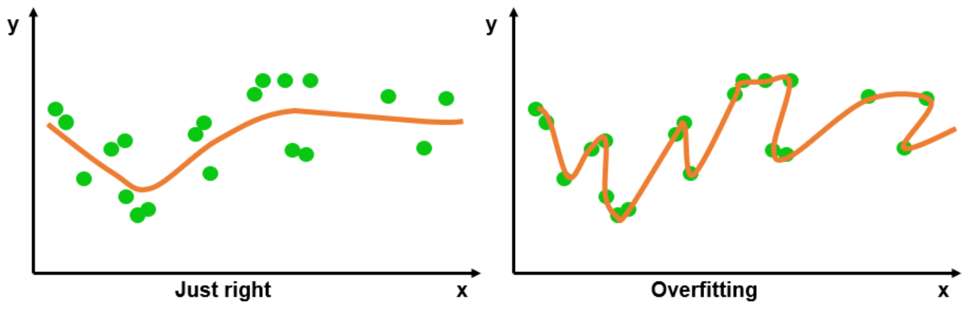


Figure 2-9: example of overfitted data

Underfitting: Also known as high bias. This is the complete opposite problem to overfitting. In this case, the algorithm is too simple compared to the complexity of the problem, so the model will not be able to correctly identify the patterns, thus outputting lousy results. In order to solve this problem, it is frequent to resort to more complex algorithms, the use of better "features" that help understand better the problem, or increase the number of

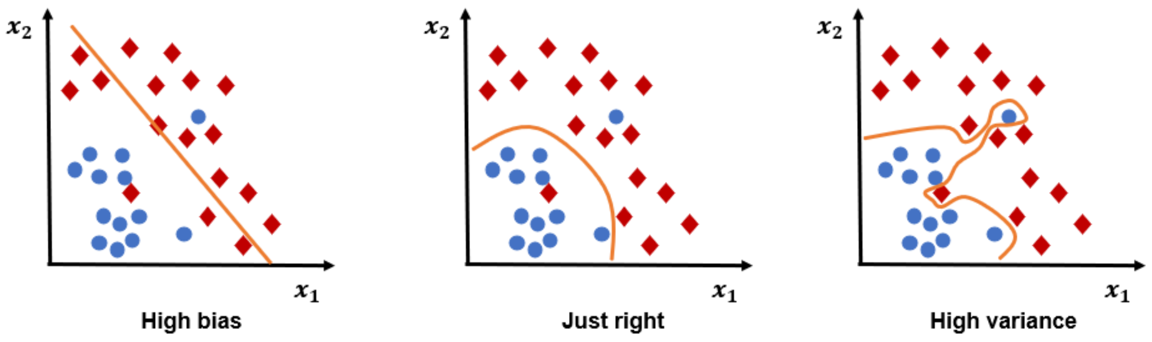


Figure 2-10: bias, variance and equilibrium

iterations.

Once the two fitting problems have been explained, a need arises to find a procedure to accurately and reliably detect these issues. A very effective way to detect these setbacks is to compare the training and development sets (they will be explained in the data structure section):

- If the training set error is very low (e.g., 1%) and the development error significantly higher (e.g., 11%) then it is a *high variance problem (Overfitting).*
- If both the training and the development errors are high (bad performance in both cases e.g., 15% and 16% respectively) then it is a matter of *high bias (Underfitting).*
- If the training set delivers a bad performance (e.g., 15%) but the development outputs an even worse (e.g., 30%) then there is *high bias and high variance (Underfitting and Overfitting).*
- If both the training and the development errors are very low (very good performance in both cases, e.g., 0.5% and 1%) then the algorithm is working perfectly.

It is important to notice that all these conclusions assume that the optimum error (the Bayes error) is approximately zero.

Whenever facing these difficulties, experts often recommend the following criteria to deal with them:

1. Initially check for high bias. If there happened to be a high variance problem, one could try augmenting the number of hidden layers and hidden units of the Neural Network, train the algorithm for a longer time or even try a different a different ML architecture.

2. Once the bias problem is solved, then the variance problem will be checked. Methods to solve it would be to increase the quantity of data used, to use regularization or even to try a different algorithm.

Depending on the problem, there will be methods that will help solve the problem more (or less) effectively. Therefore, it is very important to accurately identify the type of problem one is facing.

In the past experts thought that both bias and variance were depended on each other in a way that if one was increased the other one could be decreased. This meant that there was no possible way to obtain a very effective outcome since, at least, one of the two problems could not be reduced to the optimum values. This concept was known as the *bias/variance tradeoff*. However, there is a way to get low values for both indicators (a trade-off fit):

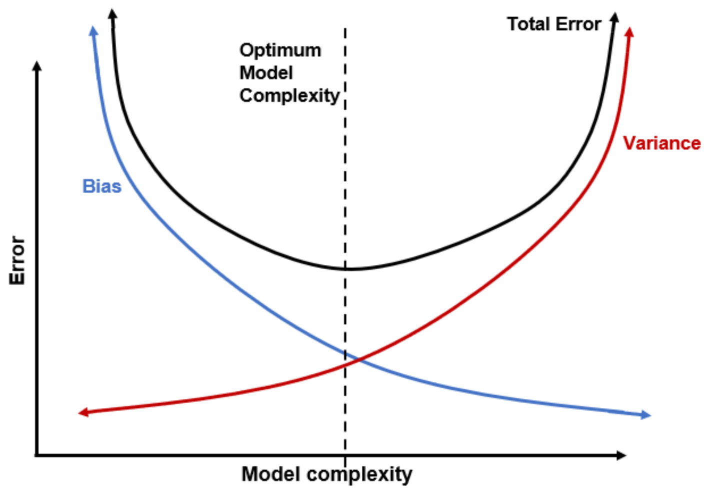


Figure 2-11: bias-variance tradeoff

2.2 NEURAL NETWORKS

2.2.1 INTRODUCTION TO NEURAL NETWORKS

2.2.1.1 Biological Neural Networks

The nervous system is made up of a network of neurons specialized in the reception, integration and transmission of information. The fundamental unit of this system is the neuron. In a way, the neuron is a simple information processor that is composed of:

- An input channel, the *dendrites*. They are short extensions of the neuron.
- A processor, the *soma*. It is the cellular body.
- An output channel, the *axon*.

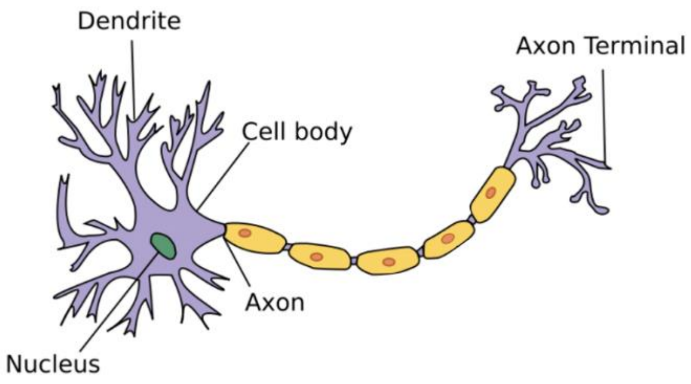


Figure 2-12: biological neuron [14]

Neurons have the ability to communicate accurately, quickly and over long distances with other cells (regardless whether they are nerve, muscle or glandular cells).

A single brain neuron can receive thousands of inputs and transmit the processed information to hundreds of outputs. This connection between neurons is known as a *synapse*. These are unidirectional connections, in which information is transmitted electrically inside the neuron and chemically between neurons, thanks to substances known as *neurotransmitters*.

The electric impulses begin at the *dendrites* and they travel through the neuron until the core is reached, here the information in processed and begins the process to generate a response. Once the response is created, the information is sent to the *axon*, this element is in charge of sending the response of the neuron to other cells. The connection between different neurons is known as *synapsis*.

When a pattern is recognized, a *steady state* is produced. The number of recognizable patterns (by a group of neurons) can be related to the number of neurons in the group and the probability of error in the recognition of that pattern.

When patterns are repeated, a rearrangement of neural connections occurs, which results in learning. This means that certain connections are strengthened while others are weakened (through neurotransmitters).

This behavior just described serves to introduce an analogy between biological and artificial Neural Networks. For a detailed and more extensive explanation of the concepts see [6], [7].

2.2.1.2 Artificial Neural Networks

Once the notion of biological Neural Network has been introduced, it becomes easier to explain how artificial Neural Networks work. To finish understanding them, one would have to go back to 1943, when scientists Walter Pitts and Warren McCulloch [8], [9], first proposed a very simple model of a biological neuron that later became known as an artificial neuron. It consisted of a neuron that had several inputs and a single output. Its mechanism was very simple, the output was activated when one or more of the inputs were active. With this, McCulloch and Pitts demonstrated that, with such a simple model, a more complex one could be created to perform any logical operation. Several examples can be seen in the following image:

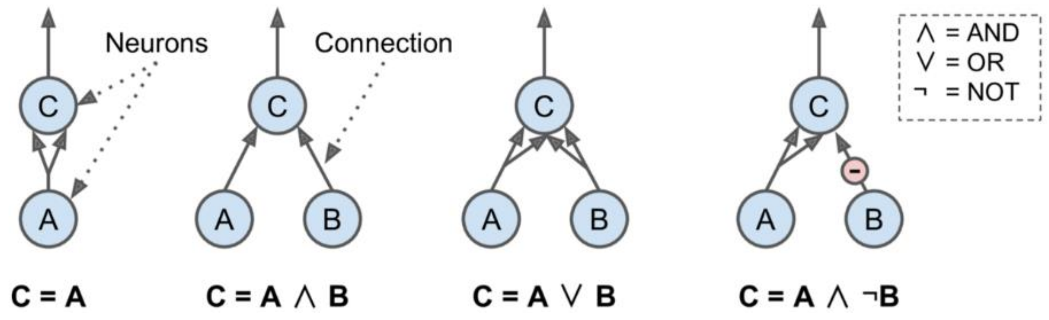


Figure 2-13: logic operators [18]

After this first contact, to better analyze its functioning, it is necessary to introduce the perceptron, one of the simplest artificial Neural Network structures, created by Frank Rosenblatt in 1957 [10]. A Neural Network can have numerous layers with many neurons. Moreover, the neurons in one layer can be interconnected with the neuron in the next layer in many different ways. In the simplest scenario, there are only two layers, input and output. Although, by convention, when describing the number of layers in a network, input is not usually counted, therefore the perceptron will only have one layer.

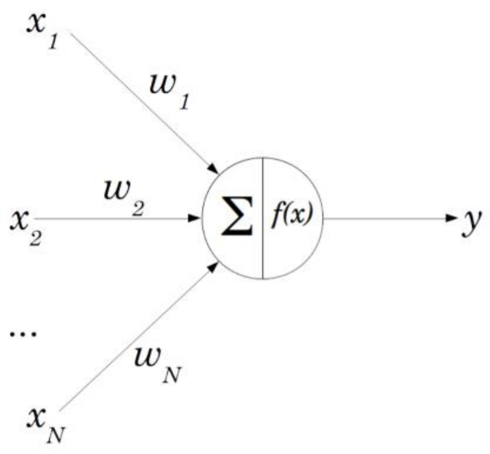


Figure 2-14: architecture of the perceptron [18]

In this structure, the inputs to the algorithm are not binary anymore and each one of them has a weight associated. The network will now focus on two tasks:

- Perform a linear combination with the products of the inputs, their corresponding weights and an independent term, which will be known as bias.

$$x_1 \cdot w_1 \; + \; x_2 \cdot w_2 + \cdots + \; x_N \cdot w_N + b \qquad (2.12)$$

- Apply a step function (which will later be known as the activation function) to the result, which in most cases is the following function:

$$sgn(z) = \begin{cases} -1 & if \quad z < 0 \\ 0 & if \quad z = 0 \\ +1 & if \quad z > 0 \end{cases} \qquad (2.13)$$

Once the operations have been performed, the result will be the prediction ŷ. Just as before, the procedure will be to calculate the error *y*- *ŷ*; if the error is equal to zero, the prediction matches the real value perfectly. If the error differs from zero, the gradient descent will be used to update the parameters used and thus reduce the error.

As long as there is an invariant part in the prediction, the bias *b* must be used. This, whose value will also be optimized, is important because it allows to shift the activation function by adding a constant value. To further explain the concept, the bias has a similar behavior to the independent term in linear equations.

The main objective of the perceptron is to classify the patterns that are linearly separable, aiming to obtain a hyperplane that separates the two zones perfectly. In the case of having a problem where the data is not linearly divisible by a hyperplane, this type of network will be of no use and a more complex algorithm will be required.

2.2.2 BASIC NOTIONS

After this brief introduction, it is now time to analyze Neural Networks in greater depth. Before diving into the new concepts, it is necessary to explain that there are several changes that need to be understood. Some of them are just changes in notation, whereas others will have a major impact.

First of all, there will be available m data and n features, this last one represents the characteristics that will describe the data. For this reason, the input *X* will no longer be a vector, and it will now be a matrix of range *(n,m)*. This affects directly the weights, as there has to be a weight for each of the neurons in the NN. Furthermore, in the gradient descent explanation, the weights will no longer be determined with the θ_t and the bias with the θ_0; from now on, the weights will be represented with the letter *w*, and the bias with the *b*. With all this, the function that will be used to calculate the gradient descent will not be one, but two, one to modify the weights and another one for the bias. As a result of this change, it will be necessary to use partial derivatives, whose nomenclature will be modified as follows to facilitate the writing of the equations:

$$dw = \frac{dJ(w,b)}{dw} \tag{2.14}$$

$$db = \frac{dJ(w,b)}{db} \tag{2.15}$$

All in all, the expression of the gradient descent will be:

$$w := w - \alpha \cdot dw \tag{2.16}$$

$$b := b - \alpha \cdot db \tag{2.17}$$

Once the concept of perceptron and the new changes in the nomenclature have been explained, we can analyze the structure that neural networks normally have.

As mentioned earlier, the NN are comprised of several neurons that are clustered into units called layers and, when the number of layers is greater than one, they are known as Multilayer Neural Networks.

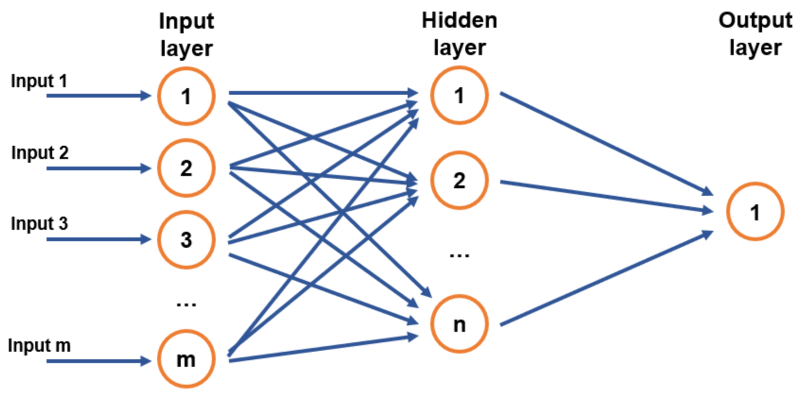


Figure 2-15: example of a multilayer NN

As shown in l Networks is made of three types of layers:

- **Input layer:** the one in which the matrix X (with the original data) is introduced. For this reason, the number of neurons in this layer will depend on the range of input data values.
- **Hidden Layer:** can be composed of numerous layers that are responsible for performing the necessary operations to obtain the most appropriate output. Each of these layers will have a number of neurons that may vary according to the complexity of the problem.
- **Output layer:** is responsible for providing the result achieved. Depending on the type of problem, it will be composed of different number of neurons. As an example, in the case of a binary classification problem, there will be a single output neuron that will tell whether the final answer is a 1 or a 0.

In addition to the above, it can be seen that between to layers ALL the units share their information. This is due to the fact that the information is always shared with a particular weight associated to it and, when it arrives to the next layer, all the data received is put together before starting the processing of it.

To facilitate the comprehension of these last concepts and make easier the understanding of the following ones, superscripts $^{[l]}$ will be used to distinguish the different layers; $^{[0]}$will refer to the input layer (X). On the other hand, subscript $_p$ will be used to distinguish the different neurons inside a particular layer. Therefore, the first neuron of the first hidden layer will perform the following operation:

$$z_1^{[1]} = w_1^{[1]} \cdot X + b_1^{[1]} \tag{2.18}$$

Notice how the row vector w_1 is multiplied by the matrix X, to which is added the vector $b_1^{[1]}$, which is a row vector whose components are all equal to b_1.

After obtaining the value of $z_1^{[1]}$, this one is used as input in an activation function. This can be of different types depending on the problem at hand and how one wants to deal with the information provided. A commonly used activation function is the sigmoid function, mentioned above in chapter 2.1.2, whose output values vary in the interval [0,1]. Another widely used option is the hyperbolic tangent, which differs from the sigmoid in that the output values are in the interval [-1,1], so that the mean of our data would be centered at 0.

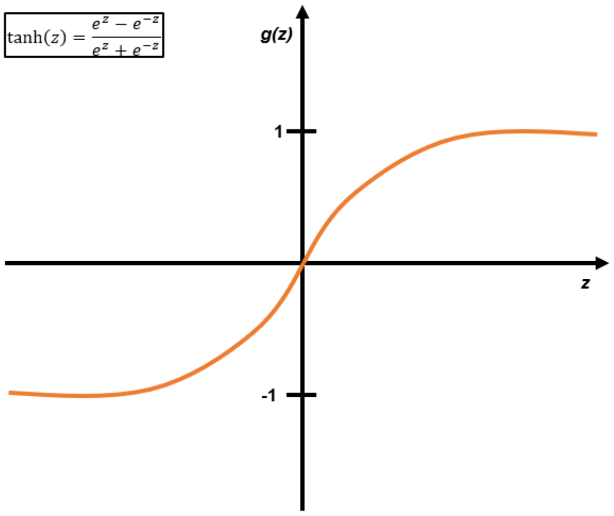


Figure 2-16: hyperbolic tangent

However, although the hyperbolic tangent function is widely used, for this project the activation function that will be used is the Rectified Linear Unit function (ReLU from now on). Its expression is $g(z)$ = max (0, z) and is also very popular in classification algorithms.

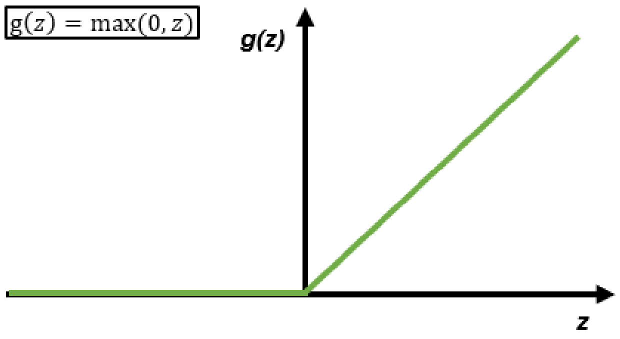


Figure 2-17: ReLU function

Another very common activation function is the Leaky ReLU function. This one is very similar to the ReLU activation function with the difference that for negative values of *z*, *g(z)* is a negative value instead of zero. Regarding the function it is interesting to point out that the value $0.01 \cdot z$ is sometimes changed into another parameter of the learning algortithm.

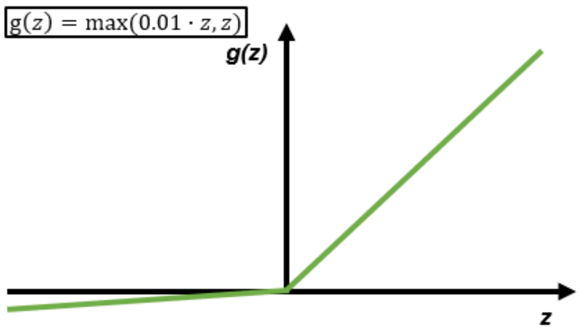


Figure 2-18: leaky ReLU function

In this case, the aim is to activate the data with a nonlinear function that facilitates the optimization of the network. The reason for this is that it helps the model generalize and to adapt to different kinds of data and find the difference between numerous outputs. As a consequence, when having a binary classification problem, all the neurons will be activated with a ReLU function (the reason for choosing the ReLU function instead of the tanh function or the Leaky ReLU will be explained in ANNEX II – Tanh vs. ReLU vs. Leaky ReLU functions). The only exception to this will be the output layer which, in our case will have just one neuron which will be activated with a sigmoid function.

Another activation that will be used very frequently, especially when working with Recurrent Neural Network (later explained), is the SoftMax (or SoftArgMax) activation function. This function normally replaces the sigmoid as the output function in RNNs (Recurrent Neural Networks). The way it works is that it receives a vector of K real values and generates another vector of K real values that sum to 1. All the inputs are transformed into numbers between 0 and 1 so that they can be seen as probabilities. If one number is negative, it will be transformed into a positive value very close to zero. It is basically a generalization of the logistic regression.

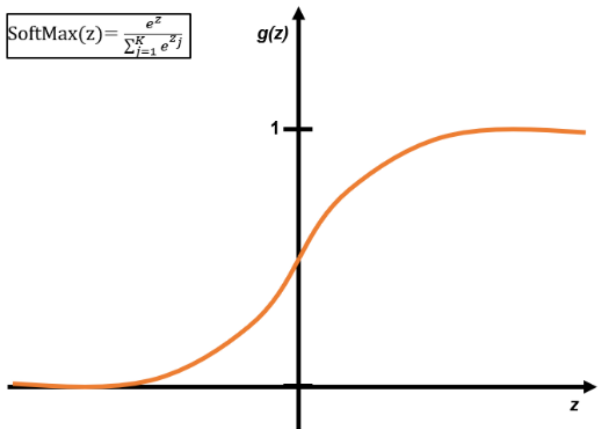


Figure 2-19: SoftMax function

The main reason for using a sigmoid activation function in the last layer is because it can only output values between 0 and 1, this is very helpful if the purpose of the algorithm is to determine a probability. As the expected output of the algorithm is the prediction of growth, then the use of a sigmoid activation function is definitely advisable.

Finally, the result of the activation of the neurons in each layer will be represented by the vector a, which will be the output of each of these. If all the above concepts are brought together and all the neurons in each layer are represented by a vector, the operations that would be carried out for a one-hidden-layer network would be the following:

$$a^{[0]} = X \tag{2.19}$$

$$z^{[1]} = w^{[1]} \cdot a^{[0]} + b^{[1]} \tag{2.20}$$

$$a^{[1]} = g(z^{[1]}) \tag{2.21}$$

$$z^{[2]} = w^{[2]} \cdot a^{[1]} + b^{[2]} \tag{2.22}$$

$$a^{[2]} = g\left(z^{[2]}\right) = \hat{y} \tag{2.23}$$

Notice that none of the previous elements have subindices, this is because a matrix notation is used. For example, $w^{[2]}$ is the matrix containing all the row vectors ${w^{[2]}}_2$, this happens for all the elements.

The process of obtaining the activation $a^{[l]}$ for each of the layers is known as the *forward pass* or *forward propagation step*.

Normally all the hidden units will be activated with a ReLU, Leakey ReLU or hyperbolic tangent whereas the output layer will be activated with a SoftMax or a Sigmoid function. The reason for this criterion is that it is the best way to avoid what is known as *vanishing gradients*. The problem with *vanishing gradients* is that if the activation functions constantly reduce the value of $a^{[l]}$, there is a point where the values are so small that the loss function is very close to zero and the training becomes harder and much slower (more computationally expensive). As both SoftMax or Sigmoid narrow down the inputs to values between 0 and 1, if they were continuously implemented the consequence would be the *vanishing gradients*. It is for this reason that the other activation functions are used in the hidden layers, because they do not generate a small derivative. Other alternatives to solve the problem at hand would be the use of *residual networks* or *batch normalization*, however, they will not be considered for this project.

Another problem is the one related with *exploding gradients*, this time the opposite happens, now the gradients are too big. However, the solution this time is easier and more robust; if the values surpass a preestablished threshold, the value will be rescaled.

It is important to keep in mind that Neural Networks are a supervised learning technique, which means that the data used includes the output that one should expect from the network. Therefore, the output $\hat{y}$ will be compared to the answer expected and an error will be obtained using the cost function described in the previous sections. This error will propagate backwards throughout the network, modifying the weights and optimizing the weights and bias for each layer, this is a technique known as *backpropagation* (or *backward pass*) algorithm, will be discussed in more detail in the next section.

Each time all the *forward propagation steps* are done and the *backpropagation steps* are performed, an *epoch* is built. Normally, enough epochs will be attempted until one obtains network parameters that allow the output values to be very close to those desired to predict.

2.2.3 BACKPROPAGATION STEP

The backpropagation computation is one of the most commonly used mechanisms to optimize the learning of NN. The functioning of this technique consists of correcting the different weights and parameters of the network by means of the error calculated between the output of the network and the real one. A very high number of iterations are carried out to train the network so that, in the end, it learns to predict new values with very high accuracy percentages.

In order to explain how Backpropagation works, it will be explained step by step how it would be carried out in a Neural Network with a single hidden layer, just like in the previous chapter (Equations (2.19), (2.20), (2.21), (2.22) and (2.23))

First, a cost function has to be chosen, which in this case will be (2.11). The main objective will be to minimize it in the following way:

$$\min_{(w,b)} J(w,b) \tag{2.24}$$

In order to minimize this equation, the gradient descent will be used **¡Error! No se encuentra el origen de la referencia.**(2.17). For this step to be performed, it is required to have the partial derivatives $dw = \frac{dJ(w,b)}{dw}$ and $db = \frac{dJ(w,b)}{db}$ for each and every one of the existing layers (which will be referred to as $dw^{[l]}$ *and* $db^{[l]}$ respectively). To obtain $dw^{[2]}$ and $db^{[2]}$, one must realize that the partial derivative cannot be calculated directly, it is necessary to use the chain rule, since the cost function J depends on the activation function *σ(z)*, which depends on *z*, which, at the same time, is a function of w^2 and b^2.Therefore, dw^2 will be as follows:

$$dw^2 = \frac{\partial J}{\partial w^{[2]}} = \frac{\partial J}{\partial a^{[2]}} \cdot \frac{\partial a^{[2]}}{\partial z^{[2]}} \cdot \frac{\partial z^{[2]}}{\partial w^{[2]}} \tag{2.25}$$

In order to solve this, it is necessary to go step by step, therefore the focus will be placed on each of the elements individually. The first element will be:

$$\begin{aligned}\frac{\partial J}{\partial a^{[2]}} &= \\ &= \frac{\partial(\frac{-1}{m} \cdot [\sum_{i=1}^{m}(y^{(i)} \cdot \log(a^{[2](i)}) + (1-y^{(i)}) \cdot \log(1-a^{[2](i)})])}{\partial a^{[2]}} = \\ &= \frac{-y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}}\end{aligned} \tag{2.26}$$

Before starting the second element, it is important to take into account that the function is continuously differentiable, this means that $\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$

$$\frac{\partial a^{[2]}}{\partial z^{[2]}} = \frac{\partial \sigma(z^{[2]})}{\partial z^{[2]}} = \sigma'\left(z^{[2]}\right) = \sigma(z^{[2]}) \cdot (1 - \sigma(z^{[2]})) \tag{2.27}$$

Finally, the last derivative can be solved:

$$\frac{\partial z^{[2]}}{\partial w^{[2]}} = \frac{\partial(w^{[2]} \cdot a^{[1]} + b^{[2]})}{\partial w^{[2]}} = a^{[1]} \tag{2.28}$$

Putting all together, the equation simplifies to:

$$dw^{[2]} = (\frac{-y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}}) \cdot \sigma'(z^{[2]}) \cdot a^{[1]} \tag{2.29}$$

Similarly, one can obtain $db^{[2]}$, with the only difference that $\frac{\partial(w^{[2]} \cdot a^{[1]} + b^{[2]})}{\partial b^{[2]}} = 1$, therefore the final equation will be:

$$db^{[2]} = (\frac{-y}{a^{[2]}} + \frac{1-y}{1-a^{[2]}}) \cdot \sigma'(z^{[2]}) \cdot 1 \tag{2.30}$$

All these equations just obtained will be the same for the different layers in the NN. However, there can be a difference depending on the activation function (ReLU or sigmoid), for this reason the activation function will always referred to as *g(z)*. Therefore, generalizing for any layer of the network, the equations to obtain dw and db are the following:

$$dw^{[l]} = (\frac{-y}{a^{[l]}} + \frac{1-y}{1-a^{[l]}}) \cdot g'(z^{[l]}) \cdot a^{[l-1]} \tag{2.31}$$

$$db^{[l]} = (\frac{-y}{a^{[l]}} + \frac{1-y}{1-a^{[l]}}) \cdot g'(z^{[l]}) \cdot 1 \tag{2.32}$$

Finally, these equations will be used for the gradient descent for the updating of all the weights and bias in the different layers of the network.

$$w^{[l]} := w^{[l]} - \alpha \cdot dw^{[l]} \tag{2.33}$$

$$b^{[l]} := b^{[l]} - \alpha \cdot db^{[l]} \tag{2.34}$$

To summarize, the steps that make up the learning process of a neural network are the following:

1. The weights and biases of the network are randomly initialized.
2. An input vector is taken from the data available and is propagated forward through all the layers until it reaches the output layer.
3. The error obtained is calculated by means of the cost function, comparing the output with the desired target.
4. This information is propagated backwards in the network, obtaining the results of the different partial derivatives.
5. Steps 2, 3 and 4 are repeated for each vector of input data (m training examples).

6. The weights and biases are modified according to the results obtained, thus completing an epoch.
7. Steps 2, 3, 4, 5 and 6 are repeated until the total error of our network is minimized.

2.2.4 DATA STRUCTURE

One key element when it comes to the training of the Neural Networks, is to find the proper way to organize the dataset available. In order to do so, it will always be necessary to divide the dataset into three different subgroups: *Train, Cross Validation and Test sets*. The data will have to be distributed homogeneously (if this was not possible, it is necessary to make sure that both, *cross validation* (also known as *Development or dev set*)) and *test* come from the same distribution. Coming from the same distribution means that if, for example there are for different classes (1, 2, 3, 4) in a classification problem, to train the network it will be necessary to have in each of the subgroups just mentioned a similar quantity of data from each class. If by any chance there were 10.000 examples from class 1 and 2.000 from class 2 it will be necessary to find a way to match both numbers because such difference will negatively impact the network as the learning will be much less effective.

Train
Validation
Test

Figure 2-20: data division

In what follows, the specific function of each subgroup and the reason why they are so important will be discussed.

The Train Set is the first and largest of these, typically taking about 70% of the data. This set is used to train the network, modifying the weights and biases until the optimal parameters for its operation are reached.

The Validation Set has a smaller size, approximately 15% of the total data. Its main function is to be able to give a first idea of how the network works with data that it has not "seen" previously. Therefore, what it does is to make a prediction on a set of data with the parameters obtained in the Train Set. In addition, it also serves to modify the hyperparameters of the problem, such as the learning rate or the number of hidden layers to be used.

The Test Set is the last group, which usually contains the remaining 15% of the data. After tuning the network parameters and hyperparameters thanks to the previous subgroups, this is used to make the final check that the network is working appropriately. To do this, it makes several predictions on completely new data and observes the results. Normally, if the steps have been followed well and no errors have been made, a high percentage of success in these predictions should be obtained to confirm that the network is working properly and can be used.

Although it has been said before that the distribution of the data should be 70-15-15; this is indicative and depends on the amount of data available. For small datasets (100, 1.000 or even 10.000 examples) these values are more than adequate. However, if the amount of data is much greater (1.000.000 examples) the most appropriate distribution that 98% would be used in the Training set and the rest, 1 percent to the Validation and the other 1 percent to the Test Set.

Another reason for dividing the data in subgroups is because it makes it easier to detect problems such as overfitting, which was explained earlier in chapter 2.1.3. To do this, the bias vs. variance trade-off will be used, which will now be explained in further detail.

The first concept is the error due to bias, this is simply the difference between the expected value of the estimator and the true value. So, if the model has a high bias, it means that it is too simple and the training data has not been well adjusted, also known as underfitting.

The variance, however, indicates how much the prediction varies according to the data used for training. If there was a high variance and the Validation data was different from the Train set, the output would be very different and with higher error, also known as overfitting.

To get a robust model, one has to use the bias vs variance trade off, which consists of finding an equilibrium point where one has a low bias and variance, trying different combinations until the optimal point is found.

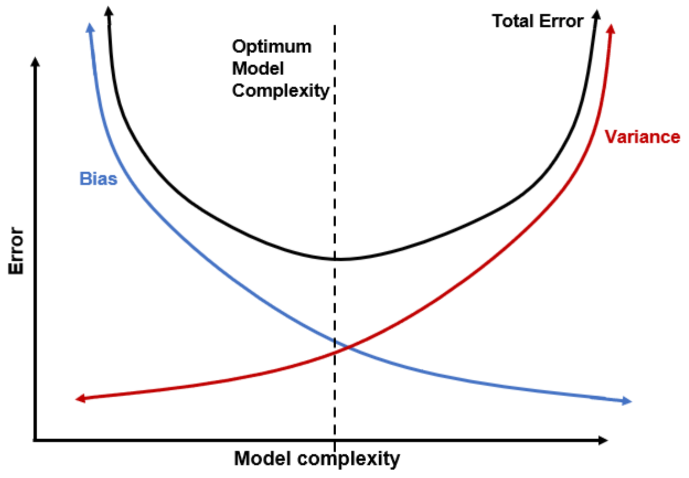


Figure 2-21:bias-variance tradeoff

2.2.4.1 Procedures for fitting problems

One of the most used procedures to reduce the variance (overfitting) of an algorithm is the *regularization technique*, as it enables to reduce the variance without having a substantial increase in its bias. The regularization is basically the introduction of another adder to the cost function. This new summand will be different depending on the type of regularization used, however, there will always be the hyperparameter λ which is known as the regularization hyperparameter. Usually, λ will be set using the development set, but it can also be obtained through a trial-and-error procedure (see what is the value that gets the best performance). The regularization can be L^1 or L^2 (also known as "Weight decay") and the cost functions associated are:

$$L^1 \qquad J(w,b) = \frac{1}{m}\cdot\sum_{i=1}^{m}\mathcal{L}\left(\hat{y}^{(i)},y^{(i)}\right) + \frac{\lambda}{m}\cdot\|w\|_1 + \frac{\lambda}{m}\cdot b^2 \qquad (2.35)$$

Where: $\|w\|_1 = \sum_{i=1}^{n_x}|w_1|$

$$L^2 \qquad J(w,b) = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right) + \frac{\lambda}{2 \cdot m} \cdot \|w\|_2^2 + \frac{\lambda}{2 \cdot m} \cdot b^2 \qquad (2.36)$$

Where: $\|w\|_2^2 = \sum_{j=1}^{n_x} {w_j}^2 = w_j^T \cdot w_j$

The last element of both equations does not make a significant difference. For this reason, it is often omitted as it is computationally expensive, and the outcome does not make a difference.

Another important element to be considered is that when using the L^1 regularization, the weights (w) will end up being sparse, this means that the matrix of weights will end up having a lot of zeros in it.

It is important to consider that if λ is very big, there is a high chance of setting the values of many weight matrices close to zero. This means that the influence of these hidden units is being cancelled. Therefore, the Neural Network gets simplified, and this will result in decreasing the overfitting (but it will also increase the variance…).

Another intuition that helps explaining why the algorithm is getting simplified is the following; if one uses the *tanh* as the activation function, it is clear to see that if *z* is small then *g(z)* will be found in the linear regime of the function. If the value of λ is high, then the weights are penalized, and they become smaller. If the weights are smaller, then *z* will be smaller too (look at equation (2.8)) and if *z* is very small then every layer will be simplified thus transforming the algorithm into a roughly linear one (just like any regression).

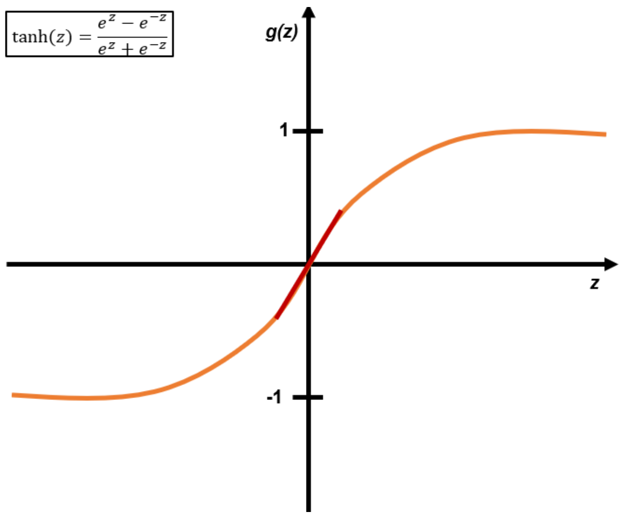


Figure 2-22:tanh function

Another technique to reduce overfitting problems is the use of *dropout regularization*. It will consist of going through each of the layers of the network and set some probability of eliminating a node (or unit) in the NN. This technique is usually implemented using "inverted dropouts"; this means that the variable used, *keep_prob*, will determine what is the probability that a given hidden unit will be maintained (opposite/inverse of what will be dropped). It is a value that can be kept fixed for every layer of the algorithm or that can be different for each one. It makes the testing easier as it reduces the scaling problem. However, it is important to understand that the elements that are being zeroed out are chosen randomly, therefore as the algorithm usually iterates, on other iterations other hidden units will be dropped. This method is very effective, but one must consider that is computationally expensive.

The reason why dropout is so useful is because it forces each unit not to rely on only one feature (as that one can be zeroed out), this pressures the unit to spread the weights. If the weights were too concentrated and the unit associated disappeared, then the algorithm would fail.

It is in the layers with the biggest matrices is where there is a high risk of overfitting, for these layers it is recommended to eliminate more units and by doing so, reduce the overfitting. If on the other hand, there were layers where one was sure that there was no overfitting, one could not use the dropout for that layer.

One big downside of the *dropout technique* is that the cost function is no longer well defined so it is no longer possible to use plotting techniques that otherwise could be used.

Other alternatives to eliminate the overfitting have to do with the *augmentation of data*. It is often the case that this is not an alternative, however, what could be done is to slightly modify the data already available so that it looks like new data. For example, if one is working on a cat-recognition model, one could flip the images, rotate them, zoom them in or add random crops.

Another way is the *Early stopping*, this approach is based on the premise that there is a specific number of iterations until which the performance is very good and from this number on, the performance gets worse. The idea is to stop the algorithm in this number of iterations.

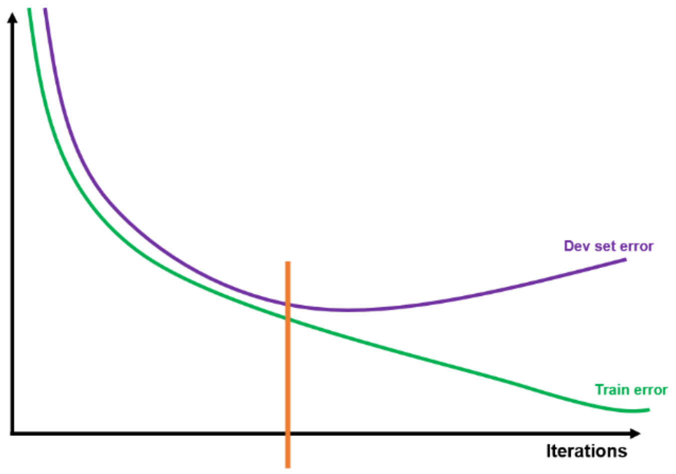


Figure 2-23: Early stopping approach

However, there is a downside to this method. When one is solving an AI problem, usually the focus is first placed on the reduction of the cost function and once this is solved, one can continue and work on the next problem, which is overfitting. This methodology of focusing on one problem at a time is called *orthogonalization*. The problem of *early stopping* is that there is no possibility of orthogonalizing because by stopping early the optimization of the cost function is broken.

The best method is the L^2 regularization, but there is going to be a need of trying many values for λ and this is computationally expensive, so it must be considered when implementing it and when deciding the amount of time needed.

2.3 *SEQUENCE MODELS AND NATURAL LANGUAGE PROCESSING*

Out of all the different applications of Deep Learning, and more precisely the Neural Networks, the ones that have experimented an exponential growth are both the Convolutional Neural Networks (CNNs from now on) and the Sequence Models. In this project there is no particular interest in the CNN, however, the use of Sequence Models (and more precisely the RNNs) can be of great use to optimize the forecasting abilities of the algorithm.

Sequence modelling is the area of AI that focuses on forecasting the elements of a sequence given a particular input. The type of data that is used in this kind of models includes clips of audio or video, time-series data, lines of text, etc. There are many different applications for the Sequence Models, some of the applications are:

- Video activity recognition: The algorithm receives a sequence of images of an individual performing an activity, then the algorithm will analyze them and output the activity that is being realized.
- Speech recognition: The algorithm will have as input a recording of a person speaking and the software will then output the sentences inputted.
- Machine translation: The machine will receive a sentence in one language and will generate the same sentence but in another language.

As it can be seen, many of the applications are strongly related with the use of words and sentences, the analysis of this kind of input is known as NLP, which stands for Natural Language Processing.

One of the main characteristics of the Sequence Models that also explains why they are so popular is that the sizes of the input and output matrices do not necessarily need to match. The following example will help to clarify this concept; if one was working on a machine translation algorithm, it is possible that the sentence inputted has a particular length and when translated the number of words is different.

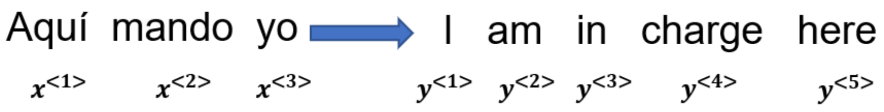

Aquí mando yo
I am in charge here
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$
$y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$

Figure 2-24: size mismatch

As everyone can see, the size of *Y* is different from the size of *X*. In normal algorithms this would result in the failure of the algorithm. This flexibility together with many other characteristics of these models enables reaching outstanding results that help solve many day-to-day problems that would otherwise remain unsolvable.

As explained before, this project will focus on Sequence Model programming. In this chapter the basic notions of NLP and the algorithms implemented to process this kind of data will be explained in detail.

2.3.1 HISTORY AND ORIGIN OF NATURAL LANGUAGE PROCESSING

The birth of NLP [11] dates back to the Second World War (1940s) where there was a need to translate crucial information from English to Russian and vice-versa; initially there were only two languages until 1960 when Chinese was also introduced. Between 1964 and 1966 the Artificial Intelligence Laboratory of MIT created ELIZA, a chatbot program created by Joseph Weizenbaum that was created to answer the questions of the users, it was one of the few algorithms that were capable of taking the Turin test.

The main problem in this area is that, although there were many initiatives to solve various problems, there were no platforms that enabled the development of them. For this reason, the first big advancements took place in 2013 when Tomáš Mikolov introduced the Word2Vec, this was two-layered NN model that transformed words so that deep NN could then analyze them.

Thanks to the advancements in Deep Learning, Recurrent Neural Networks and Long Short-term Memory networks gained popularity in 2014 and this led to the creation of the Transformer model in 2017 that was famous for the use of parallelization. This last model has been the engine of progress ever since and has enabled the creation of models like BERT, ERNIE 2.0 or XLNet that are known for using bidirectionality, a state-of-the-art technique that has proved to increase the efficiency of the algorithms.

2.3.2 RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNNs) are a type a type of network that uses sequential or time series data to predict the next probable scenario. They are used in Deep Learning and are incorporated into well-known applications such as Siri or Google Translate.

The differential factor from other models is that they have “memory” in the sense that it takes into account the conclusions reached with previous data to influence the current input and output. This so called “memory” is known as feedback loops and is what makes sure that the information persists when working with new inputs.

One may wonder why experts consider so important for an algorithm to have “memory”. The reason for this is that, especially in NLP, if the information is not passed on, the output may not be coherent and therefore useless. For example, if the sentence that the algorithm has to work on is “cats are beautiful” it is necessary that the algorithm takes into account that “cats” is plural. If this is ignored then the sentence outputted will most probably have a syntactical error.

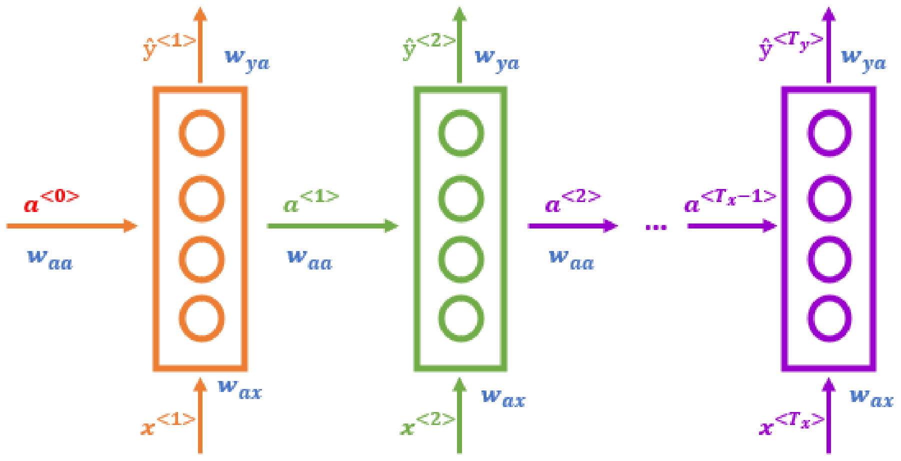


Figure 2-25: Recurrent Neural Network diagram

As one can see in Figure 2-25, when the RNN goes on to read the second element (almost always will be a word, but can also be a letter) of *X*, instead of just predicting $\hat{y}^{<2>}$ using only $x^{<2>}$, it will also consider the information from the computation of $x^{<1>}$; in other words, the information is passed on.

The parameters governing the connection from $x^{<1>}$ to the hidden layer will be written as w_{ax}, it is important to understand that the same matrix of weights will be used for every time step. On the other hand, the activations (horizontal connections) will be governed by some set of parameters w_{aa} which, as well as before, will be the same one for every time step. The same thing will happen for the output prediction for w_{ya}. Another important thing to take into account is that $a^{<0>}$ normally is a vector of zeroes.

Although this model is very useful, one of its main weaknesses is that the algorithm will take into account the previous information but not the one ahead.

Returning to Figure 2-25, the computation of the forward propagation is the following:

$$a^{<0>} = 0 \tag{2.37}$$

$$a^{<t>} = g(w_{aa} \cdot a^{<t-1>} + w_{ax} \cdot x^{<t>} + b_a) \tag{2.38}$$

$$\hat{y}^{<t>} = g(w_{ya} \cdot a^{<t>} + b_y) \tag{2.39}$$

For $a^{<t>}$, the activation functions can be either the hyperbolic tangent (tanh) or the Rectified Linear Unit (ReLU). For $\hat{y}^{<t>}$, the activation function can be the sigmoid or the softmax function. In addition, these equations are often simplified using the matrix notation:

$$a^{<t>} = g(w_a \cdot [a^{<t-1>}, x^{<t>}] + b_a) \tag{2.40}$$

$$\hat{y}^{<t>} = g(w_y \cdot a^{<t>} + b_y) \tag{2.41}$$

For further clarification, w_a is just the element that englobes all the weights. If, for example, a had 100 elements, the dimension of w_{aa} would be *(100, 100)*, if x then had 10.000 elements, the dimension of w_{ax} would be *(100, 10.000)*, therefore, the size of w_a would be *(100, 10.100)*.

On the other hand:

$$[a^{<t-1>}, x^{<t>}] = \begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix} \tag{2.42}$$

Therefore:

$$w_a \cdot [a^{<t-1>}, x^{<t>}] = w_{aa} \cdot a^{<t-1>} + w_{ax} \cdot x^{<t>} \tag{2.43}$$

With regard to the backpropagation step, it is basically the same as the one seen in the previous chapters:

$$L^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \cdot \log(\hat{y}^{<t>}) - (1 - y^{<t>}) \cdot \log(1 - \hat{y}^{<t>}) \tag{2.44}$$

$$L(\hat{y}, y) = \sum_{t=1}^{T_y} (\hat{y}^{<t>}, y^{<t>}) \tag{2.45}$$

2.3.2.1 Types of Recurrent Neural Networks

Before it has been stated that the inputs and outputs of the RNNs do not necessarily need to match lengths, this one can vary and, depending on the tradeoff between them, one will be dealing with one kind of RNN or another. The possible alternatives are the following:

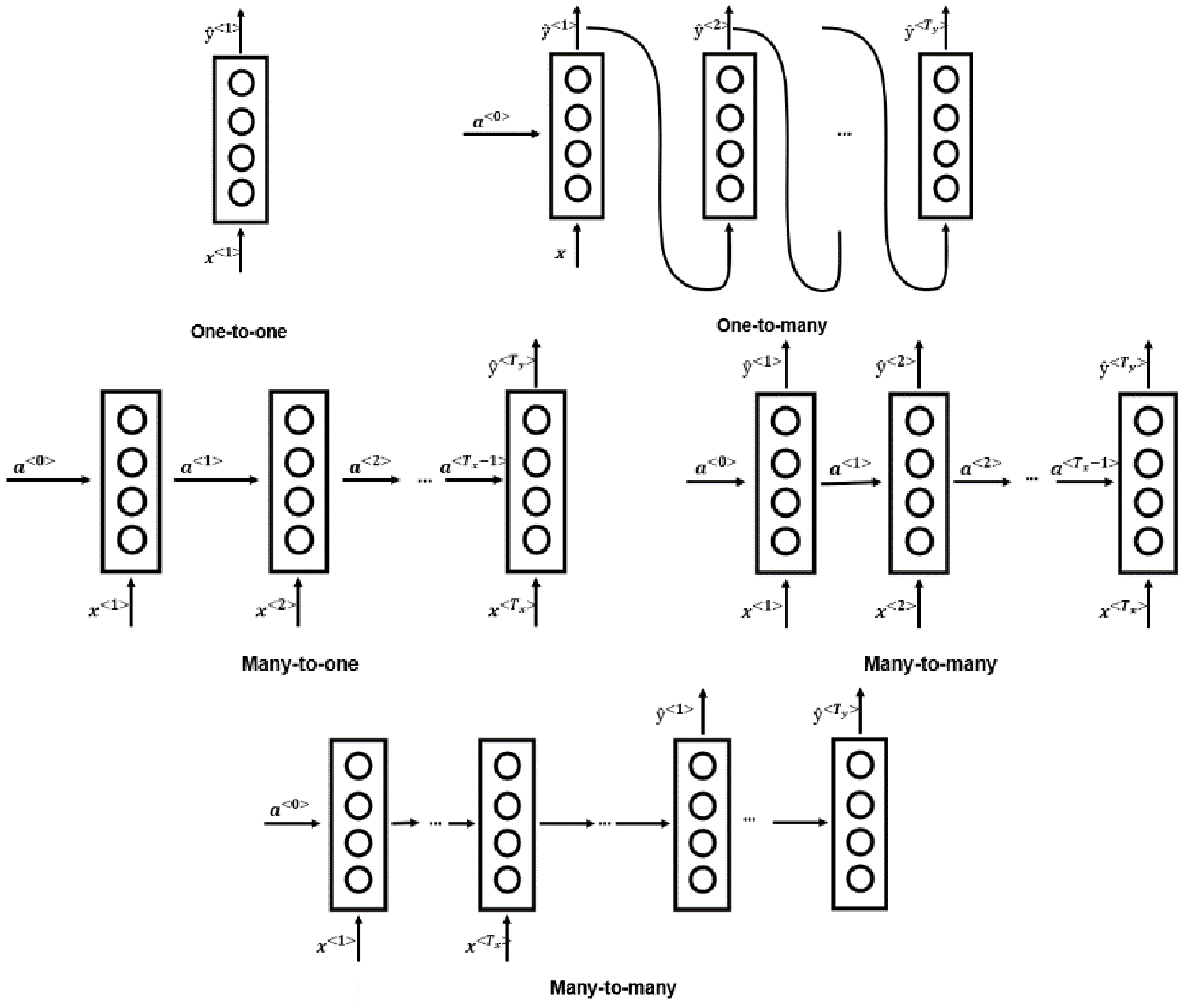


Figure 2-26: RNN types

2.3.2.2 Understanding Recurrent Neural Networks

If one had to define RNNs in very few words, it could be said that it is an algorithm which outputs the chances of getting a sentence correctly.

The training set will be large corpus of text and the first thing that would be done is to *tokenize* the sentences. To *tokenize* means creating a vocabulary and map all the words in a sentence to a one-hot vector. Another highly recommended thing would be to tokenize the end of the sentences by adding <EOS> (End Of Sentence) at the end of each sentence on the training set.

Once the vocabulary is created, it is very frequent to discover that there are words that have not been considered that appear in the dev or test set. This setback cannot be ignored as it results in the failure of the algorithm, what will be done is to replace the word with <UNK> (Unknown) which is an auxiliary token for this particular cases.

For a better understanding of the notion, an example will be used.

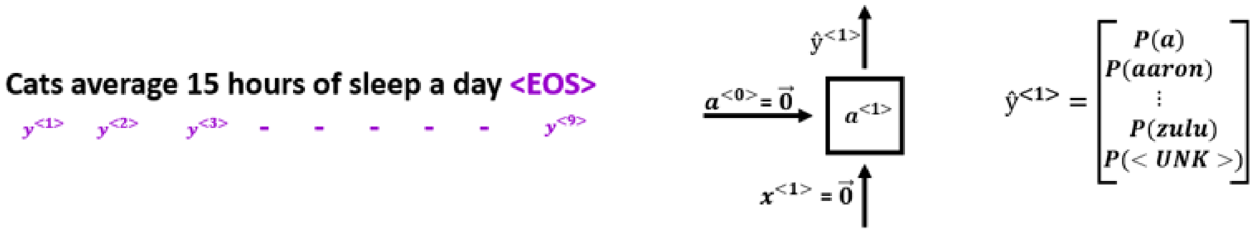

Cats average 15 hours of sleep a day <EOS>
y<1> y<2> y<3> - - - - - y<9>
ŷ<1>
a<0> = 0⃗
a<1>
x<1> = 0⃗
ŷ<1> =
P(a)
P(aaron)
⋮
P(zulu)
P(< UNK >)

Figure 2-27: RNN example

As one can see, the sentence is not the input but the output, this means that the algorithm at time zero will not receive any information and will have to compute some activation $a^{<1>}$ that will be a SoftMax prediction of the probability of the first word being a word from the dictionary previously created. After getting the probability of being each of the words in the dictionary, then the command *np.random.choice* will be used to sample according to the distribution designed by the vector of probabilities and that enables to sample the first words.

Once the algorithm has kicked off, the output of the previous block will be imputed in the following providing valuable information that reduces the complexity of the forecasting. When working with large sets of data, the algorithm will be able to, given any initial set of words, predict what is the chance of the next word very efficiently.

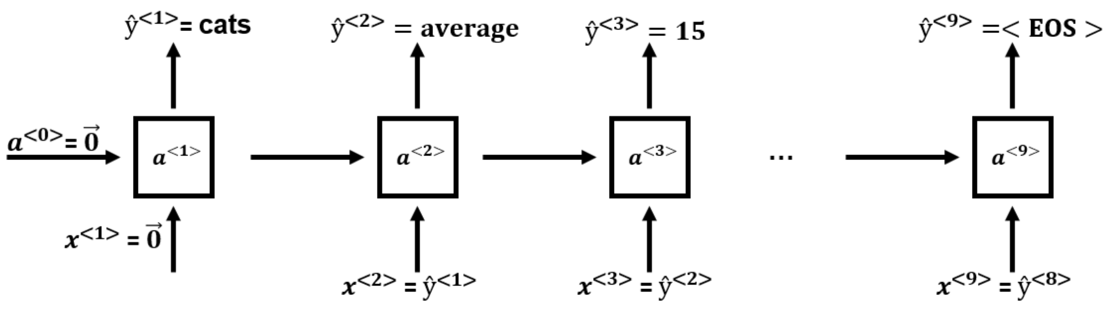


Figure 2-28: RNN example (2)

If <EOS> is a part of the vocabulary, one can keep sampling until the <EOS> token is generated. If this element was not present in the vocabulary, then what could be done is to establish a limit of words for the model to sample (20 or maybe 100 words) and continue until such number of steps is reached.

Depending on the application, one thing that could be done is to build a character level RNN. In this case the vocabulary would just be the alphabet as well as space punctuation, digits and maybe a distinction from upper and lower-case letters. As a consequence, the outputs will be individual characters. This alternative allows one to forget about unknown tokens and is also very useful as is able to assign any sequence a different-from-zero probability. On the other hand, there is a major setback that must not go unnoticed; the sequences will significantly increase its length and this will be computationally expensive.

Finally, just like with any other Neural Network, it is possible to create Deep Recurrent Neural Networks, however, these kinds of models do not usually go as deep as other types of algorithms.

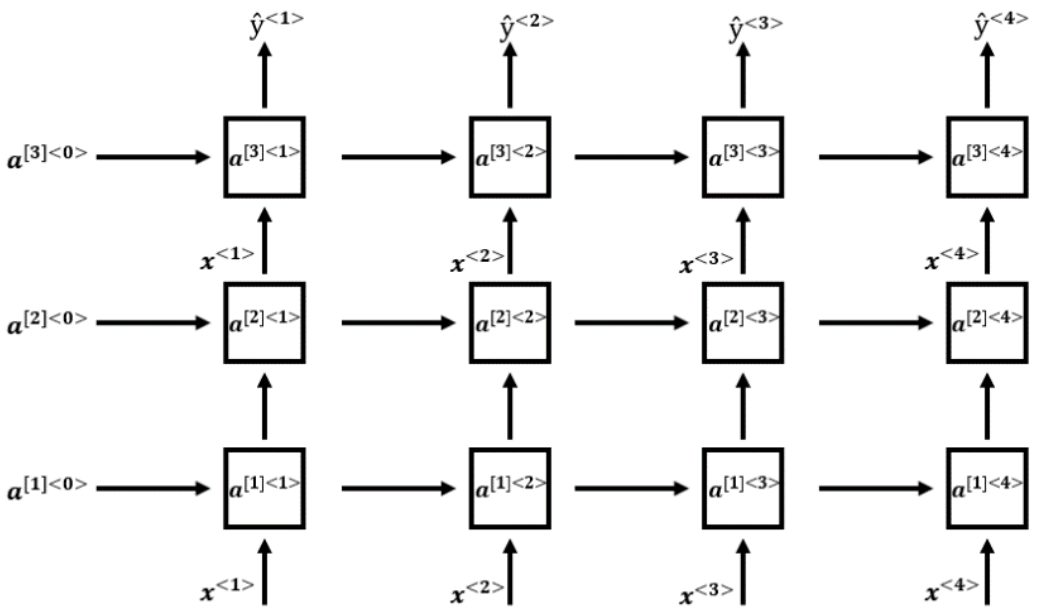


Figure 2-29: deep Recurrent Neural Network

2.3.2.3 Vanishing Gradients with RNNs

One of the main problems of the RNNs are the *vanishing gradients*. This complication (explained in chapter 2.2.2) has a major impact in the RNNs because it not only increases the computational cost of the optimization of weights, but also hinders the transfer of information between units. In other words, the "memory" of the algorithm is worsened.

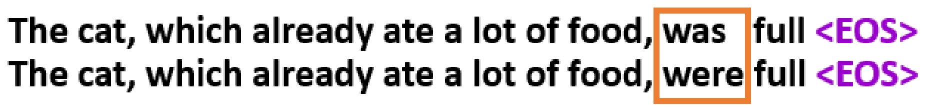

The cat, which already ate a lot of food, was full <EOS>
The cat, which already ate a lot of food, were full <EOS>

Figure 2-30: syntactical errors due to vanishing gradients

The main problem is that $x^{<3>}$ is mainly influenced by the close elements, while the ones further in the sentence have little influence on them, so if $x^{<2>}$ is plural and this determines the $<T_x - 1>$ position, the need arises for a way to get the information further in the sentence.

2.3.2.4 Gated Recurrent Unit (GRU)

It is one of the most used solutions for *vanishing gradients*. This time a new variable will be used, this element is called "memory cell" and it will be represented with the letter *c*. The memory cell will lock into some information that will be remembered and, on each step, it will be necessary to determine if the information previously saved is still necessary or it can be overwritten or updated. The notation that will be used is the following:

- $c^{<t>}$: it represents the information initially saved.
- $\tilde{c}^{<t>}$: it represents the candidate information for replacing $c^{<t>}$
- Γ_u: it is the update gate. Depending on the value (ranging from 0 to 1), $c^{<t>}$ could be overwritten.

All of them are vectors and they all share the same dimensions. They can be computed with the following operations:

$$c^{<t>} = a^{<t>} \tag{2.46}$$

$$\tilde{c}^{<t>} = tanh(w_c \cdot [c^{<t-1>}, x^{<t>}] + b_c) \tag{2.47}$$

$$\Gamma_u = \sigma(w_u \cdot [c^{<t-1>}, x^{<t>}] + b_u) \tag{2.48}$$

It is interesting that the activation function of the gate is a sigmoid function. This makes a lot of sense because all the values are brought down to digits between 0 and 1 and if the information is important the value will be multiplied by 1 and by 0 if the data is no longer important.

As it has been stated before, depending on the value of the gate, $c^{<t>}$ could be overwritten. Computationally speaking this means that:

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * \tilde{c}^{<t>} \tag{2.49}$$

Consequently, if $\Gamma_u = 0$, then the value will not be updated whereas if $\Gamma_u = 1$, $c^{<t>} = \tilde{c}^{<t>}$.

As the models and resources keep on getting more and more complex, it is very common to see diagrams of the algorithms. These images often help to visualize the interaction of the different elements in the model and help understand what exactly is being outputted. In the case of the GRU, the diagram would be:

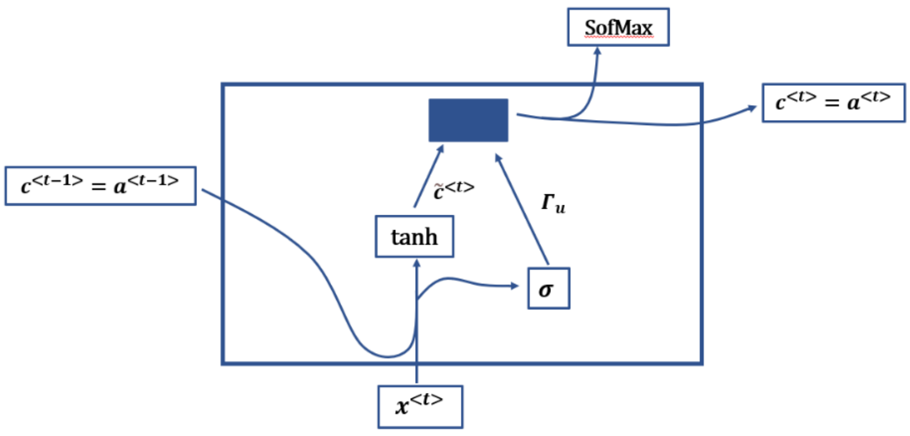


Figure 2-31: GRU diagram

2.3.2.5 Long Short-Term Memory (LSTM) Unit

Just as the GRU, this is another type of unit that enables learning long-range connections in a sequence by remembering certain elements and forgetting other. With regards to the GRU, the main difference is that here there are three gates (instead of only one); one gate will be to *update*, another one to *forget* and the last one to *output* the parameters. The key elements are:

Forget gate: It decides what information should be kept and what should be forgotten. It will receive the data from the previous hidden state and insights from the current input, then it will output a value between 0 and 1; the closer to one, the more importance should be given to the input and therefore kept. If the value is close to zero, the data should be left out.

$$\Gamma_f = \sigma(w_f \cdot [c^{<t-1>}, x^{<t>}] + b_f) \qquad (2.50)$$

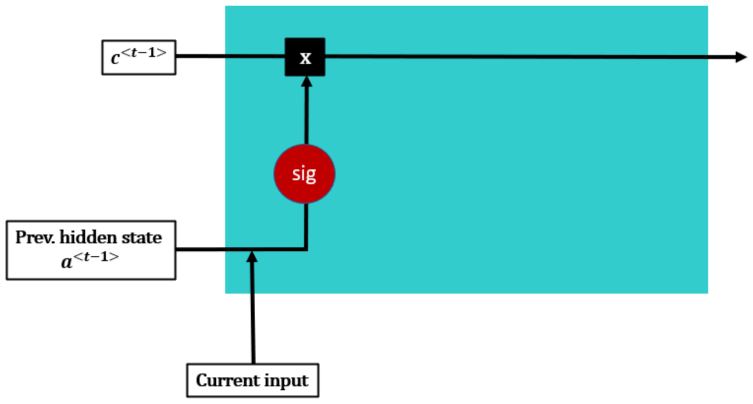


Figure 2-33: forget gate diagram

Update gate: It is used to update the cell state. In this case, the previous hidden state and the current input will be introduced in a sigmoid function, this feature will determine the importance of the data introduced. Simultaneously the same information will be passed on to a tanh function that will help regulate the network. Both outputs will now be multiplied. What is basically happening is that the tanh function adapts the data for the computation and the sigmoid determines how significant this data really is.

$$\Gamma_u = \sigma(w_u \cdot [c^{<t-1>}, x^{<t>}] + b_u) \tag{2.51}$$

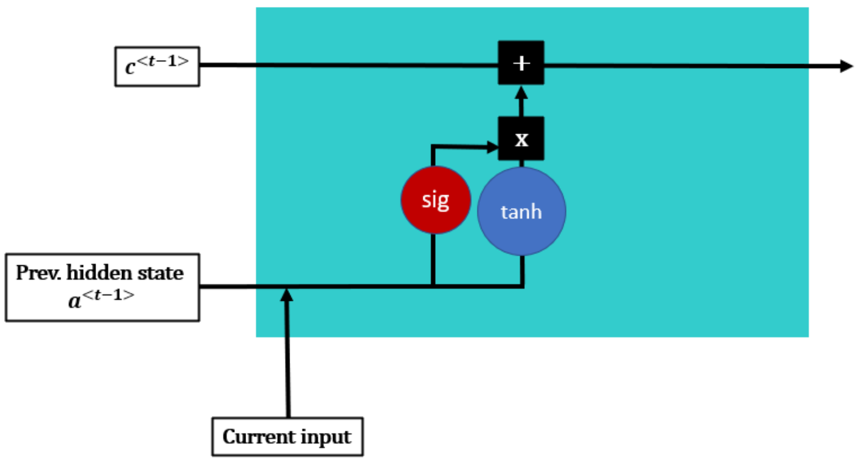


Figure 2-32: update gate diagram

Cell state: It will first be multiplied by the forget vector (output of the forget gate), this can result in the elimination of certain values if they are too close to zero. The next step will be to add up the cell state to the output of the input gate, this will modify the previous cell state by adding values that the NN considers relevant for the forecasting.

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>} \qquad (2.52)$$

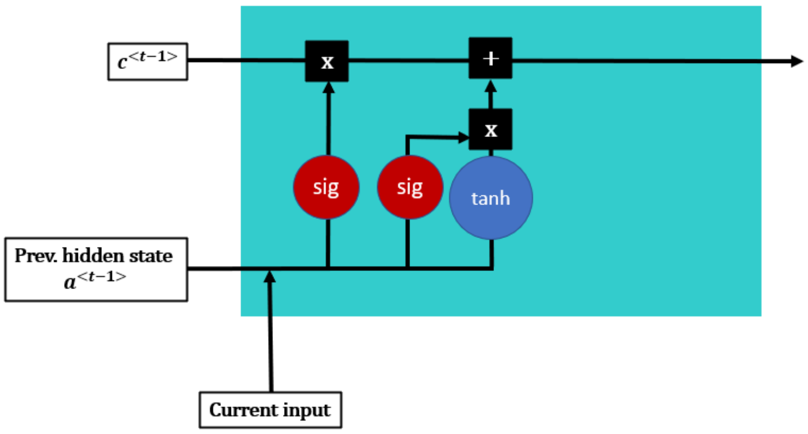


Figure 2-34: cell state diagram

Output gate: It decides what the next hidden state should be. The first step will be to pass on the previous hidden state and the new information to a sigmoid function. Then the new cell state just calculated in (2.52) will be activated with a tanh function. By multiplying both outputs the new hidden state will be created.

$$a^{<t>} = \Gamma_o * c^{<t>} \qquad (2.53)$$

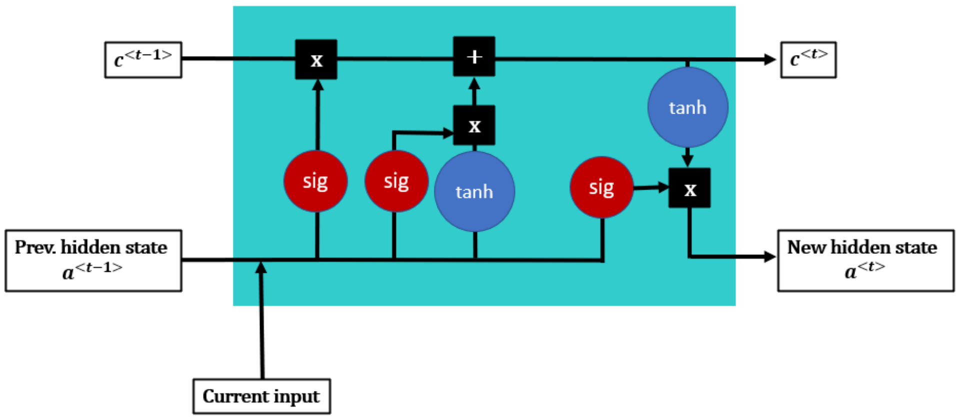


Figure 2-35: complete LSTM diagram

LSTM units are nowadays one of the most used models as they enable the creation of state-of-the-art algorithms which are especially effective in the area of technical analysis. In this project this efficiency will be put to the test as well as many other models.

2.3.3 BIDIRECTIONAL NEURAL NETWORKS

Previously, a lot of emphasis has been placed on the concept of "memory" and the idea that being able to keep this information can have a very positive impact on the performance of the model. However, the possibility of using information from the future has not been contemplated yet.

Until now, when one computed $x^{<3>}$, the only information to be considered could only come from either $x^{<2>}$ or $x^{<1>}$. With the BRNNs, one could also take into account $x^{<4>}$and all the following inputs to compute $x^{<3>}$.

"Teddy bears are on sale!"
"Teddy Roosevelt was a great President!"

Figure 2-36:BRNN illustrative example

The reason why this approach can be so useful is because it is very frequent that the context, by providing more information, helps the performance of the algorithm. By looking at the second example on Figure 2-36, one can see that if the algorithm takes into account that the text is talking about a President, the chances of $\hat{y}^{<2>} = "Roosevelt"$ increase significantly as the randomness of the forecasting is reduced.

The procedure of the BRNNs will consist in two phases; given an input sequence $(x^{<1>}, x^{<2>}, \dots, x^{<4>})$ the forward recurrent layer will first compute $\vec{a}^{<1>}$, it will then use it to obtain $\vec{a}^{<2>}$, then $\vec{a}^{<3>}$ and finally $\vec{a}^{<4>}$. Once this point is reached, the backward recurrent layer will begin and will start by computing $\overleftarrow{a}^{<4>}$ and then go back until $\overleftarrow{a}^{<1>}$ is computed. It is very important not to confuse the forward and backward propagation steps with the forward and backward recurrent layers, these two last ones are both part of the forward propagation step. Once both $\vec{a}^{<1>}$ and $\overleftarrow{a}^{<1>}$ are available then $\hat{y}^{<1>}$ can be calculated. All these concepts can be summarized with the following equation:

$$\hat{y}^{<t>} = g(w_y \cdot [\vec{a}^{<t>}, \overleftarrow{a}^{<t>}] + b_y) \tag{2.54}$$

Or by the following diagram:

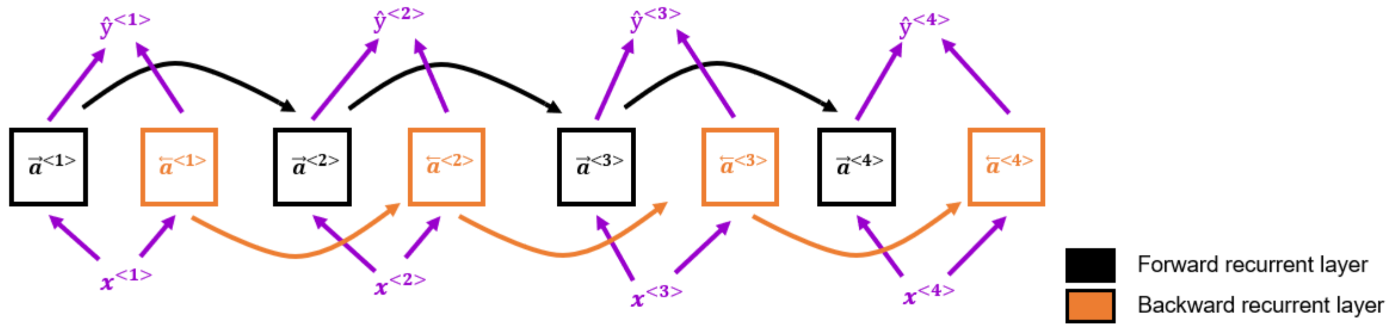


Figure 2-37: BRNN diagram

The main setback of this approach is that, in order to do a prediction, it is necessary to go through all the data first. The consequence of this is for example that the time required will increase significantly and that techniques to reduce the overfitting such as the *Early stopping* are no longer a possibility.

2.3.4 WORD REPRESENTATION

After explaining in depth the most innovative approaches to work with sequences of data and particularly NLP models, the need arises to understand the various alternatives to work with words. As one may know, computers are not able to read words and understand their meanings, however, there are many ways to make possible the association of words with numbers so that the model is able to work with them.

One of the most used techniques to represent words is the *one-hot encoding*, this technique treats each word as an individual element, it assigns to each word a vector filled with 0s but for one position that will be filled with a 1. If the vocabulary had 200 words, then each of the vectors would be 200 dimensional vectors and the number 1 will occupy a different position on each one of them.

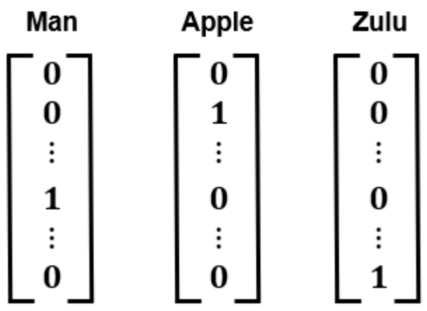


Figure 2-39: one-hot encodings

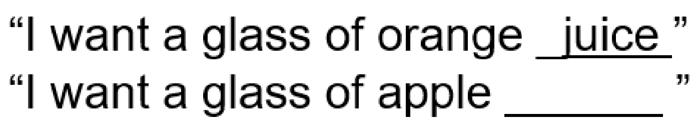
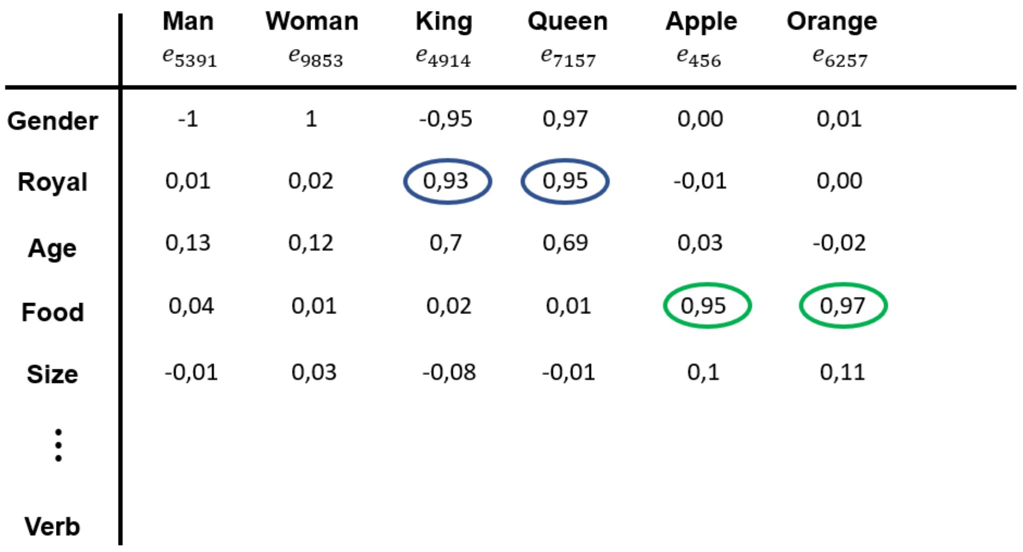

"I want a glass of orange juice "
"I want a glass of apple ______ "

Figure 2-38: generalization problem

The main problem with this technique is that it does not generalize well across words. Looking at Figure 2-38, one can see that the word that should go after "apple" is "juice", this generalization process that humans constantly carry out unintentionally cannot be done with one-hot encodings because with this representation there is no way to show the relationship between "orange" and "apple". This setback can be explained mathematically by demonstrating that the inner product between any one-hot encodings is always zero, this is why the computer is unable to recognize the relationship between elements.

The main alternative to one-hot encodings is the use of *word embeddings*, this approach increases the ability to generalize across words and therefore increases the performance of the NLP algorithms. This method basically creates a matrix whose dimensions are the number of words in the vocabulary and the number of features that will be used to classify the words, this means that for every word there will be an n-dimensional vector where each element will indicate the affinity to each of the features chosen to identify the different words.



	Man e_{5391}	Woman e_{9853}	King e_{4914}	Queen e_{7157}	Apple e_{456}	Orange e_{6257}
Gender	-1	1	-0,95	0,97	0,00	0,01
Royal	0,01	0,02	0,93	0,95	-0,01	0,00
Age	0,13	0,12	0,7	0,69	0,03	-0,02
Food	0,04	0,01	0,02	0,01	0,95	0,97
Size	-0,01	0,03	-0,08	-0,01	0,1	0,11
⋮						
Verb						

Figure 2-40: word embedding representation

As it can be seen in Figure 2-40, for each feature there will be n values (the number will vary depending on the number elements used to classify the words) that will vary from -1 to 1 depending on the strength of the connection between them. Although negative values usually represent a weak link between the word and the classifier, if this last one only has two alternatives (e.g., the gender of the words can only be masculine or feminine) then the extreme values will be used to represent these two options. Values near zero will be used to represent neutrality.

To better illustrate the concepts just explained, one can see that the word "King" (see Figure 2-40) has a very negative value for gender, this means that the word is very masculine, on the other hand, the values associated with royalty and age are, as expected, very high. If one now looks at the food feature, the value is very close to zero, this makes sense since "King" is in no sense related to the concept of food.

Going back to the generalization problem (Figure 2-40), with *word embeddings* it will be easier for the model to conclude that the last word is "juice" taking into account the similarity of both sentences. The problem may come when one has a small training set and the new input sentence uses rare synonyms, however, if the algorithm has learned a word embedding that is able to recognize the characteristics of these elements, then there will be no problem in the forecasting.

There are many ways to make sure that the word embedding is adequate:

- Learn word embeddings from large text corpus (1 to 100 billion words)
- Download pre-trained embeddings online
- Transfer embedding to new task with smaller training set

Once a word embedding is obtained it might be a good idea to continue finetuning it with more data (maybe data more particular of the field trying to analyze).

The main obstacle with word embeddings is that it is very difficult to visualize them due to their n-dimensionality. Many experts have tried to reduce the dimensioning because the visualization of the data often helps others understand the connection between different words and how close different clusters of data are from one another. The most used representation is the T-SNE representation, this reproduction of the embeddings is able to reduce the dimensionality to only two dimensions.

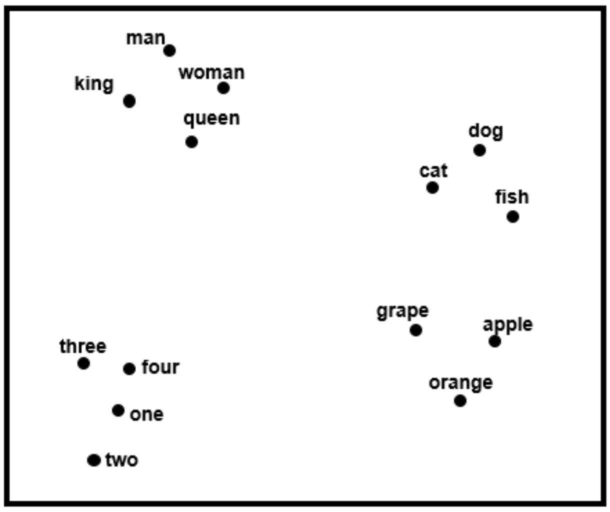


Figure 2-42:T-SNE representation

One of the main advantages of *word embeddings* is that it helps with the analogy reasoning, this means that it is very good at detecting the common factor between two words and then uses this criterion to find the connection between two other words. If, for example, there was an interest in the following match-up:

$$man \longrightarrow woman$$

$$king \longrightarrow ?$$

Figure 2-41: word association example

Then the algorithm would have determined what is the relationship between the two first words and will have looked for a word that gets the most similar outcome from the confrontation with the word "king". This mathematical process is the following:

$$e_{man} - e_{woman} \approx e_{king} - e_{x} \tag{2.55}$$

$$e_{man} - e_{woman} \approx \begin{bmatrix} -2 \\ 0 \\ \vdots \end{bmatrix} \tag{2.57}$$

$$e_{king} - e_{queen} \approx \begin{bmatrix} -2 \\ 0 \\ \vdots \end{bmatrix} \qquad (2.56)$$

However, this process is usually computed using the *cosine similarity* or the *Euclidean distance* (the cosine similarity is much more used): $sim(e_x, e_{king} - e_{man} + e_{woman})$.

$$\text{sim}(u, v) = \frac{u^T \cdot v}{\|u\|_2 \cdot \|v\|_2} \qquad (2.58)$$

Having this said, it is now much easier to explain the procedure to work with word embeddings. The first step will be to obtain an embedding matrix, this will be a database that consists on the embeddings of all the words put together. Once the matrix is available, it will be necessary to multiply it by the one-hot encodings of each of the words in the phrase to obtain its corresponding embedding. However, state-of-the-art models use a specialized function to look up the different embeddings.

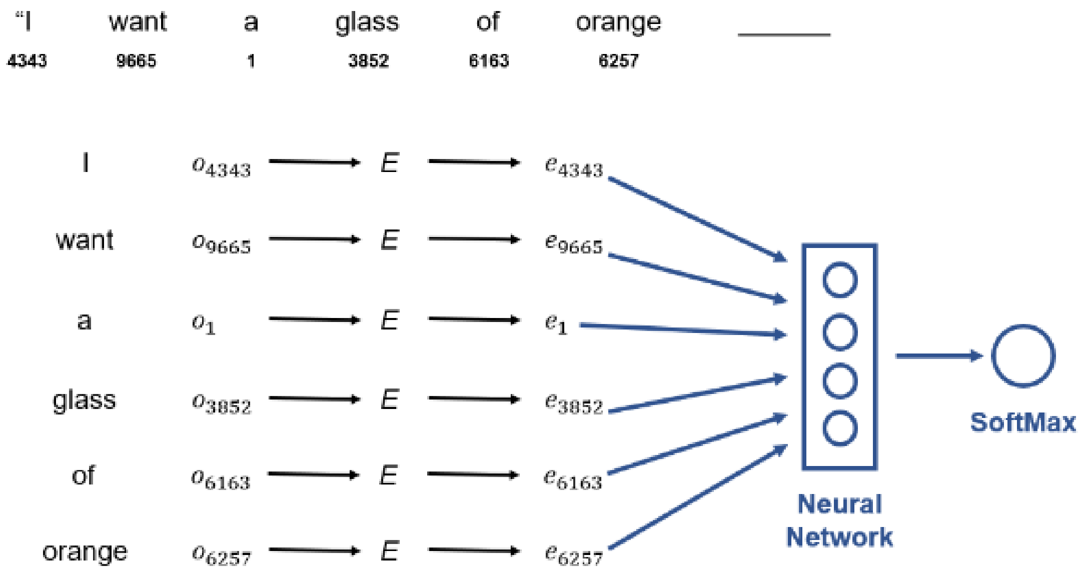


Figure 2-43: NLP with word embeddings

In the case of Figure 2-43, if there were 300 features in the embedding matrix, the total size of the input layer would be an 1,800-dimensional vector (6 words × 300 features). For sentences that are too long this might become a problem due to the computational costs, for this reason programmers tend to focus on the last *n* words in order to predict the last one. By doing so, the dimension of the vector computed is reduced preventing significant performance drops.

Other strategies to reduce the computational costs involve using just the previous word or using a nearby one that can have a significant impact on the forecasting of the following word.

Chapter 3. STATE OF THE ART

3.1 STOCK ANALYSIS

For many years now, the behaviour of company stock values has been the subject of studies by economists, mathematicians and many other experts. The intention of these studies was to create a series of criteria, indicators or even models that could help understand what are the driving forces behind the changes in order to be able to better advise investors. The aim of these elements is to reduce the randomness component of stock movements and thus reduce the risk of generating investment losses.

Throughout history, many models have been proposed to advise buying or selling in the stock market, but of all of them there have been two that have stood out above the rest and are still being used today; fundamental analysis and technical analysis. These two perspectives analyse the value of the stock using very different approaches and therefore different tools.

3.1.1 FUNDAMENTAL ANALYSIS

Fundamental analysis is defined as a financial and accounting methodology that studies the feasibility of an investment through financial statements [12].

The main objective of the analysis is to generate an accurate estimate of the value that the entity should have through a valuation of different parameters. Among many other elements, some of the factors analysed are management effectiveness, financial metrics (assets, liabilities, earnings, etc.), GDP forecasts or macroeconomic indicators such as interest rates. The ultimate goal is to determine whether the company is undervalued or overvalued and thus advise (or not) to invest in a particular asset.

The origin of this analysis dates back to 1934, when University of Columbia professors Benjamin Graham and David Dodd published the work "Security Analysis" [13], which for the first time proposed the idea of a detailed analysis of a company to determine the real value of an asset. This book became one of the most important references in the field of financial markets and its premises are still effective today. As an interesting side note, it is worth mentioning that Benjamin Graham was also Warren Buffet's mentor.

Since this is not the main purpose of this paper, the main types of approaches to fundamental analysis will be described superficially below.

Top-Down approach: focuses on macroeconomic factors (state of the economy, GDP forecasts, unemployment level, etc.). Through these elements, the analyst tries to make a forecast of the general direction and trends of the market in order to have a better judgment when predicting which sectors will prevail and which will not.

Bottom-up approach: analyses the situation from a microeconomic standpoint. The underlying philosophy of this investment approach is that there are stocks that may have a high investment potential even in stagnant sectors. In this approach, elements such as financial statements, business analysis or press releases are analysed.

3.1.2 TECHNICAL ANALYSIS

Although there are many branches within this discipline, technical analysis is based on the analysis of historical data to predict future market behaviour. It is based on the notion that history repeats itself and does so in recognizable patterns [14]. Furthermore, the main advantage of this analysis is the suppression of emotions (fear or greed are elements that significantly influence on one’s market performance).

The origin of this type of analysis dates back to the 18th century in the Far East with the figure of Munehisa Homma [15], the Japanese who introduced the basis of fundamental analysis as well as Sakata charts (today known as candlestick charts). Another important figure in this field of study was Charles Dow, who laid the foundations of modern technical analysis (Dow Theory) as well as introducing the Dow Jones Industrial Index. Some other pioneers in the industry who managed to optimize the technique were William P. Hamilton, Robert Rhea, Edson Gould or John Magee [16].

The three basic principles of technical analysis are [14], [17]:

<u>Markets discount everything</u>: considers that markets are a point of exchange of information and these reflect what elements influence investment decisions and what do not. this concept is popularly explained by saying that the influencing factors are "priced in" and therefore the price tells everything one needs to know to understand market trends.

<u>Market inertia</u>: proposes that trends continue until they are challenged by a "breaking" element.

<u>Markets move in waves</u>: if a company announces something important, this results in a disruption in the market equilibrium that can be compared (metaphorically speaking) to the behaviour of a wave. The trader's objective is to catch the wave at the bottom, ride it to the top and then surf it. If the metaphor is set aside, what is meant is that the analyst should be able to exploit changes in the value of the stock (taking advantage of the ups and downs) to generate profits.

3.1.3 FUNDAMENTAL VS. TECHNICAL ANALYSIS

After having talked a little about each of the main schools of thought separately, it is now convenient to confront them to see what are the main characteristics of each one of them.

For this purpose, the following table will be used:

	Fundamental Analysis	**Technical Analysis**
Function	More suitable for investing	More suitable for operating
Objective/Target	Determine whether a share is overprized or underprized	Finding the best time to enter or exit the market
Used by	Long-term investors	Short-term investors
Information treated	Financial statements, balance sheets, corporate communications, etc.	Price and trading volume
Application	Stocks, bonds, derivatives and more	All asset classes
Time range	Long periods of time	Short periods of time
Origin	First proposed in 1934	First proposed in the 18th century

Table 3-1: Fundamental vs. Technical analysis

With respect to the debate as to which of the two is better, it is important to understand that they are both very useful methods that serve different purposes. Therefore, it cannot be said that one is better than the other; in fact, combining them is not a bad idea and may even improve the accuracy of one's forecasts.

3.2 DEEP LEARNING AND STOCK PREDICTION

Although both approaches are very useful and it has been shown that they can both be used to generate profitability, throughout history there have been times when, due to various factors, there has been a greater tendency to use one or the other. In particular, following the publication of the Columbia professors, fundamental analysis became the primary methodology for asset valuation and remained the main reference point until the digital revolution reached the financial markets. From that point on, fundamental analysis took a back seat and the focus shifted to technical analysis and other computational methodologies.

The reason why technical analysis gained momentum again is because it is an approach based on the pursuit of behavioural patterns. This search was strongly reinforced with the advent of new technologies because they improved the ability to identify behavioural indicators and thus increase forecasting skills.

Through new technologies, numerous models for asset prediction have been developed; these models can be classified into three groups [18]:

<u>Linear models</u>: they use a single formula to generate a line that is capable of optimally representing the data provided. It is a model that is easily interpreted and this has allowed it to be used for a large number of applications in the stock market. Its first application was in the early 1980s through the Arima (or Arimax) model [19], which was designed to analyse the average of an asset. Another important breakthrough came in 1999 [20] when it was confirmed that Garch models could be used to predict stock market volatility. Later, in 2010, this type of model was used to demonstrate that the asymmetry of the data is a fundamental element for predicting the volatility of a market [21].

<u>Non-linear methods</u>: this is an approach in which the information used is considered to behave in a way that cannot be represented by a straight line. It is a very useful resource because economic theory emphasizes the presence of cycles and nonlinearities in the behaviour of stock prices. Neural Networks are a well-known type of nonlinear model. Some

examples of applications are; in 1999 Machine Learning tools were used to predict the value of stocks in small markets such as the Warsaw Stock Exchange [22]. In 2008 [23] experts began to predict stock values using Backpropagation Neural Networks. In 2011 [24] it was proved that Deep Learning models were significantly more efficient than other approaches.

Hybrid models: consists of the use of two or more methods or models to improve the prediction. Some famous models are that of Pai and Lin in 2005 [25] where they propose to jointly use the Arima model and the S.V.M. (Support Vector Machine) model, another interesting example is the work of Zang and Wu (2009) [26] who proposed the hybrid use of backpropagation neural networks and IBCO (Improved Bacterial Chemotaxis Optimization) to reduce the mean square error of the predictions. One of the last contributions to the field is the paper of Asadi, Hadavandi, Mehmanpazir and Nakhostin (2012) that proposed a combination of models of data processing, genetic algorithms as well as LM (Levenberg-Marquardt) models for stock prediction [27].

Although all these different methods allow the computational skills of the model to be improved, another way to improve performance is through the use of more complex input variables such as macroeconomic factors or other indicators related to the market or sector in which the company operates [28], [29].

Nowadays, in addition to everything indicated so far, the impact of social networks on stock performance is beginning to be taken into account. The reason for this is because it has been evidenced that events can occur in these media (influencers' publications, activity in chat forums, etc.) that trigger changes in the stock market [30]–[32] (this is further explained in chapter 5.2.2).

Chapter 4. JUSTIFICATION AND OBJECTIVES

4.1 JUSTIFICATION

As indicated in the previous chapter, there has been a transition in the approach used for the optimization of predictive models. The reason for this transition is that it has been demonstrated that there are elements that indirectly have an influence on the behavior of the stock.

The fact that this influence is indirect then raises the uncertainty as to whether all factors are being taken into account or whether there is information that is not being captured that helps understand market trends.

All of this is coupled with the fact that digitization has enabled time optimization in almost every aspect of society and, with it, market timings have also been reduced, making it much more unstable than it was twenty years ago. This increase in market volatility has resulted in investors demanding a greater understanding of the sector and therefore a demand of models that are able to make more efficient short-term forecasting.

As there has been a growing interest in the use of information from social media, a proper execution of the project intends to shed light on the impact and the importance that should be given to this kind of data.

Moreover, another problem that has arisen from the development of technologies is the increase in the value of information. This rise in value has resulted in many individuals opting to make a profit from their access to data by privatizing it, resulting in a limitation of the publicly available material. The problem with limiting information is that this often leads (if not paid for the access) to a significant limitation of the model, since the more data there is, the more different cases there are and the better the model generalizes afterwards.

In the case of researchers for large companies this is usually not a problem since the company can afford these costs, however, in the case of startups this is a very big effort that carries a great risk (if the expected returns are not achieved this may result in the end of the company).

To help these investigators, free (opensource) platforms have been created in order to encourage the sharing of data and knowledge to achieve more efficient models. However, there is a lot of criticism regarding the use of these platforms as it is said that the yields obtained are very poor due to the quality of the data not being adequate. In this project it is intended to demonstrate that the quality of the data is good enough to obtain more than acceptable returns in the prediction of stock market values.

The objectives of this project align with the Sustainable Development Goals (SDG) related to economic growth (SDG 8) as well as the fight against inequality (SDG 10) and quality education (SDG 4)[33].

4.2 OBJECTIVES

This chapter describes the objectives that were set at the start of the project, the achievement of these will lead to the successful completion of the project. The aims of this project will be divided into two; primary goals and secondary ones.

4.2.1 PRIMARY OBJECTIVES

The main objectives will be the following:

1. ***Creation:*** The creation of three models of varying complexity that allow predicting, based on different input data matrices, the value that a company stock should have. The first model will be one that uses logistic regression to predict outputs whereas the other two will be models that use artificial neural networks. The difference between these last two algorithms is that one will have one hidden layer while the other one will have the flexibility to change its dimensions (for a deeper analysis).

2. ***Analysis:*** Each of the models will be analyzed separately and it will be determined which one has the best forecasting ability. In addition, it will also be determined which type of data helped obtain the best results.

3. ***Hyperparameter tuning:*** the performance of each model can be significantly enhanced through the analysis of the different hyperparameters and their subsequent optimization. The purpose of this work will be to determine which of these indicators have the greatest influence on the performance of the model.

The achievement of these main objectives will allow the generation of solid conclusions with respect to the purpose of this project that will help others to better understand the area being studied.

4.2.2 SECONDARY OBJECTIVES

Secondary objectives are those that are intended to be accomplished that will help the student's professional training and development as an engineer.

1. **Reinforce** the **knowledge** of the programming language **Python** (all the coding on this project Will be based on this language). The aim is to improve the programming skills in one of the most used languages in the programming community.

2. **Acquire knowledge** about **Deep Learning**, more precisely artificial Deep Neural Networks as well as Natural Language Processing (NLP). These are critical elements for the creation and implementation of the algorithms explained in the main objectives.

3. **Familiarize with the use of databases**: learn how to organize, normalize and store the data (data cleansing) in order to facilitate the later use of it as training and test set of the model.

4. **Learn how to work with opensource databases** and familiarize with the notions of **Transfer Learning** in order to increase the efficiency of the model.

5. **Develop critical thinking** in order to decide what values should be given to the different hyperparameters and meta-parameters in the algorithm in order to enhance its performance.

Chapter 5. METHODOLOGY

Once all the theoretical concepts that have been used throughout this project have been explained, it is necessary to give a detailed explanation of how they have been applied and the methodology used to implement them in order to reach the objectives stablished. This methodology has been structured as transparently as possible to facilitate the understanding of all the measures taken and the criteria used to make these decisions.

The criterion used for the wording of the methodology of this project is based on the chronological evolution of it. Therefore, the first element to be discussed will be the obtaining of a suitable database, followed by the selection and creation of the different models used for the analysis of the available data and finishing with the explanation of the processes used for optimizing the algorithms.

Although it has been said before, it is important to remember that all the algorithms that have been used in this project have been programmed using the Python programming language and the code editor used has been Atom.

5.1 FORMATION OF THE STUDENT

The first step of the student's training consisted of attending the course "Introduction to Machine Learning" imparted by the tutor of this project, Marco Lippi. This course was part of the curriculum for students studying for a PhD in Industrial Innovation Engineering and consisted of a brief introduction to the world of Artificial Intelligence as well as an overview of the most frequently used prediction models and the state-of-the-art ones nowadays.

After this first course, two other specialized programs would follow in which helped the student to acquire the skills that would be used to program and execute the different algorithms used in this project. The first specialized program consisted in 3 courses offered by the University of Michigan (via Coursera) and they were focused on the development of skills related to the use of Python and data structures as well as the use of Python to access web data. The second specialized program was the most important one, it was imparted by the company Deeplearning.ai (via Coursera) and it consisted on 5 courses that analysed in depth all the concepts connected with Deep Learning, the structuring of Machine Learning projects, Deep and Convolutional Neural Networks as well as hyperparameter tuning and model optimization. This learning process of the student required approximately 3 months (November-January).

5.2 DATABASE SELECTION

The first key element to discuss is the selection of the database to work with. This choice is crucial because a balance must be sought between data variety, data quality and data quantity. Moreover, in this project it is necessary to look for two different types of data; on the one hand, a large numerical database that allows to see the evolution of the stock of a company over time and, on the other hand, a database of publications in social networks that talk about the company in the same period of time.

Before starting, it is necessary to understand what exactly is the target, since it is not a search of elements that can be done separately. It is a search in which it is necessary that both databases coincide in temporality; if it is decided to use the monthly average value of a stock then all the tweets of those dates must be collected and grouped in that same period of time. In addition, in order to carry out an in-depth analysis, it has been used, instead of the information of a single company, that of the main companies in a sector; by doing so, it is also possible to take into account the influence that the different competitors have on the company. However, this last information will only be used in the Deep Neural Network, as it is too complex for the other two.

5.2.1 STOCK VALUES

Regarding the numerical data, anyone who is minimally familiar with the world of the stock market knows that the historical information of any listed company is collected and is of public domain, therefore anyone can go to Yahoo Finance (among others), search for the historical data of a particular company and download it to their computer. Taking this into account, the real challenge has to do with the textual information.

The only call that had to be made regarding this kind of data was the type of share value to be used. When downloading these numerical datasets, one has to know that the dataset consists on the date of the values, the minimum value of the stock, the maximum, the opening price, the closing price and the volume exchanged. Depending on the value used the performance of the algorithm may vary. However, later on (chapter 5.3) it will be seen that, for a particular reason, the only price that can be used is the closing price.

5.2.2 TWITTER PUBLICATIONS

Before discussing how the database that was later used in the project was obtained, it is important to explain why it was decided to use publications in social networks and why this source of information was chosen instead of more traditional methods such as newspapers.

5.2.2.1 Social Networks

Social networks initially emerged as a form of communication between friends; however, the proliferation of different platforms (Facebook, Twitter, Instagram, LinkedIn) has enabled all network users to have access to a wealth of information and, in turn, to share not only additional information, but also their opinions and points of view.

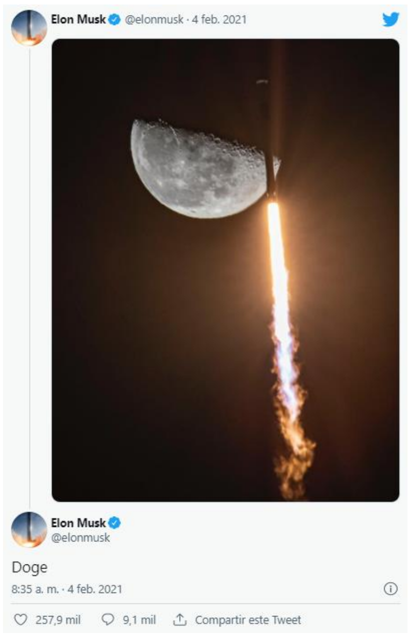
The impact of these exchanges has a direct effect on many social behaviors. Whereas before the massive use of social networks, it was the recognized experts who gave their opinion on the objective value of a stock and recommended its purchase, sale or maintenance, today many network users also share their ideas and influence market behavior in the same way without necessarily implying that they are experts or that they have a more valid or educated opinion than others.

The reason why it has been decided to choose these publications instead of those of the mass media is because it has been demonstrated [31] that the impact of the networks on the behavior of a company's actions is greater than that produced by the conventional media.

Building on the above, the figure of an influencer seems closer to the general public, more approachable, and therefore their opinions can have more impact than a traditional financial advisor regardless of the fact that their academic background is, in general, much poorer.

For further insight, the following example will be introduced.

On February 4th, Elon Musk, Co-founder of PayPal and CEO of Tesla Motors published the following tweet:


Elon Musk @elonmusk · 4 feb. 2021

Elon Musk
@elonmusk

Doge

8:35 a. m. · 4 feb. 2021

257,9 mil 9,1 mil Compartir este Tweet

Figure 5-1: Elon Musk's tweet. Source: Twitter

“Doge” was a refers to the cryptocurrency Dogecoin which started its activity from an Internet meme. Specifically, its creators (Billy Markus and Jackson Palmer, former IBM and

Adobe employees respectively) were inspired by the viral jokes made with a dog (Shiba Inu). The consequence of this tweet was the following:

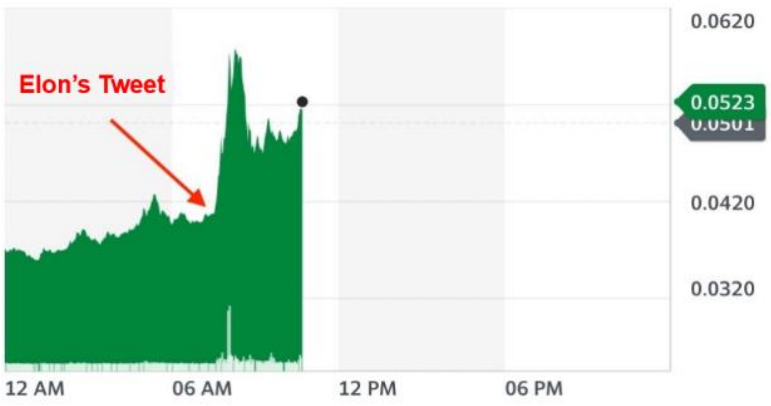


Figure 5-2: evolution of Dogecoin after Elon Musk's tweet. Source: Coin Market Cap

As it can be seen in Figure 5-2, in a matter of minutes, the share price soared and stabilized throughout the day, registering a growth of 45%. After this tweet came a few others and by the end of the month Dogecoin registered an approximate growth of 700% [34].

This example is not intended to publicize the influence of Tesla's CEO, but to show how someone, not necessarily tied to a company or a currency, can affect its stock market performance in a significant way.

5.2.3 DATABASE

Having said that, the first choice to be made was which Social Network to use. The three main options considered (due to their high usage) were Instagram, Facebook and Twitter. However, Instagram was rapidly dismissed due to the fact that most of the information is transmitted through images, this meant that the algorithm used would have to be a highly efficient Convolutional Neural Network (CNN) and this would not guarantee a high performance in the stock forecasting.

The choice between the other two alternatives was not simple, both networks were widely used and the use of sentences to express oneself was much more frequent. The reason why

Twitter was chosen in the end was mainly due to the fact that the publications had character limitations (which reduced the complexity of the inputs) and, in addition, it is a social network where there is a very high participation in discussion forums. This participation in the forums encouraged the expression of sentiments (towards the subject debated) through short sentences, which is very helpful for sentiment analysis algorithms. Moreover, most of the cases where it has been proved that the social networks affect the value of a company's shares were through the information posted in Twitter [31].

One may also wonder why Reddit was never considered, the reason for this is because it is an emerging social network that has very strict APIs and therefore it became very difficult to gather a significant quantity of information in order to obtain remarkable performances in the posterior computation of the data.

After a period of searching for an appropriate database, it was decided to download one that had been used for a study similar to the one in this project [35]. This database was available on GitHub (open source) and consisted of, for 88 companies studied, a daily collection of tweets that talked about them for two years (2014 & 2015).

5.2.3.1 Data Cleansing

The main problem of this database was that, depending on the company, there were days where no data was collected. This is a major setback, because when running the algorithm if the data had missing values the computation would stop and no results would be obtained. The solution for this had two parts; the first part consisted on choosing the business industry that had the biggest amount of data possible, this was the technology sector and it comprised 6 of the most important enterprises of the industry:

Sector	Stock Symbol	Company
Technology	$GOOG	Alphabet Inc.
	$FB	Facebook, Inc.
	$T	AT&T Inc.
	$VZ	Verizon Communications Inc.
	$INTC	Intel Corporation
	$CSCO	Cisco Systems, Inc.

Table 5-1:companies to work on

However, even though this was the sector with the largest amount of data available, there were still days on which there were no posts. To solve this problem, an algorithm was created to report the days on which, for each company, no publications had been registered.

Once the missing days were available, the files for those dates were created manually and a single publication was written in them in which only the company’s acronym was written. The reason for doing this was simple, if an empty file was inserted, the algorithm would indicate that it could not read the file and therefore it would stop runing, however, if only the acronym was inserted the algorithm would read it but it would associate a neutral feeling (a number very close zero), this is perfect because later on, when the sentiment analysis algorithm is explained, it will be seen that the dates on which no tweets were published will have a neutral value, which is the same as saying that nothing important happened on that date that could change the value of the company's share.

Until now, there was a .json file for every date registered. Inside these files one would find that for each post there would be three different keys; “text”, “created at” and “user_id_str”. For security reasons, the user IDs had been encrypted and transformed into numbers so that it was not possible to identify whether this user was an influential person who could reach many people or not. For this reason, this element was not useful for prediction and was therefore discarded. The next step would then be to create a model that would go through all the .json files and extract both the “text” and “created at” keys. Once they were extracted, the tweets would directly get stored in a matrix whereas the tweets would first undergo a computation through which the date format was modified so that it was only displayed the day, month and year of the publication (as the numerical data was daily, it did not make sense to store the hours of the publications).

Once both pieces of information where stored in matrices, they where saved into another .json file but this time it only had two pieces of data, two matrices that had all the tweets posted throughout the two years and the dates of their publication.

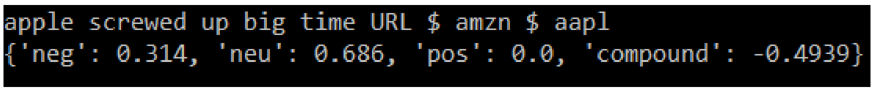
5.3 SENTIMENT ANALYSIS

Once the data is ready to work with, it is necessary to create an algorithm that is capable of transforming the sentences, with the tweets as input, so that they can be used to predict the value of the company. As explained before, computers are not able to read a sentence and interpret it, but there are tools (one-hot encodings, word embeddings, etc.) that allow to transform the words so that they can be analyzed. In this case a step further will be taken and these resources will be used for the creation of an algorithm known as Sentiment Analysis.

A Sentiment Analysis algorithm is able to, receiving as input a sentence, generate a number that scores the emotion conveyed in that message. This score is a value ranging from -1 to 1 where negative values are associated with pessimism and positive ones with optimism.

Initially, an attempt was made to create the algorithm from scratch (it consisted on an LSTM model that used Deep BRNNs), but in order for it to output accurate scores it required an amount of clean training data that was not possible to obtain (at least 500,000 sentences). For this reason, an open-source pre-trained algorithm was used. This model, known as VADER (Valence Aware Dictionary and sEntiment Reasoner) [36], is a Sentiment Analysis tool that is specifically designed to detect the sentiments expressed in social media. This algorithm uses Deep Bidirectional Neural Networks as well as LSTM units to increase its efficiency.

The use of this model was very simple; a statement is inputted and a vector is generated that separately analyzes the pessimism, neutrality and optimism conveyed in the sentence. In turn, a compound value is also generated that summarizes the sentiment of the phrase.

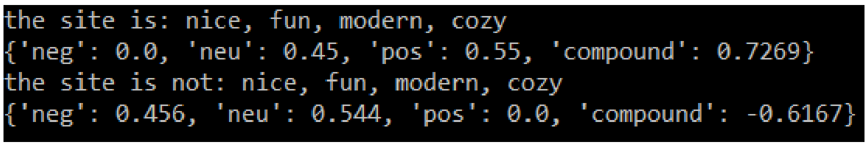


```
apple screwed up big time URL $ amzn $ aapl
{'neg': 0.314, 'neu': 0.686, 'pos': 0.0, 'compound': -0.4939}
```

Figure 5-3: computation example

Looking at Figure 5-3, it is a sentence with high pessimism, this can be seen in the use of words such as "screw up" coupled with sentiment intensifiers such as "big time". As it can be seen, the algorithm is able to perceive this negativity and registers it by assigning a high level of pessimism, but above all by generating a compound value that differs very little from that of pessimism, this means that, although there are more words in the sentence (mostly neutral), the model recognizes the importance of this words.

Building on the previous example, another element that makes the VADER algorithm valuable is that it is a model that has the ability, on the one hand, to recognize sentiment intensifiers and, on the other hand, to take into account the chronological order of the words. This last advantage is very important, because one of the major problems of Sentiment Analysis algorithms is that they evaluate words individually and do not take into account the words that precede them, with the result that phrases such as "the site is: nice, fun, modern, cozy" and "the site is not: nice, fun, modern, cozy" generate very similar outputs when they mean the opposite. If the sentences just explained were computed, this would be the consequence:

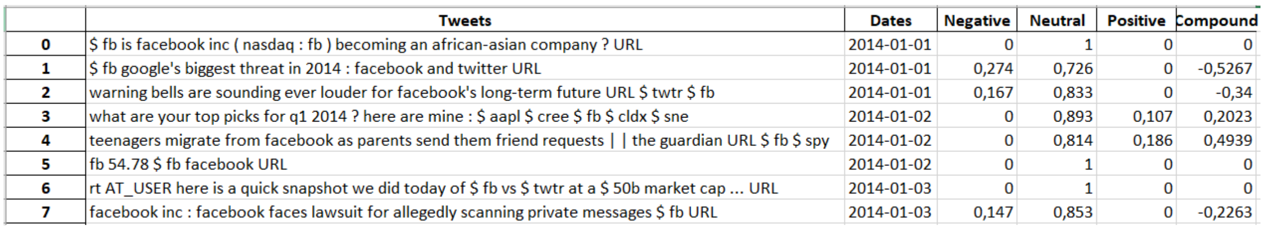


```
the site is: nice, fun, modern, cozy
{'neg': 0.0, 'neu': 0.45, 'pos': 0.55, 'compound': 0.7269}
the site is not: nice, fun, modern, cozy
{'neg': 0.456, 'neu': 0.544, 'pos': 0.0, 'compound': -0.6167}
```

Figure 5-4: computational example (2)

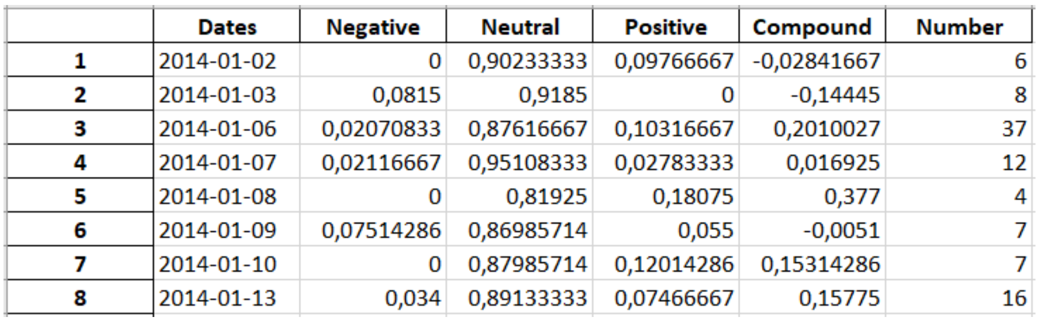
Just as expected, the VADER algorithm does not fall into the trap and outputs appropriate values for each situation.

Returning to the methodology, once the sentiment algorithm is ready, two Excel documents will be created; on the one hand, a file will be created in which all the company's publications during the two years will be collected (together with their ratings); in this case it is important to separate the different types of information by columns, as this will facilitate the selection of data (if necessary) in the predictive models. On the other hand, a file is created in which, for each day, the average number of ratings and the number of publications per day are collected. The reason for collecting the number of publications per day is because the days when there is more activity in the networks (in the stock market forums) usually coincide with the days of greater instability in the stock market, therefore, it is important to take into account this hyperparameter.



	Tweets	Dates	Negative	Neutral	Positive	Compound
0	$ fb is facebook inc (nasdaq : fb) becoming an african-asian company ? URL	2014-01-01	0	1	0	0
1	$ fb google's biggest threat in 2014 : facebook and twitter URL	2014-01-01	0,274	0,726	0	-0,5267
2	warning bells are sounding ever louder for facebook's long-term future URL $ twtr $ fb	2014-01-01	0,167	0,833	0	-0,34
3	what are your top picks for q1 2014 ? here are mine : $ aapl $ cree $ fb $ cldx $ sne	2014-01-02	0	0,893	0,107	0,2023
4	teenagers migrate from facebook as parents send them friend requests | | the guardian URL $ fb $ spy	2014-01-02	0	0,814	0,186	0,4939
5	fb 54.78 $ fb facebook URL	2014-01-02	0	1	0	0
6	rt AT_USER here is a quick snapshot we did today of $ fb vs $ twtr at a $ 50b market cap ... URL	2014-01-03	0	1	0	0
7	facebook inc : facebook faces lawsuit for allegedly scanning private messages $ fb URL	2014-01-03	0,147	0,853	0	-0,2263

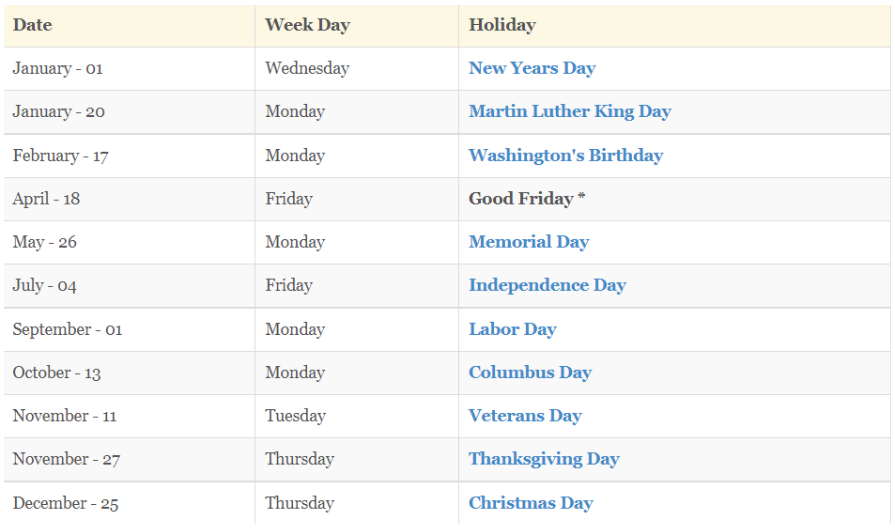
Figure 5-5: extract of Excel 1



	Dates	Negative	Neutral	Positive	Compound	Number
1	2014-01-02	0	0,90233333	0,09766667	-0,02841667	6
2	2014-01-03	0,0815	0,9185	0	-0,14445	8
3	2014-01-06	0,02070833	0,87616667	0,10316667	0,2010027	37
4	2014-01-07	0,02116667	0,95108333	0,02783333	0,016925	12
5	2014-01-08	0	0,81925	0,18075	0,377	4
6	2014-01-09	0,07514286	0,86985714	0,055	-0,0051	7
7	2014-01-10	0	0,87985714	0,12014286	0,15314286	7
8	2014-01-13	0,034	0,89133333	0,07466667	0,15775	16

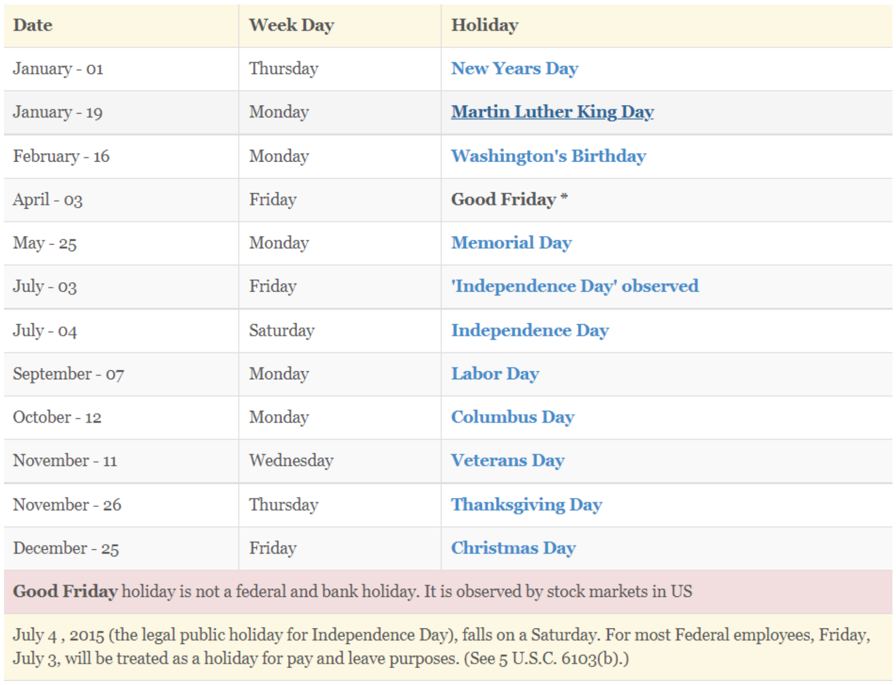
Figure 5-6: extract of Excel 2

Before talking about the models, there is one last roadblock to talk about. Once the two Excel files have been created, there is a final modification that must be made so that the computation does not give problems. As explained at the beginning of the methodology, there needs to be temporal equality between the numerical data and the data produced with the Sentiment Analysis algorithm, this temporal equality goes beyond the data having a daily or temporal frequency, there needs to be a share value and a sentiment score for each day. The problem is that every day things are posted on social networks, however, not every day shares are quoted. To be more precise, Saturdays, Sundays and national holidays are the days when there is no stock trading in the US. In addition to the weekends, these were the days where the stock exchange was closed in 2014 and 2015.



Date	Week Day	Holiday
January - 01	Wednesday	New Years Day
January - 20	Monday	Martin Luther King Day
February - 17	Monday	Washington's Birthday
April - 18	Friday	Good Friday *
May - 26	Monday	Memorial Day
July - 04	Friday	Independence Day
September - 01	Monday	Labor Day
October - 13	Monday	Columbus Day
November - 11	Tuesday	Veterans Day
November - 27	Thursday	Thanksgiving Day
December - 25	Thursday	Christmas Day

Figure 5-8: public holidays in 2014



Date	Week Day	Holiday
January - 01	Thursday	New Years Day
January - 19	Monday	Martin Luther King Day
February - 16	Monday	Washington's Birthday
April - 03	Friday	Good Friday *
May - 25	Monday	Memorial Day
July - 03	Friday	'Independence Day' observed
July - 04	Saturday	Independence Day
September - 07	Monday	Labor Day
October - 12	Monday	Columbus Day
November - 11	Wednesday	Veterans Day
November - 26	Thursday	Thanksgiving Day
December - 25	Friday	Christmas Day
Good Friday holiday is not a federal and bank holiday. It is observed by stock markets in US		
July 4 , 2015 (the legal public holiday for Independence Day), falls on a Saturday. For most Federal employees, Friday, July 3, will be treated as a holiday for pay and leave purposes. (See 5 U.S.C. 6103(b).)		

Figure 5-7: public holidays in 2015

To solve this problem, it was decided to combine the information from the days when there was no stock market trading with the next day when there was, thus showing how the value of the stock (the first day when it was trading) is affected by what happened on the days when there was no market activity. For this to be possible, it is strictly necessary to use the closing price of the share as an input (if the opening price was used, then the computation would be much complicated).

To better illustrate the effectiveness of this decision, the following example will be used. If on December 25 (a national holiday) it is discovered that the CEO of a listed company is corrupt, on the same day there will be many publications discussing the news, however, the value of the stock will not be affected because the stock market was closed that day. The following day, activity on the forum will be significantly reduced, but since the previous day's postings are taken into account, the rating attributed to that day will do justice to what happened the day before.

Once this last problem has been solved, all databases are in the right configuration to be computed in such a way that no relevant information is lost during the process.

5.4 FORECASTING ALGORITHMS

After having talked about the methodology used to choose the database, after having explained the different problems encountered and the solutions used to provide a suitable database, it only remains to talk about the methodology used to create the predictive algorithms.

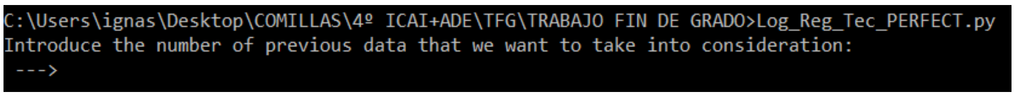
So far it has been talked about algorithms that have been used to treat and clean the data, these algorithms are very important because without them it is not possible to have a reliable source of information, however, the models that will be explained below are those that compute these data and generate predictions of future values of the shares.

Since there are several models, it has been decided that they will be explained in order of increasing difficulty, therefore, the logistic regression model will be discussed first and then the Neural Networks will be discussed, starting with the model that has only one neuron and then moving on to those that have greater depth.

5.4.1 LOGISTIC REGRESSION

The first thing that has to be done whenever describing a forecasting model is to explain how to enter the data (X matrix). Logically, depending on whether one is dealing with historical stock values or sentiment scores (or both at the same time), the dimensions of the matrix will vary but, in general, the goal is to create a code that works exactly the same regardless of the type of input data. In this way it is possible to see which ones provide more information to the algorithm (through better performance).

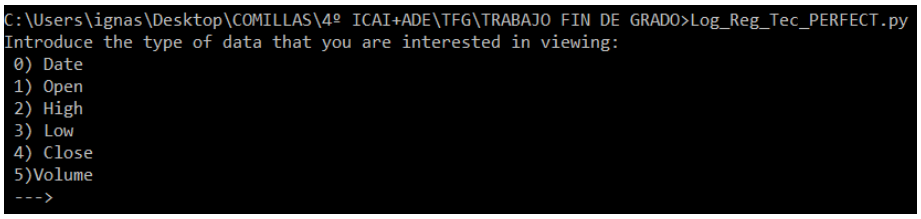
In relation to the creation of the X matrix, it is necessary to take into account that one of the most important hyperparameters in forecasting is the determination of the amount of data to be considered for each input, in other words, the number of days to be taken into account for the prediction of a single date. As one can imagine, the performance will be different if one considers only the information of the previous day or that of the last 15 days. For this reason, the algorithm starts by requesting the number of previous days to be considered.



```
C:\Users\ignas\Desktop\COMILLAS\4º ICAI+ADE\TFG\TRABAJO FIN DE GRADO>Log_Reg_Tec_PERFECT.py
Introduce the number of previous data that we want to take into consideration:
 --->
```

Figure 5-9: number of previous days to consider

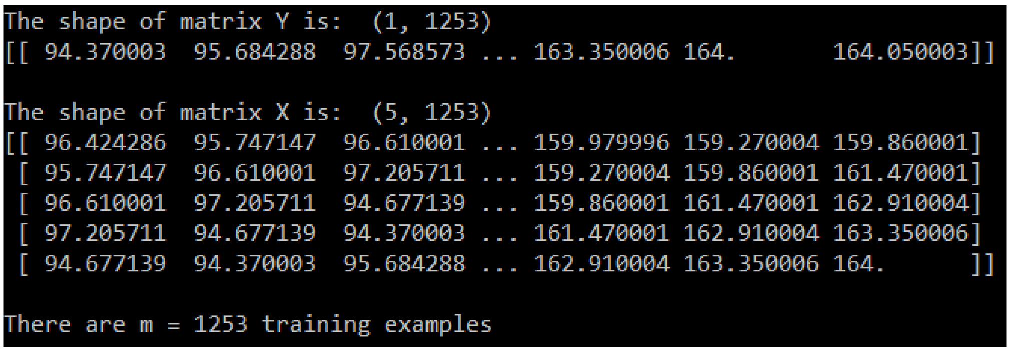
Although when working with sentiment data the type of share value is preset (closing price), when working only with share values there is the possibility of choosing the type of data to be analyzed, this choice becomes another hyperparameter to compute:



```
C:\Users\ignas\Desktop\COMILLAS\4º ICAI+ADE\TFG\TRABAJO FIN DE GRADO>Log_Reg_Tec_PERFECT.py
Introduce the type of data that you are interested in viewing:
 0) Date
 1) Open
 2) High
 3) Low
 4) Close
 5)Volume
 --->
```

Figure 5-11: type of data to use

Once these hyperparameters have been selected, the first function of the algorithm can be activated. This first function creates the X and Y matrices having as inputs the name of the company and the previous hyperparameters.

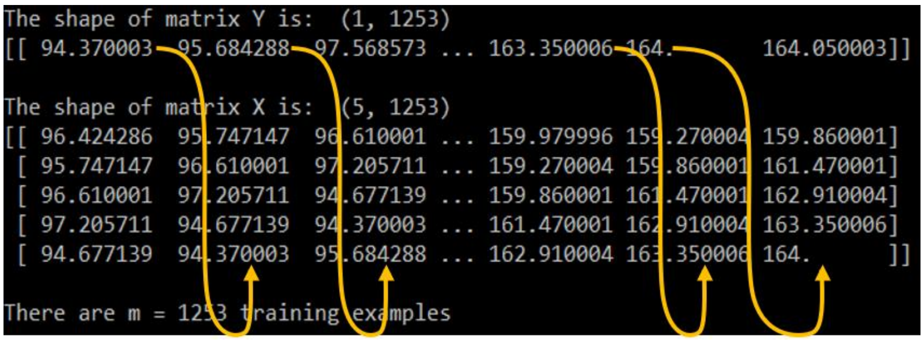


```
The shape of matrix Y is:  (1, 1253)
[[ 94.370003  95.684288  97.568573 ... 163.350006 164.       164.050003]]

The shape of matrix X is:  (5, 1253)
[[ 96.424286  95.747147  96.610001 ... 159.979996 159.270004 159.860001]
 [ 95.747147  96.610001  97.205711 ... 159.270004 159.860001 161.470001]
 [ 96.610001  97.205711  94.677139 ... 159.860001 161.470001 162.910004]
 [ 97.205711  94.677139  94.370003 ... 161.470001 162.910004 163.350006]
 [ 94.677139  94.370003  95.684288 ... 162.910004 163.350006 164.      ]]

There are m = 1253 training examples
```

Figure 5-10: value_extractor(filename, h1, h2)

As it can be seen in Figure 5-11, if it is decided to consider the five days prior to the day to forecast, then the input matrix will have five rows and will have five columns less than the maximum possible (in this case 1258), this is because if it is decided to use the information from five days in advance, then there is no possible way to compute the input data for the first five values. Meanwhile, the matrix of outputs will have the same number of columns but only one element (or row) per column.

Something that has been used as a checkpoint to make sure that the performance of the function is adequate is the fact that the output of a particular vector of data has to be the last element of the input vector of the next training sample. This must be fulfilled because as it is a chronologically ordered database, the output of one day is one of the inputs of the next one; if this is not fulfilled the function is not working properly.



```
The shape of matrix Y is:  (1, 1253)
[[ 94.370003  95.684288  97.568573 ... 163.350006 164.        164.050003]]

The shape of matrix X is:  (5, 1253)
[[ 96.424286  95.747147  96.610001 ... 159.979996 159.270004 159.860001]
 [ 95.747147  96.610001  97.205711 ... 159.270004 159.860001 161.470001]
 [ 96.610001  97.205711  94.677139 ... 159.860001 161.470001 162.910004]
 [ 97.205711  94.677139  94.370003 ... 161.470001 162.910004 163.350006]
 [ 94.677139  94.370003  95.684288 ... 162.910004 163.350006 164.        ]]

There are m = 1253 training examples
```

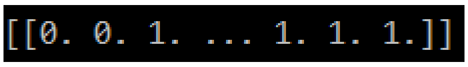
Figure 5-12: function checkpoint

As one will recall, logistic regression is a traditional model characterized by having a binary output, meaning that the algorithm is only capable of generating two types of outputs, 0 or 1. As can be imagined, this greatly conditions the type of prediction to be made (now it is no longer possible to predict the exact output value), therefore it will be necessary to modify the values of the Y matrix in order to proceed with the execution of the model.

The decision taken to solve the problem is that if the original value of the Y matrix is greater than the average of the associated values of the X matrix, then this will be replaced by a 1, otherwise a 0 will be recorded. Looking back at Figure 5-10, the first training example should be transformed to:

$$\frac{96.424 + 95.747 + 96.610 + 97.206 + 94.677}{5} = 96.133$$

$$96.133 > 94.370 \xrightarrow{\qquad} y = 0 \tag{5.1}$$

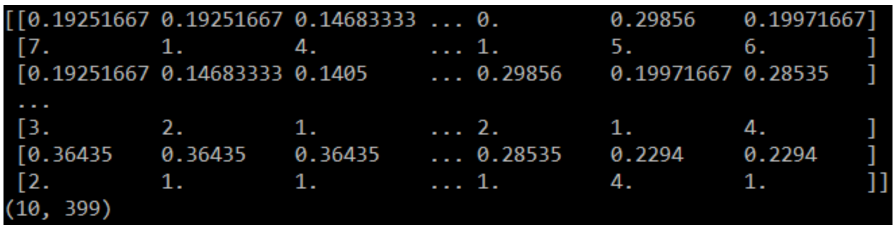


```
[[0. 0. 1. ... 1. 1. 1.]]
```

Figure 5-13: new values of Y

As it can be seen, the function works appropriately.

In the case of an analysis with sentiment data (and when considering both the historical and sentiment data), the input matrix will be slightly different, in this case for each date two values (later it will be three) will have to be provided, the average sentiment value of all posts of that day and the number of posts of that day. As a consequence, the dimensions of the matrix are modified as follows:

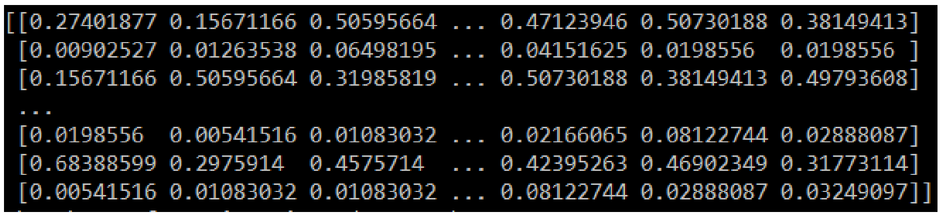


```
[[0.19251667 0.19251667 0.14683333 ... 0.         0.29856    0.19971667]
 [7.         1.         4.         ... 1.         5.         6.        ]
 [0.19251667 0.14683333 0.1405     ... 0.29856    0.19971667 0.28535   ]
 ...
 [3.         2.         1.         ... 2.         1.         4.        ]
 [0.36435    0.36435    0.36435    ... 0.28535    0.2294     0.2294    ]
 [2.         1.         1.         ... 1.         4.         1.        ]]
(10, 399)
```

Figure 5-14: matrix X with sentiment ratings before being scaled

As it can be seen in Figure 5-14, if it is decided to use the previous five days for the forecasting, the size of each training sample (the number of columns) is twice this value. The great thing about these algorithms is that you do not have to explain the difference between the first and the second element of the sample, through the optimization of the parameters it is the algorithm itself that is decide which elements are more important and which one should be dismissed.

If one looks at the caption of Figure 5-14, it is said that the matrix is not scaled yet, this is very clear because after the scaling no value can be smaller than zero nor greater than one. The reason why it was decided to show the matrix before the scaling is because this way it is very clear to see how there is no relation between the two types of data inserted. After the scaling of the data, the matrix will look like this:



```
[[0.27401877 0.15671166 0.50595664 ... 0.47123946 0.50730188 0.38149413]
 [0.00902527 0.01263538 0.06498195 ... 0.04151625 0.0198556  0.0198556 ]
 [0.15671166 0.50595664 0.31985819 ... 0.50730188 0.38149413 0.49793608]
 ...
 [0.0198556  0.00541516 0.01083032 ... 0.02166065 0.08122744 0.02888087]
 [0.68388599 0.2975914  0.4575714  ... 0.42395263 0.46902349 0.31773114]
 [0.00541516 0.01083032 0.01083032 ... 0.08122744 0.02888087 0.03249097]]
```

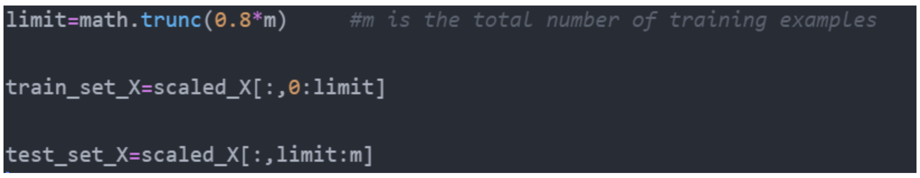
Figure 5-15: X matrix scaled

The equation used to scale the data between zero and one is the following:

$$\frac{x - \min(x)}{\max(x) - \min(x)} \tag{5.2}$$

Where *x* represents any vector of data. Later on, it will be seen that another way to scale the data is by using scalers from libraries available on Python. A scaler that is widely used and later on will be implemented in this project is the SKLEARN library, its scaler is called MinMaxScaler() and it is a function that requires as input the extreme values that can be available, in this case the values 0 and 1 will have to be imputed.

Finally, the last step to be performed before running the algorithm is the separation of the database so that a training set and a test set are available; this is normally divided into three groups (training, validation and test), but since the amount of data available is very low and only the training and test set are required to measure performance, the validation set will be omitted. The distribution of the data will be such that the training set will have 80% of the data and the test the remaining 20%. To separate the data, all that is necessary is the determination of a value to be used as the boundary of one matrix and the beginning of the next. Once the data have been separated, the model can be run.



```
limit=math.trunc(0.8*m)     #m is the total number of training examples

train_set_X=scaled_X[:,0:limit]

test_set_X=scaled_X[:,limit:m]
```

Figure 5-16: setting of the limit value

The model is the only thing that really changes with respect to the other algorithms, however, there is still a clear path that all of them follow; the first step is the creation of the parameter matrices (W and b), then the optimization processes of these will begin through the forward and backward propagation steps. Once the optimizing of the matrices is finished, there will be a process of prediction where the parameters obtained through the model will be put to the test. This last assessment will output a train accuracy and a test accuracy, and they are these values that will be used to decide what hyperparameters to modify (Hyperparameter tuning).

In the particular case of the logistic regression, it is important to remember that there are no hidden layers and therefore the only activation function that will be used is the sigmoid function. In addition, the cost function will be introduced manually, later on it will be seen that it can also be introduced through preestablished optimizers. The reason for deciding to code manually the cost function is because it makes easier for the reader to understand the process undergone by the data. When later on the complexity of the algorithms is increased, less attention will be placed on these elements and more will be placed on elements such as the process to enter hidden units or how the weight and bias matrices propagate through the model.

Once the models are ready and running, the impact of the data of different origins and the effects of the hyperparameter tuning will be analyzed, this analysis will be carried out in the chapter focused on the analysis of results. However, the methodology of this analysis will be explained now and it will consist on the following systematic process:

1. Selection of a type of data (stock values, sentiment ratings or both).
2. Analysis of the performance of the model.
3. Modification of the different hyperparameters (number of previous data to consider, leaning rate, number of epochs, etc.)
4. Return to step 1 and choose a different type of data.

5.4.2 ONE-LAYERED NEURAL NETWORK

The model employing a single neural network is a midpoint between the recently seen logistic regression algorithm and those known as Deep Neural Networks [37].

The main difference that will be present in this second model is that now the computation of the input matrix will be computed (through the hidden units) more than once. Previously, the data would interact with a single matrix and work was done to optimize the parameters of this matrix, the problem with this method was that if for some reason the matrix would fall into a local minimum, then the model would consider that the optimum had been reached and therefore the iterative process would not be able to get out of that minimum and the global one would not be found.

This situation does not occur in Neural Networks, the reason for this is that, by computing the data with several weight matrices (initialized with different values), the possibility of any unit finding the global minimum is significantly enhanced. On the other hand, this matrix also has its disadvantages; as it has more computational units, it increases the flexibility of the algorithm and with it the tendency to increase overfitting problems (this will be an important problem to be solved as efficiently as possible). In addition, this matrix is much more powerful than the one used in the previous case; therefore, the computational costs will also be higher and this may sometimes determine the choice between two models (if one takes a lot of time to compute the forecasting, maybe it is better to use another model).

Another important feature of this model that should be taken into account is that the number of hidden units is variable. This is the same as saying that the number of hidden units becomes a new hyperparameter which, like the others (those mentioned in the previous section), must be optimized to get the most out of this new model.

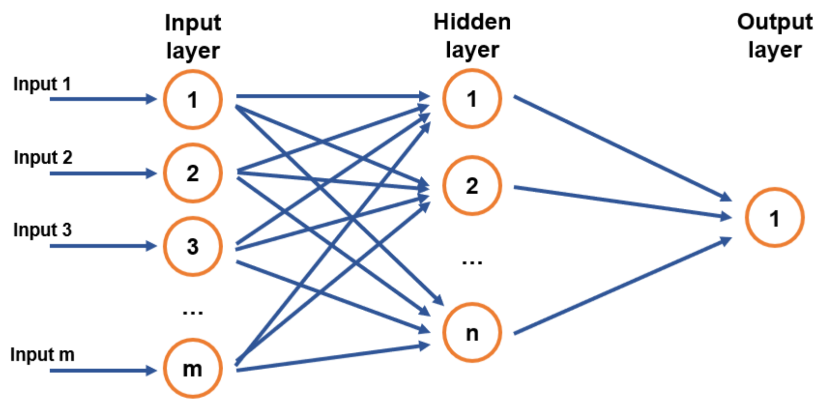


Figure 5-17: one-layered Neural Network diagram

As for the programming of the model, it does not vary much from that of the logistic regression. As before, only the sigmoid function will be used as the activation function (to avoid exploding gradients). The main difference is that now the model introduces one more mathematical dimension to the computation by adding the hidden units. Other than that, everything else is the same; forward and backward propagation steps, gradient descent, etc.

To facilitate the interpretation of the results, the analysis of the results of this section will be carried out following the methodology used in the previous section; first the reaction of the model to the historical data of the stock will be analyzed, then an optimization of the model will be carried out through an analysis of the impact of each of the hyperparameters and, once the best possible performance has been obtained, the same steps will be repeated using the sentiment ratings as a data source and lastly the same will be done but with the two data sources at the same time.

5.4.3 DEEP NEURAL NETWORK

In this project many concepts and ideas have been discussed and explained, proposals from different experts in the field have been commented and several models have been developed until arriving at this point; Deep Neural Networks. This type of algorithms are pioneers in the world of predictive models and now they will be put to the test.

As mentioned before, the main difference between Neural Networks and other models is that they have more than one hidden layer. The difference between the single-layer Neural Networks and the deep ones is that the latter also have (redundancy aside) depth. What this means is that, while in the other models what is computed in the first layer (regardless of the number of units in it) is what the output unit receives, in the Deep Neural Networks the output of this first layer is simply the input in another new layer with X units ready to compute again the conclusions of the previous level.

Logically, the process of computing the conclusions already generated by another layer allows the model to reach a much higher level of detail and accuracy than any other model. In particular, this type of networks is perfect for the detection of behavioral patterns. For this reason, it is expected that this model will be the one that best interprets and recognizes the relationship between different types of data.

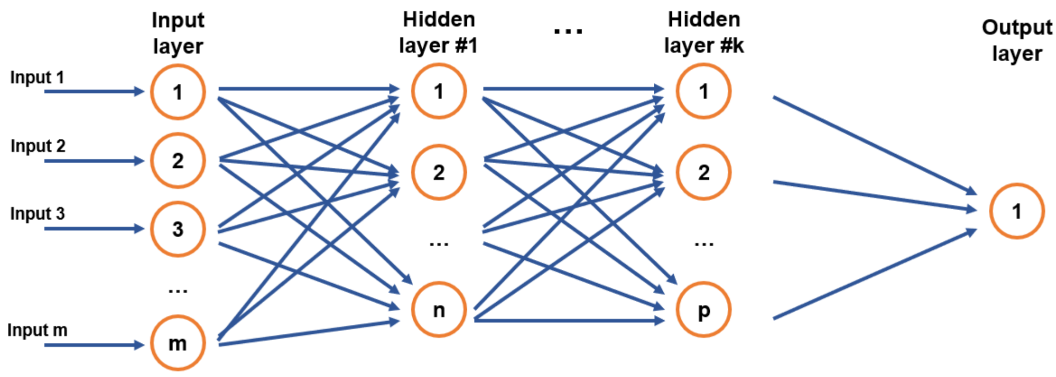


Figure 5-18: Deep Neural Network diagram

From the point of view of the programming of the model, in this algorithm there will be a greater number of changes with respect to the algorithms seen so far. The reason for so many modifications is that since there is more than one depth layer, it is necessary to use an activation function that does not reduce the values too much in order to avoid vanishing gradients. The decision made is to use the ReLU function; consequently, it will be necessary to code some functions that allow to propagate the information forward and backward effectively (in the previous model it was not necessary to do this because in both cases there was only the sigmoid function). Aside from this, the process followed will be the same as before; the data will be imputed, it will be propagated forward, the prediction will be compared to the expected out put and then the backpropagation and the optimization of the parameters will be carried out.

Nevertheless, a new hyperparameter will be analyzed, this hyperparameter will be the number of depth layers and, like the others, it will have to be modified to get the best possible prediction.

Finally, it is important to note that, although this model is much more complex than the previous ones, this does not necessarily mean that it will generate better results, the reason for this is that as the amount of data available is very small compared to what is usually computed in these models (millions of data), it is possible that the model is not able to generalize well across the information and therefore output a prediction that is worse than the ones generated by models like the Logistic Regression.

Before starting the next chapter, it is important to know that all the analysis of data will be carried out using the information of the company Facebook Inc. The reason for this choice is because it is one of the most famous companies worldwide and this results in a higher volume of tweets debating their situation.

In addition, during the timeframe studied began some of the most important accusations [38] and this generated a lot of turmoil in the share price. It is intended to take advantage of this situation (the more unstable the share, the more complex the prediction and the greater the challenge for the model).

Chapter 6. ANALYSIS OF RESULTS

In this chapter, the results obtained throughout the project will be explained and analyzed. The intention of this section is not only to present yield values but also to show the reasoning associated with them and, based on the different results, to see what measures are used to improve these performances.

As in the previous chapter, the process for the analysis of the results will start with the analysis of the least complex algorithm and, once all the different components of the model have been analyzed, more elaborate models will be discussed. For each different type of model the same methodology will also be followed, first analyzing the model using only historical stock values, then analyzing how it responds to sentiment scores and finally working with both types of data simultaneously.

In addition, in the more complex models it will also be analyzed how the models respond to inputs from competing companies. The reason for only doing this with the more complex models is because they have more computational power and this allows them to better interpret the information, in the simpler models the input of this data will hinder the prediction.

6.1 LOGISTIC REGRESSION

As has already been indicated numerous times, logistic regression is a model that makes predictions using a relatively simple model. Even so, the fact that it is not very complex does not mean that it is worse at predicting. As explained at the beginning of this work, it is very common for algorithms of lower complexity to generate better results in small databases. Having said this, the analysis of the results may begin.

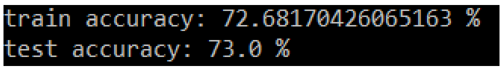
6.1.1 ANALYSIS OF STOCK VALUES

The first analysis to be carried out will consist on studying the performance of the logistic regression model when only data related to the historical value of the stock is entered.

The first thing to do will be to run the model and see the first results, it is important to remember that throughout the analysis the changes in both the training and the test set will always be analyzed, as these are the ones that will allow to see if there is an overfitting or underfitting problem.

Another thing to keep in mind when analyzing the results is that the hyperparameters can be determined in two ways; on the one hand, one can resort to the conclusions of other researchers and use the values that they have defined as optimal. On the other hand, one can also determine them through a process of trial-and-error. If it is decided to resort to a trial-and-error process, it is necessary to bear in mind that this is not a random selection of parameters, but a process in which the values are chosen using criteria based on theoretical grounds.

The first results had been obtained using the closing price data and for each sample it was decided to take into account the five preceding days. In addition, the learning rate was stablished at 0.05 and the number of iterations was set at 10,000. To this setting, the performance values were the following:



```
train accuracy: 72.68170426065163 %
test accuracy: 73.0 %
```

Figure 6-1: first computation (1)

As can be seen, the results obtained are not very good since in both cases the efficiency obtained is somewhat low. On the other hand, the difference between the two indicators is quite low, which means that there is hardly any problem of overfitting (which is usually the most frequent setbacks in these cases) and, therefore, the problem to be focused on is that of underfitting.

As already explained in chapter 2.1.3, the three most recommended ways to solve this problem are:

- Train the algorithm longer: do more iterations, change the learning rate.
- Augment the data size: increase the number of previous days to take into account.
- Try a bigger network (this does not apply to the case of the Logistic Regressions)

Taking these recommendations into consideration, the first change to be made is the number of previous data to be considered. After several experiments with different values, it can be clearly observed that, as the number of previous days to be considered increases, the performance values increase to the point that there starts to be an inevitable overfitting.

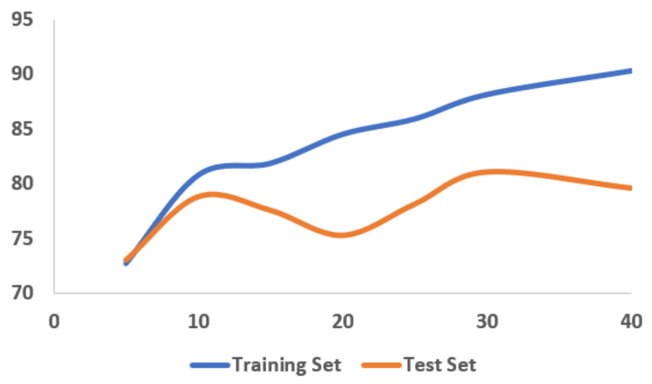
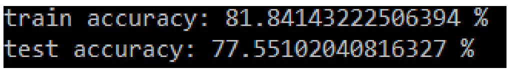


Figure 6-2: performances given considering different amounts of previous data

Taking this into account, the decision made was to take an action very similar to that of Early Stopping, so it was decided to choose a value that would provide a training set performance higher than 80%, but avoiding that the difference between the two indicators exceeded 5%, thus avoiding a problem of overfitting that would be difficult to solve. Consequently, the use of the previous <u>fifteen days</u> fulfills this requirement.

By doing so the new performances are the following:



```
train accuracy: 81.84143222506394 %
test accuracy: 77.55102040816327 %
```

Figure 6-3: performance after tuning of first hyperparameter

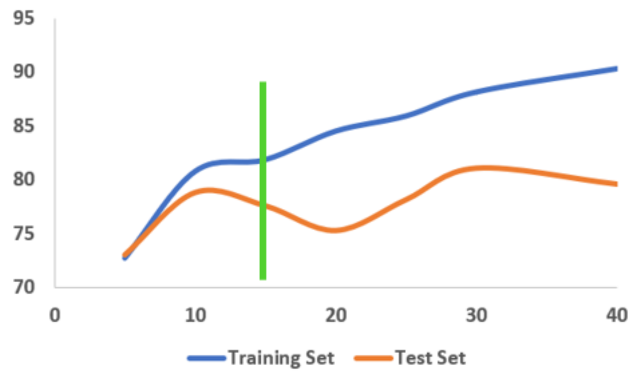
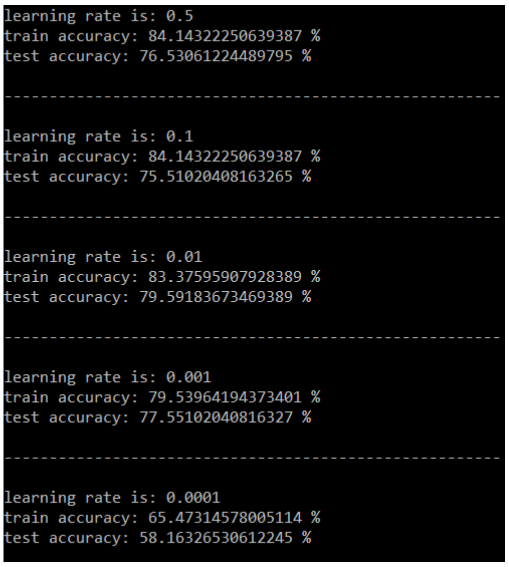


Figure 6-4: Early stopping in the forecasting

The following hyperparameter that will be optimized will be the number of iterations. These hyperparameter indicates the number of times the algorithm will go through the same data. In this case, a very high number of iterations will never have a negative effect on the performance of the training set, however, it will increase significantly the computational cost (which means more running) and there is a risk that the model memorizes the inputs and then a lower performance in the test set will be outputted. The challenge is to find an equilibrium between performance efficiency in both datasets as well as time efficiency.

What can be observed in this analysis is that, as the number of iterations increases, there is a first part (20,000 iterations) in which the little existing overfitting is reduced to minimum values and it is possible to have both indexes above eighty percent efficiency (81.8414% for the training set and 80.6122% for the test set). From this point on, the overfitting increases again and the algorithm starts to be somewhat slow without achieving significant yields.

Finally, the last element that remains to be optimized is the learning rate, which consists of the step size at each iteration while moving towards a minimum of a loss function. As will be seen below, a slow learning rate together with an elevate number of iterations will be ideal in order to output a have a good performance, however, if the learning rate is too small, then the performance will plunge.



```
learning rate is: 0.5
train accuracy: 84.14322250639387 %
test accuracy: 76.53061224489795 %

-------------------------------------------------------

learning rate is: 0.1
train accuracy: 84.14322250639387 %
test accuracy: 75.51020408163265 %

-------------------------------------------------------

learning rate is: 0.01
train accuracy: 83.37595907928389 %
test accuracy: 79.59183673469389 %

-------------------------------------------------------

learning rate is: 0.001
train accuracy: 79.53964194373401 %
test accuracy: 77.55102040816327 %

-------------------------------------------------------

learning rate is: 0.0001
train accuracy: 65.47314578005114 %
test accuracy: 58.16326530612245 %
```

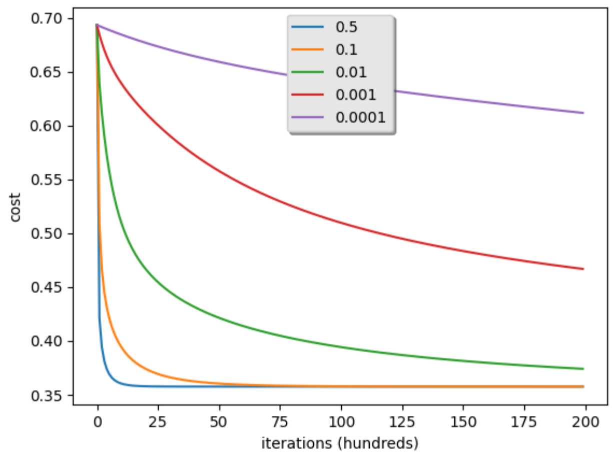
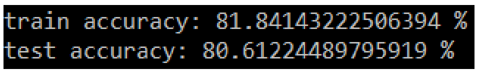


Figure 6-5: performances regarding different learning rates

As it can be seen in Figure 6-5, it does not appear that there are any cases in which the percentages obtained in the analysis of the iterations are exceeded, in which case a value of 0.005 was used. The reason why this value was initially chosen is because it is the one recommended by the Deep Learning expert Andrew Ng [39]. Therefore, the final performance obtained is the following:



```
train accuracy: 81.84143222506394 %
test accuracy: 80.61224489795919 %
```

Figure 6-6: final performance

In comparison to the initial values obtained, hyperparameter tuning has enabled to optimize the efficiency of the forecasting by almost 10%.

This process was then repeated with all the different types of price value (opening, closing, high, low and volume exchanged), however, none of the other cases exceeded the percentages shown above.

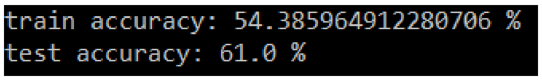
6.1.2 ANALYSIS OF SENTIMENT RATINGS

Next, the same process will be carried out, the main difference now is that the input matrix has different dimensions as there are two different types of data inputted this time; sentiment ratings (average of all the posts of each day) and quantity of posts per day.

Since the amount of data is relatively low and the current model is not too complex, there is a possibility that the initial performance of this second model will be worse than that of the previous section. The reason for this is because if the algorithm does not realize that there are different types of data, then it may compute them homogeneously and therefore loose the advantage of the extra information provided by this model (the less complex the algorithm is, the more chances there are of this happening).

Before beginning it is important to remember that in this case the hyperparameter used previously to choose which type of price to use (opening, closing, etc.) no longer exists due to a problem encountered during the data cleansing (for further insight read chapter 5.3). Therefore, the only type of price that can be used is the closing price.

Following the steps in the previous section, it was decided to initially take into account the five preceding days. In addition, the learning rate was stablished at 0.05 and the number of iterations was set at 10,000. To this setting, the performance values were the following:



```
train accuracy: 54.385964912280706 %
test accuracy: 61.0 %
```

Figure 6-7: first computation (2)

As it can be seen in Figure 6-7, the performances now are significantly worse that the initial ones encountered previously. In this case there are both severe overfitting and underfitting problems. The fact that the initial performance is so low makes a lot of sense because the same hyperparameters are being used for a dataset that is much complex.

The problem of having to deal with significant overfitting and underfitting is that, because of the bias-variance tradeoff (explained in chapter 2.1.3) it is very difficult to reduce both elements simultaneously and therefore the best alternative would be to resort to more complex algorithms. However, the hyperparameter tuning process will still be carried out in order to see how much of the overfitting and underfitting problems can be reduced simultaneously.

The first element to optimize will again be the number of previous data to be considered. After several experiments, it can be seen that the situation is the same as in the previous case, but with the difference that this time the optimum point is reached with a higher value of reference days (30). This means that, if few previous days are taken into account, the algorithm is not able to understand well how the different elements contribute value to the prediction. However, as the number of reference days increases (and with it the size of the input matrix), the model starts to recognize a pattern and is able to generate better predictions.

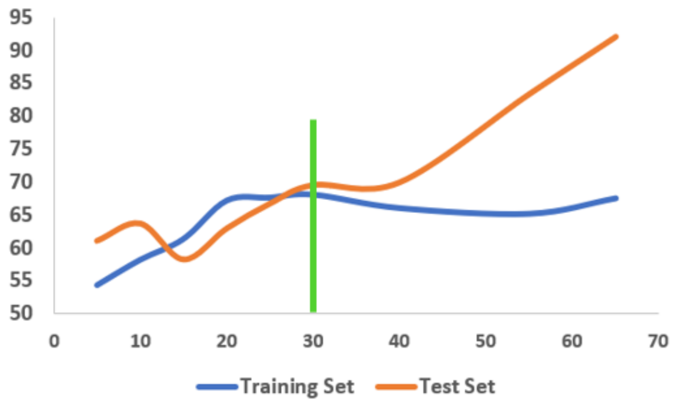
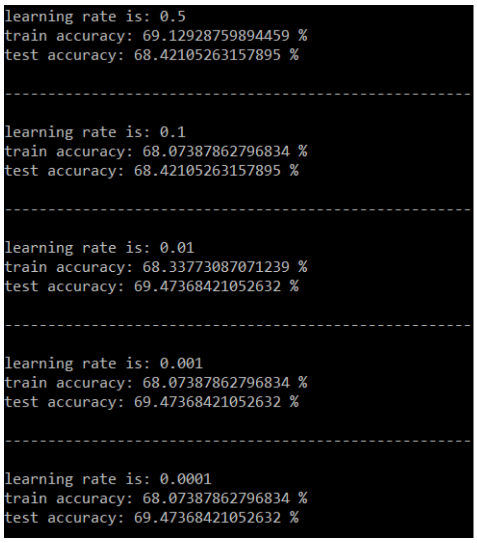


Figure 6-8: performances given considering different amounts of previous data (2)

As before, this does not mean that a larger input size will always give better performance, from a certain dimension onwards there is significant overfitting and therefore it has been decided that only the 30 days prior to the prediction will be taken into account, as this is very effective in reducing overfitting and the focus can be on underfitting.

Continuing with the number of iterations, in this case the effect of the modification of this hyperparameter is minimal. This means that a, regardless of the number of iterations to be performed, the indicators do not vary more than 0.2%. This can be interpreted to mean that the primal information is well learned in few iterations, therefore, increasing the number of times the model goes through the data will change very little because the cost function can hardly be further optimized.

As for the learning rate, the situation changes with respect to the previous section, the main difference is that now the use of one learning rate or another does not significantly affect the performance indexes. In the previous scenario, the use a learning rate that was too low meant that the algorithm was not able to predict effectively, but now it is able to do so, because the data (given enough reference days) is easier to interpret and therefore the prediction efficiency is consistent for the different values. In this situation, the selection of one ratio or another cannot be done by looking at the performance percentages, it must be done by looking at the cost graph and the hyperparameter that allows to obtain a lower cost must be chosen.



```
learning rate is: 0.5
train accuracy: 69.12928759894459 %
test accuracy: 68.42105263157895 %

--------------------------------------------------

learning rate is: 0.1
train accuracy: 68.07387862796834 %
test accuracy: 68.42105263157895 %

--------------------------------------------------

learning rate is: 0.01
train accuracy: 68.33773087071239 %
test accuracy: 69.47368421052632 %

--------------------------------------------------

learning rate is: 0.001
train accuracy: 68.07387862796834 %
test accuracy: 69.47368421052632 %

--------------------------------------------------

learning rate is: 0.0001
train accuracy: 68.07387862796834 %
test accuracy: 69.47368421052632 %
```

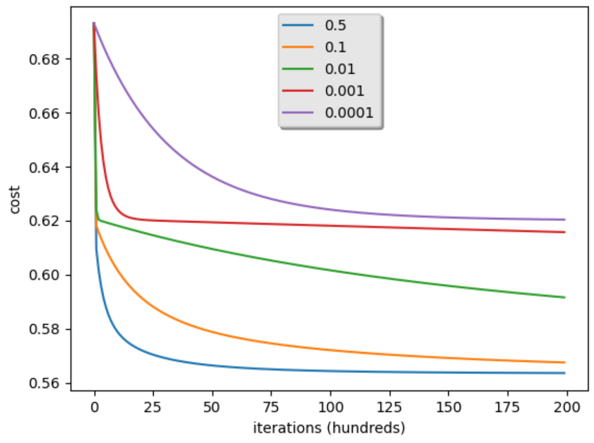
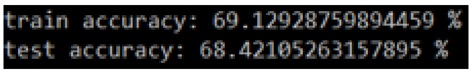


Figure 6-10: performances regarding different learning rates (2)

As a conclusion to this second analysis, it can be said that, with more elaborate data, priority should not be given to the number of iterations or the value of the learning rate. Efforts should be focused on providing sufficient amount of information for each training sample. All in all, the final performance will be:



```
train accuracy: 69.12928759894459 %
test accuracy: 68.42105263157895 %
```

Figure 6-9: final performance (2)

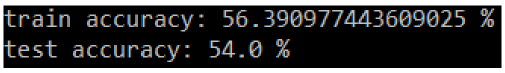
In this case, the performance indicators are improved significantly with regards to the initial values (around 15%), moreover, it has also been possible to efficiently reduce the overfitting which in this particular case had become a major setback. Be that as it may it is also interesting to see that the use of more elaborate data in this model did not bring with it an increase in efficiency; in the more complex models it will be seen that this no longer happens.

6.1.3 ANALYSIS OF HISTORIC STOCK AND SENTIMENT RATINGS

Once the influence of the different types of data on the model has been studied separately, it is time to put the two different types together in the same input matrix and see how the algorithm responds.

This is a situation similar to that of the previous section because a matrix containing more complex information is being introduced. In the case that the information is interpreted adequately, the performance of the model will be significantly better compared to the yields seen so far, otherwise, the information will cause the performance to drop and the best solution will be to use a more powerful model at the computational level.

The initial computation, using the closing price (remember that now it is the only one that can be used), the five days prior to the prediction, 10,000 iterations and a learning rate of 0.005 will output the following result performance values:



```
train accuracy: 56.390977443609025 %
test accuracy: 54.0 %
```

Figure 6-11: first computation (3)

As in the case where sentiment data were used, the first computation shows very poor performances. Similarly, this is due to the fact that high data complexity cannot be adequately be interpreted with few reference values, so it is expected that by increasing this hyperparameter the model will be enhanced. On the other hand, another element that is striking is that, although the performance values are low, a low overfitting problem is also maintained, however, it is important not to jump to conclusions with such complex data as is it possible that this low overfitting is a consequence of an initial instability due to the fluctuation of the performance indicators.

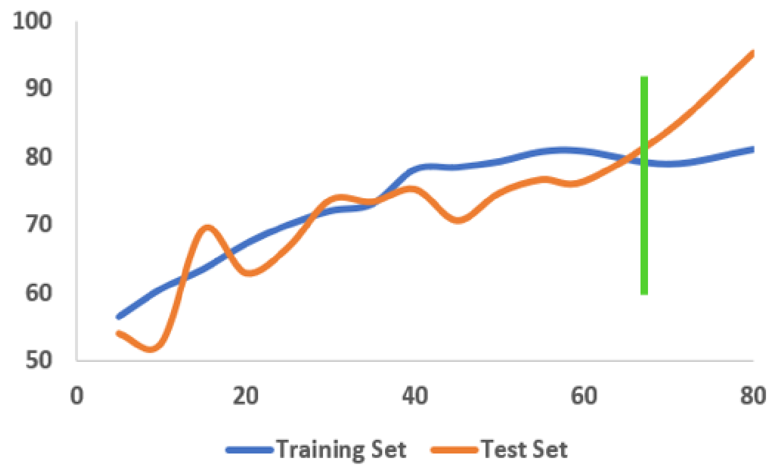
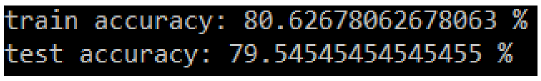


Figure 6-13: performances given considering different amounts of previous data (3)

After analyzing the performance of the model for different amounts of reference data, it is concluded what was commented in the previous paragraph; with few reference values the yields are not constant and, although there is not an abysmal difference between them, they are fluctuating in an unstable way until a significant amount of them is used (about 50 days of reference). Therefore, it is decided that the value that guarantees good performances and, at the same time, little overfitting as well as stability is the use of 65 previous days. The performance indicators for this hyperparameter are:



```
train accuracy: 80.62678062678063 %
test accuracy: 79.54545454545455 %
```

Figure 6-12: performance values imputing the data from 65 days prior the forecasting

Comparing these values with those of the first computation, the performance of the model is improved by almost 25%.

Continuing with the systematic process, the next hyperparameter to deal with is the number of iterations.

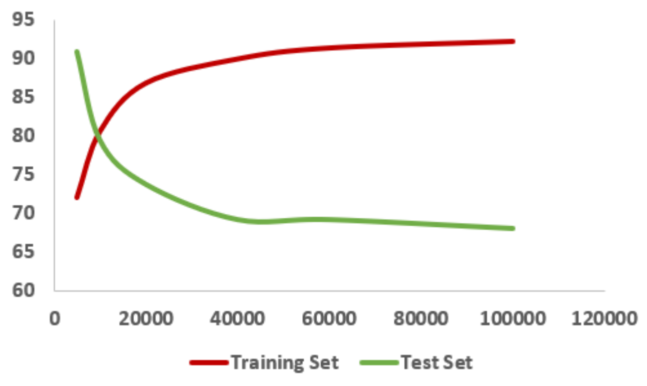
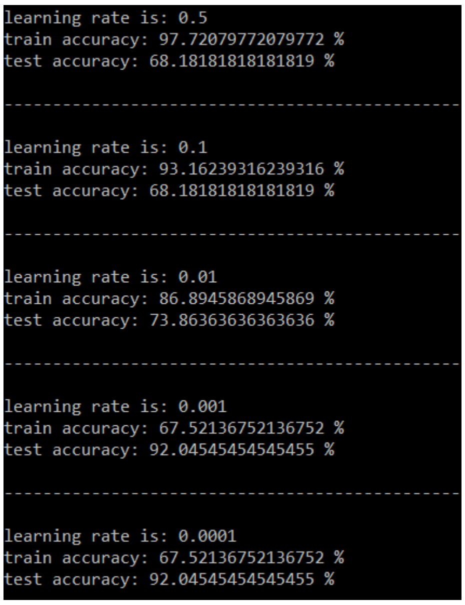


Figure 6-14: performance for different number of iterations

Through Figure 6-14 it can be clearly seen how an increase in the number of iterations only results in an increase in the difference between the two performances (the test set reaches very quickly very low efficiencies). This makes sense because it means that, after a certain number of iterations, the algorithm starts to memorize the parameters and stops learning. Looking at the graph it has been decided that the number of iterations to use will be the one preestablished (10,000), as it is the one that delivers the minimum overfitting as well as a good performance (around 80%).

Finally, analyzing the learning rate it turns out that, although a ratio of 0.5 reduces the cost function adequately, the overfitting problems are attenuated resulting in very low performances. This problem occurs for all the different values studied, with the exception of the value 0.005 (recommended by Andrew Ng) where the overfitting remains low (approximately 1%). It has therefore been decided to keep the default value and therefore to maintain the yields indicated in Figure 6-12.



```
learning rate is: 0.5
train accuracy: 97.72079772079772 %
test accuracy: 68.18181818181819 %

-------------------------------------------------------

learning rate is: 0.1
train accuracy: 93.16239316239316 %
test accuracy: 68.18181818181819 %

-------------------------------------------------------

learning rate is: 0.01
train accuracy: 86.8945868945869 %
test accuracy: 73.86363636363636 %

-------------------------------------------------------

learning rate is: 0.001
train accuracy: 67.52136752136752 %
test accuracy: 92.04545454545455 %

-------------------------------------------------------

learning rate is: 0.0001
train accuracy: 67.52136752136752 %
test accuracy: 92.04545454545455 %
```

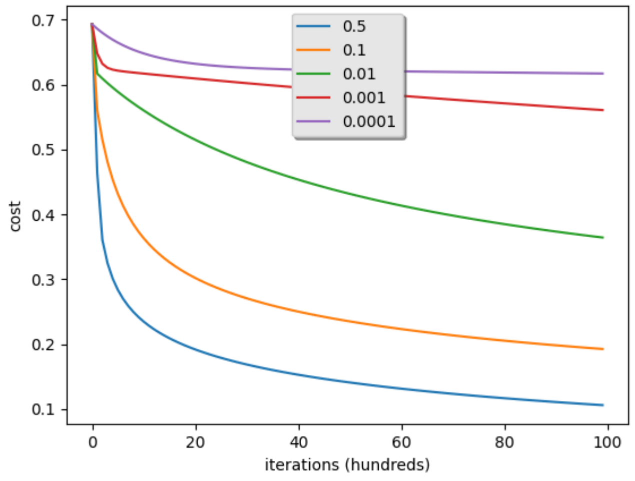


Figure 6-15: performances regarding different learning rates (3)

As indicated above, the final yields are the ones shown in Figure 6-12. When compared to the initial outputs, a significant improvement can be observed (approximately of 25%). It is interesting how by modifying the hyperparameters one is able to improve the predictive capability of the model so strikingly.

6.2 ONE-LAYERED NEURAL NETWORK

Although explained earlier in the methodology chapter, the process to be followed to analyze this model will be exactly the same as the one adopted for logistic regression. It should also be noted that in this case there is a new hyperparameter whose impact is sure to attract attention; the number of hidden units. The main difference in the analysis of this model is that now, instead of using the training and test accuracies, the cost function will be a very important indicator that be more taken into account study the efficiency of the model, the reason for this decision is none other than the intention to demonstrate how the cost function can also be used to optimize a model efficiently.

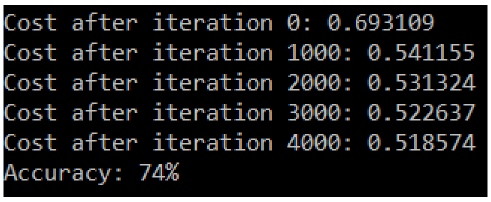
In addition, these models will have a predetermined learning rate, the reason for this is because numerous papers [40]–[42] recommend to prefix a value of 1.2. For this reason, it is not necessary to optimize this setting as there is a high number of experts who endorse the use of this particular value. Having said this, the analysis of the results may begin.

6.2.1 ANALYSIS OF HISTORIC STOCK VALUES

The first analysis to be carried out will consist on studying the performance of the monolayer Neural Network where only data related to the historical value of the stock is entered.

The first thing to do will be to run the model and see the first results, it is important to remember that throughout the analysis the changes in the evolution of the cost function will be analyzed, the reason to do so is because this is an element that can indicate how well the algorithm is being optimized. However, other indicators will also be considered since the use of only the cost function can sometimes be misleading.

The initial computation will use the closing price as the type to stock to use, it will consider the five days prior to the prediction, will use 2 hidden units, a learning rate of 1.2 and 5,000 iterations. The performance delivered is the following:



```
Cost after iteration 0: 0.693109
Cost after iteration 1000: 0.541155
Cost after iteration 2000: 0.531324
Cost after iteration 3000: 0.522637
Cost after iteration 4000: 0.518574
Accuracy: 74%
```

Figure 6-16: first computation (4)

By looking at Figure 6-16, it can be seen how the cost function is hardly reduced with respect to the initial value, moreover, this is a rather bad result since acceptable cost function values should be below 0.2 [42]. Furthermore, the efficiency of the model is not particularly good. Looking at the output from another point of view one can conclude that the prediction has room for improvement.

After analyzing the impact of the number of previous days to be taken into account, it is concluded that the value of the cost function is constantly optimized as the value of the hyperparameter increases.

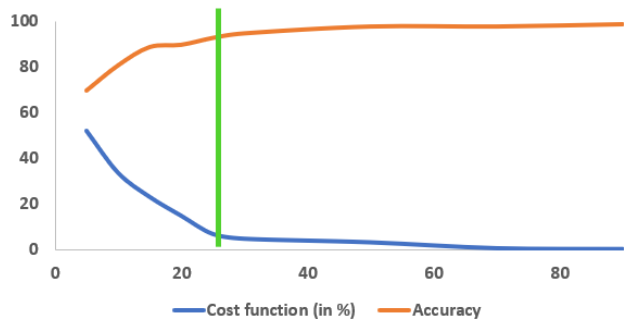


Figure 6-17: performances given considering different amounts of previous data (4)

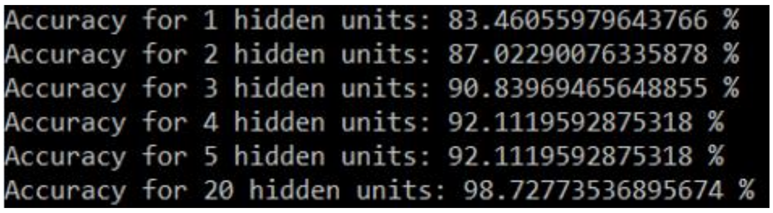
Although this sounds like very good news, it is necessary to be critical and analyze these results in detail. If, in addition to these indexes, those used in the previous chapter are used, it can be clearly observed that for high values of reference days the difference between the performance of the training and the test set is abysmal (more than 25%). This clearly means that there is a problem of overfitting that must be dealt with.

After reanalyzing the different parameters and taking into account all the indexes, it is concluded that the optimum value is the one given for 10 reference days. By doing so, both the training and the test set have values higher than 80% (85.489% and 82.291% respectively) and the overfitting problems are much lower than those obtained for a greater number of reference values.

When analyzing the optimal number of iterations, the same thing happens as before, from 40,000 iterations on, the cost function always has values lower than 0.01 and the efficiency of the training set does not go below 97%. If the critical sense is again used, one realizes that, with the database provided, it is not possible to achieve such good results (a database of millions of data would be necessary for such performances); the only logical explanation is that there must be a problem of overfitting. Indeed, for values of 40,000 iterations the performance of the training set is 78,24674%, which confirms the previous statement.

After analyzing the different results, it is concluded that the optimal number of iterations is to use 20,000 iterations. This results in an efficiency of 87.023%, the value of the cost function is 0.20019 and the difference between the test and the training set is smaller than 4% (83.78412%).

Finally, if the impact of the number of hidden units to be used is analyzed, it can be clearly observed (see Figure 6-18) how the use of more hidden units leads to better efficiency in the training set. This is something that makes a lot of sense since, as already introduced in the methodology section, the use of more units leads to the search for the optimal performance from a wider variety of approaches and this increases the possibility that some of them will find the global minimum of the cost function.



```
Accuracy for 1 hidden units: 83.46055979643766 %
Accuracy for 2 hidden units: 87.02290076335878 %
Accuracy for 3 hidden units: 90.83969465648855 %
Accuracy for 4 hidden units: 92.1119592875318 %
Accuracy for 5 hidden units: 92.1119592875318 %
Accuracy for 20 hidden units: 98.72773536895674 %
```

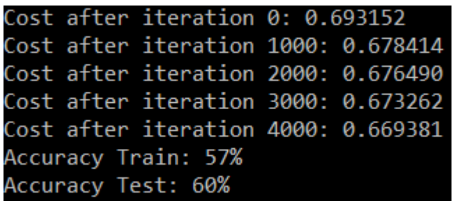
Figure 6-18: accuracy for different quantities of hidden units

Be that as it may, it is often recommended not to use too many hidden units because they often lead to overfitting problems, so that with the training data the model works very well (very high performance), but when new data enters (test set) the parameters are not able to generalize properly and the performance is very poor. For this reason and taking into account the size of the input matrix, the recommendable quantity of hidden units is between 2 and 5[40]. In the case of this model, it was decided to use 3 hidden units, this allowed to have really good performances for both cases (90.8397% in the training set and 88.2853% in the training set).

Comparing this final computation to that of the logistic regression for this data, one can see a significant improvement in the computations, the reason for such difference in the accuracy is because the Neural Network is one of the best existing models to work with data with a non-linear behavior (like the stock values).

6.2.2 ANALYSIS OF SENTIMENT RATINGS

As before, it is now necessary to repeat the same procedure but with a slightly more complex database. Taking into account the results of the previous section, it is expected that in this second study the results will be slightly better. The reason why a higher performance is expected here is because now one is dealing with more elaborated data and, being a more complex network, it is expected that the model will be able to interpret them properly so that the conclusions can be generalized more effectively than in the previous case (remember that this was not the case for the logistic regression where the use of only stock values resulted in better accuracies).



```
Cost after iteration 0: 0.693152
Cost after iteration 1000: 0.678414
Cost after iteration 2000: 0.676490
Cost after iteration 3000: 0.673262
Cost after iteration 4000: 0.669381
Accuracy Train: 57%
Accuracy Test: 60%
```

Figure 6-19: first computation (5)

In comparison to the first computation of the last case, it is interesting how this time the cost function is optimized even less than before, this is something that makes sense since the same hyperparameters have been used and now the data is more complex. Moreover, the performance percentages are very low. In this case, no conclusions will be drawn about overfitting because it is known that for the first computations these values are the result of huge fluctuations. For that reason, overfitting will be the subject of study in the hyperparameter optimization processes.

Considering what happened in the analysis of the historical stock values, for the sentiment scores it was decided to confront the cost function with the difference between the training performance and the test set. This allows to see for which value a balance is obtained between a minimum value of the cost function and a maximum similarity between the two performance indicators.

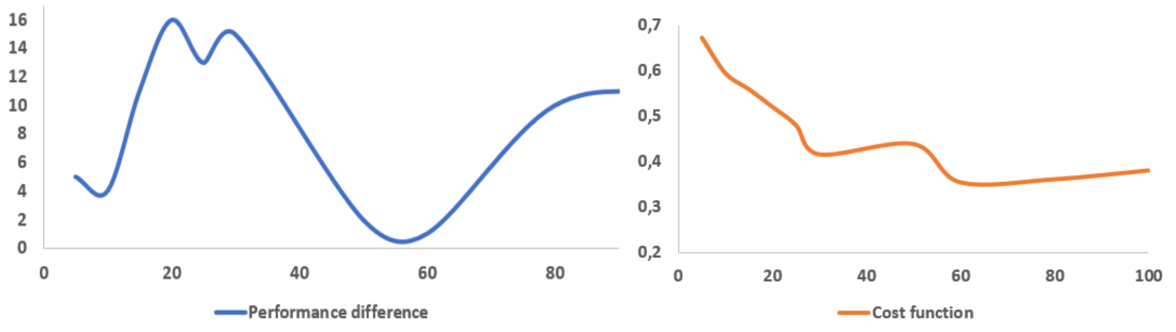
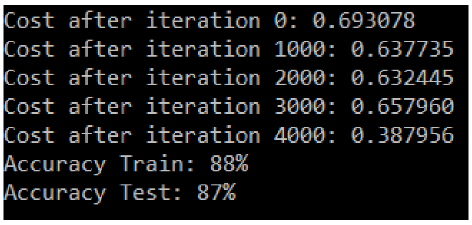


Figure 6-20: performance difference & cost function

The use of these two graphs is very useful because it allows to effectively visualize the behavior of the algorithm for different values of the hyperparameter. Looking at Figure 6-20 it can be seen very clearly that the algorithm is quite unstable in its predictions if few reference values are used. As more reference values are taken into account, the efficiency of the model is improved and both the cost function and the overfitting are optimized. Similarly to other situations, what happens is that for too many values the cost function stagnates and the overfitting skyrockets.

With all this, it is decided that the number of previous days to be taken into account will be 60 days and the performance of the model will be the following:



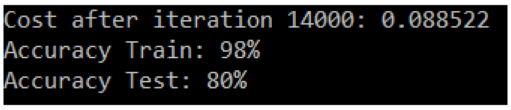
```
Cost after iteration 0: 0.693078
Cost after iteration 1000: 0.637735
Cost after iteration 2000: 0.632445
Cost after iteration 3000: 0.657960
Cost after iteration 4000: 0.387956
Accuracy Train: 88%
Accuracy Test: 87%
```

Figure 6-21: performance considering the 60 previous dates

It is interesting how, compared to the yields obtained when only the stock values were used, now it is necessary six times more data to output good accuracies. However, these performances are better than the ones obtained at the same point of the analysis for the last analysis.

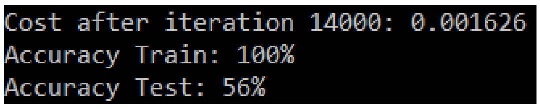
Analyzing now the response of the algorithm at different numbers of iterations, what happens is that the model is not able to significantly reduce the cost function without increasing overfitting. For example, if 60,000 iterations are used the model is able to optimize the cost function down to 0.1776, however, the training/test performance ratio would be 96/84 (which is not very good). If instead of 60,000, 15,000 iterations were used, the cost function would be reduced to 0.262654 and the ratio would now be 88/87. These last values are significantly better because, although the cost function can be improved, this is the minimum value that avoids overfitting problems.

After analyzing the effect of different numbers of units in the model, the same conclusions are reached as in the previous section; the input data is not complex enough to use more than three hidden units (in this case the best performance was obtained for two hidden units). To corroborate this conclusion, the effects of using 5 and 8 units will be shown below.



```
Cost after iteration 14000: 0.088522
Accuracy Train: 98%
Accuracy Test: 80%
```

Figure 6-23: performance of the model with 5 hidden units



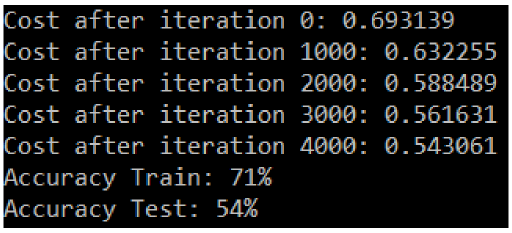
```
Cost after iteration 14000: 0.001626
Accuracy Train: 100%
Accuracy Test: 56%
```

Figure 6-22: performance of the model with 8 hidden units

The model memorizes the database and optimizes the cost function down to minimum values. As can also be observed, the higher the number of hidden units, the worse the generalization capability of the model.

6.2.3 ANALYSIS OF HISTORIC STOCK AND SENTIMENT RATINGS

In this last study of this model, the complexity of the data will be increased again so that now the model will have to work with three data inputs for each reference day; the value of the stock, the sentiment scores and the number of publications that mentioned the company on that day. The aim of this section is to see how the model reacts to this increase in complexity.



```
Cost after iteration 0: 0.693139
Cost after iteration 1000: 0.632255
Cost after iteration 2000: 0.588489
Cost after iteration 3000: 0.561631
Cost after iteration 4000: 0.543061
Accuracy Train: 71%
Accuracy Test: 54%
```

Figure 6-24: first computation (6)

The main difference of this first computation with respect to the other initial two, is that in this case the optimization is constant. What is meant by this is that in the first case the optimization of the cost function takes an initial step and thereafter the optimization is much slower; in the second case the optimization is slow from the beginning. Unlike these, the first computation now optimizes steadily and at a much higher pace than the others.

Following the established methodology, the impact of the number of reference parameters to be used will now be analyzed. Since in the previous section it has been very useful, the training and test set performance values and the cost function will continue to be used as indicators of the model's performance.

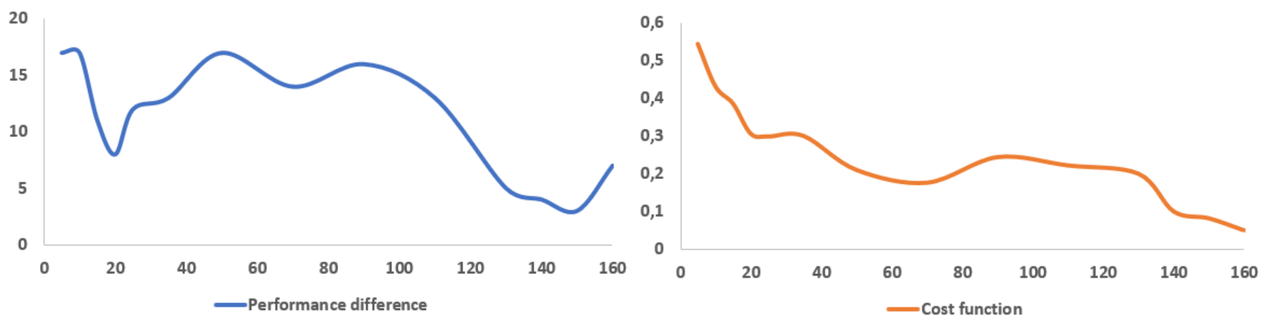


Figure 6-25: performance difference & cost function (2)

In this case the situation is quite similar to when only sentiment scores were used. In this case it can be seen how both indicators show for few reference values quite unstable behaviors characterized by high values of overfitting. However, as the number of reference values increases, one can see how the overfitting is progressively reduced as well as the cost function. As before, after a certain value, the function representing the yield difference starts to increase (overfitting increases again) while the cost function continues to be minimized. It is just before this point, at 140, that it is decided to stop and establish the number of reference values to be used.

By using 140 reference values the cost function has a value of 0.11026 and the performances of the training and test sets are 92% and 88% respectively.

Through the analysis of the effect of the number of iterations to which the model is exposed, it can be observed how (ratifying the conclusions of the previous sections) by rapidly increasing the number of iterations increases, the overfitting skyrockets (for 35,000 iterations the difference between the accuracies is almost 20%). Nevertheless, small increases in the number of iterations allow a slight improvement in the behavior of the model. For example, by using 12,000 iterations one is able to achieve training and test performances of 92% and 90% respectively and a value of 0.100048 for the cost function.

The latter result is significantly good, since the value of the cost function is very close to zero and there are almost no overfitting problems. However, just out of curiosity, it will now be analyzed how the model changes if, for the hyperparameters just determined, the number of hidden units to be used is modified.

Just like in the previous cases, increasing the number of computational units leads (for the same hyperparameters) to more overfitting; for five hidden units the performances are 95% for the training set, 82% for the test set and a cost function of 0.06184.

Although explained previously, the explanation for these findings is that a greater use of units leads to a more detailed analysis of the input matrix. If this increase in the detail of the analysis is performed on data that lacks a sufficient level of complexity, the units end up learning the values of the input matrix by heart and this leads to the model losing all its effectiveness resulting in a significant number of erroneous predictions.

All in all, the best performance is obtained for 140 days, for 12,000 iterations, a (prefixed) learning rate of 1.2 and two hidden units. The ratings obtained are 92% for the training set, 90% for the test set and a cost function of 0.100048.

6.3 DEEP NEURAL NETWORKS

Finally, the main course of this project is reached; the Deep Neural Networks. The reason for this (explained in detail in chapter 2.2) is that here the predictions created by the different neurons are the input for new layers of neurons (instead of being the output of the model). This fact allows, once again, to increase the accuracy of the analysis which in turn will allow to increase the efficiency of the predictions.

Although this is a very appealing idea, while working with single-layer neural networks, it has already been seen that there are scenarios in which the complexity of the model far exceeds the complexity of the data and this leads to significant overfitting problems.

For this reason, the need arises to increase the complexity of the data (the idea is that, through greater sophistication of the data, greater advantage can be taken of the model). To this end, the same analysis that has been carried out will be performed, with the difference that now, for each input, both the company's data and the data of rival companies in the market will be taken into account.

For the purposes of these analysis, the main company will continue to be Facebook Inc. and the following companies will be considered as market competitors:

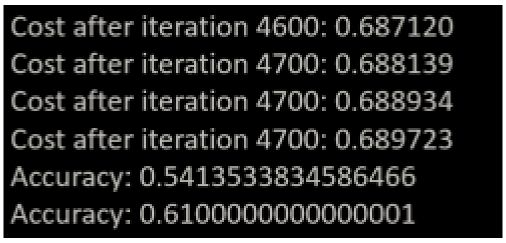
- Alphabet Inc. (GOOGL)
- AT&T Inc. (T)
- Intel Corporation (INTC)
- Cisco Systems Inc. (CSCO)
- Verizon Communications Inc. (VZ)

Finally, it is important to remember that in this final model it will be necessary to determine the number of layers to use and the number of hidden units to use in each one of them. This results in the appearance of one last hyperparameter that will also need to be optimized.

Having this said, the analysis of the model may begin.

6.3.1 ANALYSIS OF HISTORICAL STOCK VALUES

Just like with the other models, an initial computation will be created and this one will consist on a two-layered neural network where both layers will have two hidden units, it will consider the five days prior to the prediction, the model will compute 5,000 iterations and it will have a learning rate of 0.009 (value recommended by experts [43], and that is why it will be maintained through the whole analysis). The values obtained are the following:



```
Cost after iteration 4600: 0.687120
Cost after iteration 4700: 0.688139
Cost after iteration 4700: 0.688934
Cost after iteration 4700: 0.689723
Accuracy: 0.5413533834586466
Accuracy: 0.6100000000000001
```

Figure 6-26: first computation (7)

Analyzing Figure 6-26, it can be seen how the cost function is barely optimized for the initial values of the hyperparameters. Moreover, at this point, one ignores whether both the training and test accuracies are reliable or maybe just a consequence of a random initialization. Be that as it may, the optimization of hyperparameters will help understand how well the model interprets the data provided.

After analyzing the behavior of the model for different amounts of reference data, it is concluded that a large amount of prior information is necessary to be able to significantly optimize the cost function. As can be seen in Figure 6-27, using 80 days of prior data the model is not enough to reduce the value of the cost function below 0.5. It is only after 130 days of data that acceptable values of the cost function start to be obtained (without major overfitting problems). As expected, if the amount of reference data to be used continues to increase, what happens is that the cost function continues to minimize, but the model is unable to increase its efficiency in the test set (overfitting).

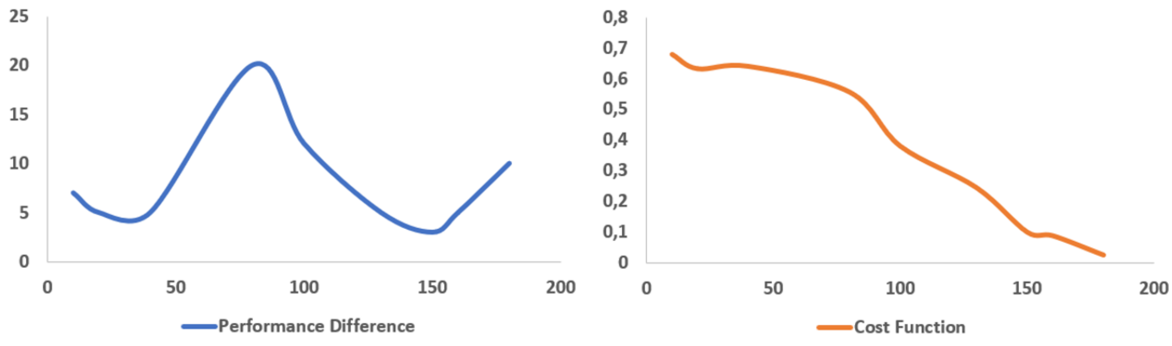


Figure 6-27: performance difference & cost function (3)

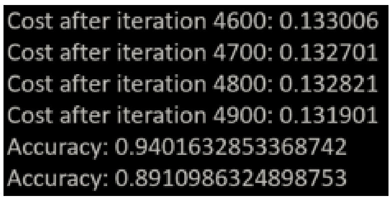
The fact that the model requires such big amounts of data is not a surprise. As it has been seen numerous times along the project, the increase in complexity of the data has always resulted in an increase in the demand of the quantity of data to compute for each training sample. Be that as it may, it is still necessary not to forget the overfitting, because with significant amounts of data it becomes the main threat to the performance.

Returning to the model, for 150 previous data the algorithm has a cost function of 0.135035 and the training and test accuracies are 92.1793% and 89.7536% respectively.

It is very interesting how, just by modifying this first hyperparameter, one is able to obtain outstanding performances; its impact on all the models has been notorious.

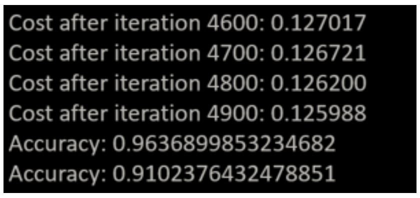
Regarding the number of iterations, it is surprising how small (compared to the first hyperparameter) its impact is on the behavior of the model. Looking at the evolution of the cost function for 5,000 iterations is enough to see that the pace of the optimization process is so small that the only consequence of this augmentation is the increase of the computational time, and this is something that is usually considered when the output is significantly enhanced, which is not the case. For this reason, the value of iterations to be used will be maintained at 5,000.

Since the learning rate is a preset value, it is now time to analyze the novelty of this model; the number of deep layers as well as the number of hidden units. The two hyperparameters must be studied hand in hand because for each hidden layer the number of units must be determined. Although the performance of the model will now be analyzed for a wide range of hyperparameter values, experts recommend that for small amounts of data, few layers should be used and not too many units in each layer.



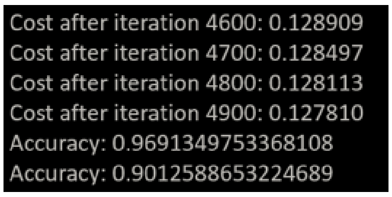
```
Cost after iteration 4600: 0.133006
Cost after iteration 4700: 0.132701
Cost after iteration 4800: 0.132821
Cost after iteration 4900: 0.131901
Accuracy: 0.9401632853368742
Accuracy: 0.8910986324898753
```

Figure 6-30: performance for three layers (3, 2, 4)



```
Cost after iteration 4600: 0.127017
Cost after iteration 4700: 0.126721
Cost after iteration 4800: 0.126200
Cost after iteration 4900: 0.125988
Accuracy: 0.9636899853234682
Accuracy: 0.9102376432478851
```

Figure 6-29: performance for four layers (2, 5, 2, 4)



```
Cost after iteration 4600: 0.128909
Cost after iteration 4700: 0.128497
Cost after iteration 4800: 0.128113
Cost after iteration 4900: 0.127810
Accuracy: 0.9691349753368108
Accuracy: 0.9012588653224689
```

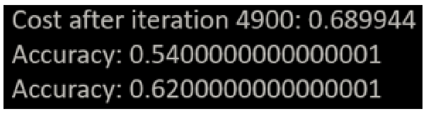
Figure 6-28: performance for five layers (4, 3, 2, 5, 2)

Surprisingly, the results obtained were far from what was expected. After the computation of the different models what happened was that none of them were able to surpass the accuracies of the initial case (for the three cases) regardless of the constitution of the model (see Figure 6-30, Figure 6-29 and Figure 6-28).

After further investigation of the results obtained it was concluded that what was happening was that, since the model was already very powerful with two layers of two hidden units, the use of a more complex network brought with it more overfitting and therefore less accuracy for inputs that the model had not seen during its training.

6.3.2 ANALYSIS OF SENTIMENT RATINGS

This time the initial computation returns a value of the cost function worse than the one generated in the previous cases (only sentiment ratings considered). This is logical because the function has gone from computing 2 values for each day of input to 12 (2 types of data x 6 companies). Even so, the difference between this cost function and the one obtained in Figure 6-19 is not so different. This is useful to realize the influence of the depth of the model in the information analysis.



```
Cost after iteration 4900: 0.689944
Accuracy: 0.5400000000000001
Accuracy: 0.6200000000000001
```

Figure 6-31: first computation (8)

Moreover, in this case there is a situation that is interesting to comment on (it is not the first time that it is encountered, however, previously it has not been discussed). If one looks at Figure 6-31, it can be seen that the training set performs worse than the test set, this is very rare because the model learns with the training set and is tested with the other one. Although it is logical that the model obtains better results with the training set, there may be cases in which the learned parameters allow better prediction of the test set.

This type of situation usually occurs when the model's performance is very poor; in these cases, the parameters have not yet been fully learned and, since they are elements that come from the same data distribution, it is possible that the parameters will predict accurately more cases than with the learning set. Later, when there are much better performances, it is more difficult for this to happen because the parameters have been able to better interpret the information available in the learning set.

Looking at Figure 6-33, it can now be seen that the optimization of the number of days to be used as a reference gives particularly good results. Although the shape of the graph is very similar to the ones seen in the previous sections, the main difference is that now at the optimum point the figures of the cost function are reduced to values very close to zero and, simultaneously, minimum overfitting is achieved. Even though the optimum value for the overfitting is reached using 140 previous days, if 155 are used, the overfitting is still very small and the cost function has an even smaller value. The performance of the model is shown in Figure 6-32.

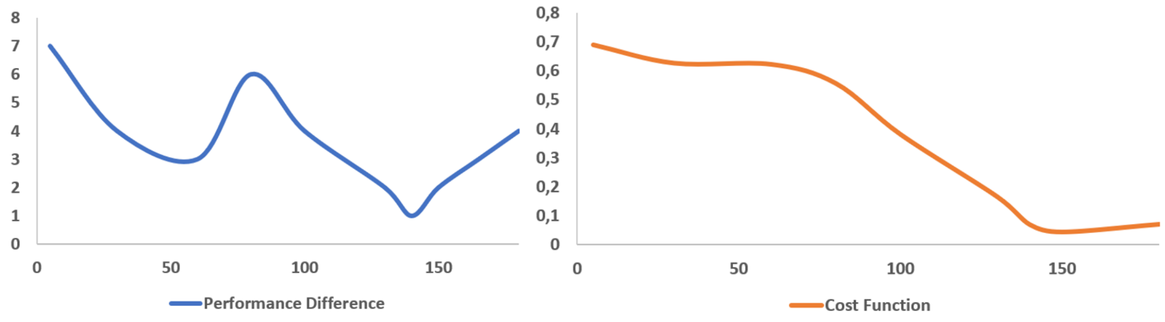
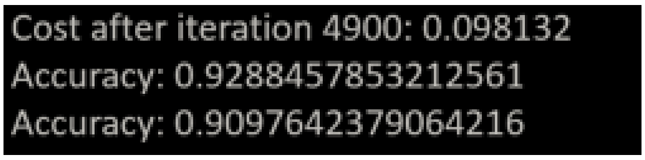


Figure 6-33: performance difference & cost function (4)

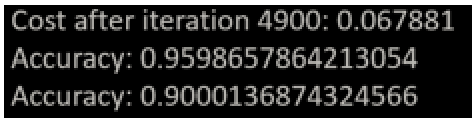


```
Cost after iteration 4900: 0.098132
Accuracy: 0.9288457853212561
Accuracy: 0.9097642379064216
```

Figure 6-32: performance for 155 previous days

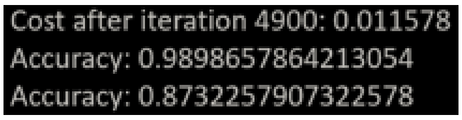
As before, it is observed that increasing the number of iterations leads nowhere. It can be seen that for the first iterations the reduction of the cost function is very fast, but from iteration number 1,000 onwards it slows down quite a lot and this slowdown increases to the point that between iterations 4,800 and 4,900 the value of the cost function is reduced by less than 0.0003 units. For this reason, the number of iterations to use will continue to be 5,000 as increasing this value will increase the computation costs and the improvement will not be significant.

As for the number of layers to be used, in this case neither the same yields nor the same cost function is obtained, however, none of the values are able to improve the ones obtained for the base case (two layers with two hidden units each).



```
Cost after iteration 4900: 0.067881
Accuracy: 0.9598657864213054
Accuracy: 0.9000136874324566
```

Figure 6-34: performance for three layers (2, 5, 3)



```
Cost after iteration 4900: 0.011578
Accuracy: 0.9898657864213054
Accuracy: 0.8732257907322578
```

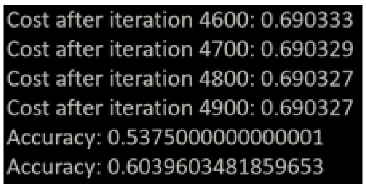
Figure 6-35: performance for six layers (2, 7, 5, 3, 4,2)

6.3.3 ANALYSIS OF HISTORIC STOCK AND SENTIMENT RATINGS

The case to be studied next is the one in which the greatest amount of information will be associated to each date; the share values of the company and its competitors, the Twitter posts (the average of the sentiment) and the amount of these posts for each company (a total of 18 parameters). The objective of this last analysis is to determine whether the algorithm is able to properly parse the information and then make accurate predictions with the conclusions reached.

The first computation indicates that, compared to the case where only sentiment data was used, the model has more problems in computing the information. Clearly this is because by using only the preceding five days it is very difficult to create a pattern that explains the behavior of the stock (it is important to note that now, for each day, 30% more information is being computed).

As before, the performance of the test set is better than that of the training set. It is important to remember that this happens quite often when the parameters have not been learned well (yet); once the hyperparameter optimization starts, the weight matrices will assign more coherent values and this will no longer happen.



```
Cost after iteration 4600: 0.690333
Cost after iteration 4700: 0.690329
Cost after iteration 4800: 0.690327
Cost after iteration 4900: 0.690327
Accuracy: 0.5375000000000001
Accuracy: 0.6039603481859653
```

Figure 6-36: first computation (9)

On this occasion, the analysis of the first hyperparameter once again provides good news, giving record values in prediction efficiency and maintaining a very low overfitting.

As can be seen, using 180 days as a benchmark results in a difference between the training and test set returns of approximately 1% and a record cost function of 0.051422. The fact that the best results are obtained using twenty-five days more than in the last section makes sense since in this case one is dealing with more elaborate data. The graph showing the evolution of the different indicators will be presented below together with the final values of the model.

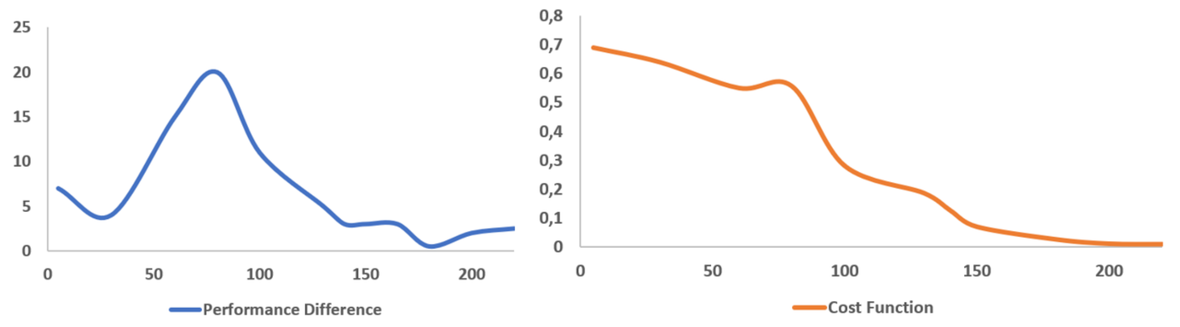
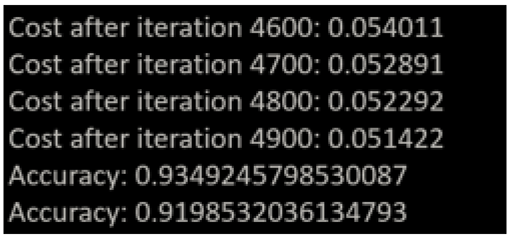


Figure 6-37: performance difference & cost function (5)

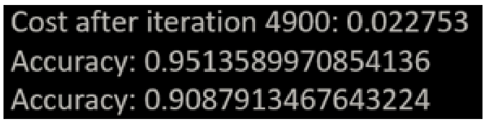


```
Cost after iteration 4600: 0.054011
Cost after iteration 4700: 0.052891
Cost after iteration 4800: 0.052292
Cost after iteration 4900: 0.051422
Accuracy: 0.9349245798530087
Accuracy: 0.9198532036134793
```

Figure 6-38: performance for 180 reference days

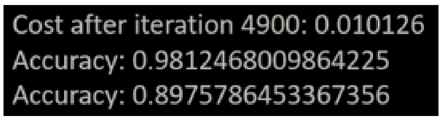
Although it has already been explained in the previous sections, it is reiterated that in this type of models it is not recommended to increase the use of the number of iterations because, given the complexity of the data and the number of them used as a reference in each computation, the computational costs increase significantly while the improvements in the model are almost imperceptible (if not worse).

In the same line as the other Deep Neural Networks, the use of more hidden layers has not resulted in an increase of efficiency in the model (values similar to those already obtained are achieved), however, in some cases they have triggered major overfitting problems.



```
Cost after iteration 4900: 0.022753
Accuracy: 0.9513589970854136
Accuracy: 0.9087913467643224
```

Figure 6-40: performance for three layers (4,2,5)



```
Cost after iteration 4900: 0.010126
Accuracy: 0.9812468009864225
Accuracy: 0.8975786453367356
```

Figure 6-39: performance for four layers (5,1,3,3)

Chapter 7. CONCLUSIONS AND FUTURE PROJECTS

The conclusions obtained in this study will be initially analysed based on the main objectives stablished in chapter 4.2.1.

1. ***Creation of three models of varying complexity that allow predicting (for different types of data) the value that a company stock should have.*** This objective has been effectively fulfilled as the following partial objectives have been achieved:

a) Logistic Regression model. Through this project it has been possible to build a model that uses logistic regression to analyse the characteristics of the elements of an input matrix. In addition, during the construction of this model it has been possible to effectively provide the algorithm with great flexibility in order to be able to adequately interpret the different types of input values and to optimize the predictive capabilities of this tool.

b) Neural Network models. Through this project, it has been possible to implement models that use artificial neural networks to predict the value of a share in the stock market. In addition, by giving the model a high level of flexibility, it has been possible to understand in detail how this type of tool works and to identify the type of data for which this model or a less elaborate model such as logistic regression should be used.

2. Analysis on the performance of the different models and analysis of the impact of different types of data. This objective has been fully accomplished as the following partial objectives have been achieved. The analysis and comparison of the different models will be done separately for each type of data used. After the individual analyses for each case, it will be concluded which model is the most recommendable:

a) ***Evaluation of the performance of the different models and their predictive effectiveness using only stock values.*** When only the evolution of the stock value is taken into consideration, it can be observed that the three models show a particularly good predictive capacity (all more than 80% efficiency in both the test and the training set). It is also interesting that from the first computation, the values of the logistic regression model are satisfactory and there is no time when the difference between the performances is remarkable (always around 1%); this leads to the conclusion that this type of model is able to learn and generalize very well when the data is not very complex. Be that as it may, this model is no match for neural networks. As can be seen in the previous section, this type of more elaborate models allows the cost function value to be reduced to a minimum of 0.135035 and, in the case of deep networks, test set performances of 89.8% are achieved, which is a very good value for previously unseen data. These last values are very good and allow a significant reduction of investment risks.

b) ***Evaluation of the performance of the different models and their predictive effectiveness using only sentiment scores.*** Unlike the previous case, the increase in data complexity is noticeable in the predictive capabilities of the logistic regression model. Although low overfitting is maintained, the predictive efficiency does not exceed 70% in any case; this is interpreted as a greater difficulty on the part of the model to understand the pattern that explains the relationship between the X and Y matrices. As expected, this complexity is not an obstacle in the other two models and the percentages are still pretty good.

Another interesting thing to comment is that, in all three models, the efficiency in this case is slightly worse than with the historical values of the stock, this allows to conclude that the individual use of this type of data provides less exploitable information than the use of only the numerical evolution of the stock.

c) ***Evaluation of the performance of the different models and their predictive effectiveness using both sentiment scores and the stock values.*** The latter case is much more similar to the first one seen above. Here almost all the indicators return to percentages above 80% efficiency (in the logistic regression the test set gives an efficiency of 79.55%). The reason for this is that, although the information provided is even more elaborate, it allows the pattern of behaviour to be shown more clearly than when only the sentiment scores were used. In this case, record values are also recorded for both neural network models, obtaining efficiencies of 92% and 90% in the single-layer model and 93.5% and 92% in the deeper model.

As can be seen, in the best-case scenario, the yield values do not exceed 93%. This is something that may surprise many because in other cases (see Figure 6-35, Figure 6-28) it has been seen that the training set is capable of reaching much higher values. The reason why the best performance is set at 93 is because it is the performance that allows the model to learn the behavioural patterns of the data and convey them in an optimal way; for higher values of the training set much lower values of the test set are obtained. This means that the model learns the values by heart and due to its inability to generalize it becomes useless.

Taking into account all the results obtained, it can be concluded that there is a proportional relationship between the effectiveness of the models and their level of complexity; the higher the complexity, the better the performance of the algorithm. For this reason, it is concluded that the model that, for the available data, best predicts the value of a stock is the Deep Neural Network.

In addition to determining which model was the most recommendable, another purpose of the work was to determine which data source generated the best predictions for each system. From the results obtained, it can be concluded that the most elaborate data source (that of stock values together with sentiment ratings) generates the best results. What is interesting here is that the second most useful data source was not the sentiment ratings but the stock values; the reason for this is because, being simpler information, it was easier for the different models to understand the relationship between the input matrices and therefore predict future behaviours more accurately. These findings are very interesting because they reinforce the theoretical basis of the technical analysis, which suggest that stock markets move in recognizable patterns and therefore are not only driven by external actions such as social media postings [14].

3. ***Analysis of the impact of the different hyperparameters on the performance of the models.*** After analysing the effect of each of the hyperparameters separately, it is concluded that the element that most influences the performance of the different models is clearly the number of previous days to be taken into account. This makes sense, since in many studies regarding model performance through hyperparameter tuning it is stated that it is one of the most important ones (if not the most important) [39], [44], [45]. The reason for this setting to be so important is because the increase of the number pairs of data for each of the training examples promotes and facilitates pattern recognition.

Another hyperparameter that should not go unmentioned is the learning rate. The reason why this setting is so important is because it determines the speed of optimization of the cost function and, if not chosen properly, it will lead to severe overfitting, thus reducing the effectiveness of the model. Since this is an element that is very difficult to choose, in this project it was decided to use the recommended (by experts) values for each model.

It is also noteworthy how increasing both the hidden units and the number of layers of the neural network at almost no time has generated benefits. This is because the quantity, variety and quality of the data used in this project is not high enough. One has to think that Deep Neural Networks are used for tasks such as autonomous driving, in that type of models the complexity of the elements is so great that a complexity of the model according to it is needed, in this case these levels of sophistication were not achieved and therefore it has not been possible to see the real impact of the elements that are revolutionizing so many areas of society.

Through this project it has been observed how it is possible to create highly effective models from open-source databases. In addition, it has been confirmed that the joint use of variables not directly linked to the stock market can help to improve the predictive capabilities of the model. But above all, these conclusions give rise to new questions such as "Does the performance of one industrial sector condition the development of another? Does it influence enough to be a factor to be computed in the data matrices?", "If instead of publications on Twitter, headlines from the main media were used, would better performance be achieved?". Another important issue is the amount of data available, in this case this element is believed to have been a limiting factor of the model and therefore a future project could be developed collaborating with entities with access to larger databanks so that this is not a constraint on the results of the project.

Chapter 8. BIBLIOGRAPHY

[1] M. R. Vargas, C. E. M. Dos Anjos, G. L. G. Bichara, and A. G. Evsukoff, "Deep Leaming for Stock Market Prediction Using Technical Indicators and Financial News Articles," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2018-July, Oct. 2018, doi: 10.1109/IJCNN.2018.8489208.

[2] A. E. de Oliveira Carosia, G. P. Coelho, and A. E. A. da Silva, "Investment strategies applied to the Brazilian stock market: A methodology based on Sentiment Analysis with deep learning," *Expert Syst. Appl.*, vol. 184, p. 115470, Dec. 2021, doi: 10.1016/J.ESWA.2021.115470.

[3] W. Jiang, "Applications of deep learning in stock market prediction: Recent progress," *Expert Syst. Appl.*, vol. 184, p. 115537, Dec. 2021, doi: 10.1016/J.ESWA.2021.115537.

[4] A. Géron, "Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems," 2019, Accessed: Aug. 10, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=HHetDwAAQBAJ&oi=fnd&pg=PP1&dq=aurélien+géron+hands-on&ots=0Lpm2wngSo&sig=_o5fanFb-XbgJZlAuD5NWh54HpE.

[5] K. Gurney, "An Introduction to Neural Networks," Oct. 2018, doi: 10.1201/9781315273570.

[6] V. Kreinovich, "From Traditional Neural Networks to Deep Learning: Towards Mathematical Foundations of Empirical Successes," *Stud. Fuzziness Soft Comput.*, vol. 393, pp. 387–397, 2021, doi: 10.1007/978-3-030-47124-8_31.

[7] J. Skolnick, W. Jones, K. Alasoo, D. Fishman, and L. Parts, “Computational biology: deep learning,” *Emerg. Top. Life Sci.*, vol. 1, no. 3, pp. 257–274, Nov. 2017, doi: 10.1042/ETLS20160025.

[8] W. McCulloch, W. Pitts, and D. Hebb, “Breve Historia de las Redes Neuronales Artificiales (Articulo 1) Created with Sketch.,” *steemit.com*, Accessed: Aug. 11, 2021. [Online]. Available: https://steemit.com/spanish/@iars.geo/breve-historias-de-las-redes-neuronales-artificiales-articulo-1.

[9] L. L. Leyva and M. A. Pérez, “Neurona artificial de McCulloch & Pitts,” 2007, Accessed: Aug. 11, 2021. [Online]. Available: https://148.204.103.131/handle/123456789/8640.

[10] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, Nov. 1958, doi: 10.1037/H0042519.

[11] A. Anta, L. Chiroque, P. M.-… del lenguaje natural, and undefined 2013, “Sentiment analysis and topic detection of Spanish tweets: A comparative study of of NLP techniques,” *journal.sepln.org*, 2013, Accessed: Aug. 11, 2021. [Online]. Available: http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/download/4658/2760.

[12] B. G. Inchausti, C. R. Maya, and M. A. Gisbert, “El papel del análisis fundamental en la investigación del mercado de capitales: Análisis crítico de su evolución,” *Rev. Esp. Financ. y Contab.*, vol. 31, no. 114, pp. 1111–1150, 2002, doi: 10.1080/02102412.2002.10779470.

[13] B. Graham, D. Dodd, and S. Cottle, “Security analysis,” 1934, Accessed: Aug. 17, 2021. [Online]. Available: https://bridgecollege.net/download/75/accounting-and-finance/6485/00899.pdf.

[14] J. Chen, “Essentials of technical analysis for financial markets,” 2010, Accessed:

Aug. 17, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=202I0ZWA6tIC&oi=fnd&pg=PR13&dq=technical+analysis+of+the+financial+markets&ots=kj5ZIs13C4&sig=x4plLnpCU4MvDr0cl5OCecYVuK8.

[15] A. Lo and J. Hasanhodzic, “The evolution of technical analysis: financial prediction from Babylonian tablets to Bloomberg terminals,” 2010, Accessed: Aug. 17, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=aJv9DwAAQBAJ&oi=fnd&pg=PR7&dq=technical+analysis+munehisa+homma&ots=crkJzWRQwU&sig=GR_Z-fHu7JjQY5VwsK3ObH8oJ0c.

[16] B. Dormeier, “The History of Technical Analysis,” 2011, Accessed: Aug. 17, 2021. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=aOFJdcsn76QC&oi=fnd&pg=PT3&dq=technical+analysis+munehisa+homma&ots=oR_89TYun7&sig=PI9PIhZDij-dWC3aD5zOVTz8NRM.

[17] G. Griffioen, “Technical analysis in financial markets,” 2003, Accessed: Aug. 17, 2021. [Online]. Available: https://papers.ssrn.com/sol3/papers....1&srcabs=566882.

[18] M. García, A. Jalal, L. Garzón, J. L.-E. de Economía, and undefined 2013, “Métodos para predecir índices bursátiles,” *redalyc.org*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.redalyc.org/pdf/3290/329029209003.pdf.

[19] H. Wang and B. Raj, “On the Origin of Deep Learning,” Feb. 2017, Accessed: Aug. 18, 2021. [Online]. Available: http://arxiv.org/abs/1702.07800.

[20] P. Franses, H. G.-I. J. of forecasting, and undefined 1999, “Additive outliers, GARCH and forecasting volatility,” *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available:

https://www.sciencedirect.com/science/article/pii/S0169207098000533?casa_token =aBM7uAAWKPEAAAAA:o2Z380q6kCSJA2XlWAEcmyoeolczgkZeoP4hxlImR b1Xw8SOFuys_AgV6O9ZZjXQEG95wAr35Q.

[21] H. Liu, J. H.-E. S. with Applications, and undefined 2010, "Forecasting S&P-100 stock index volatility: The role of volatility asymmetry and distributional assumption in GARCH models," *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417409010689.

[22] S. Z.-P. A. S. M. and its Applications and undefined 1999, "Nonlinear index prediction," *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378437199000916.

[23] S. Chaigusin, … C. C.-2008 I., and undefined 2008, "The use of neural networks in the prediction of the stock exchange of Thailand (SET) Index," *ieeexplore.ieee.org*, pp. 670–673, 2008, doi: 10.1109/CIMCA.2008.83.

[24] E. Guresen, G. Kayakutlu, T. D.-E. S. with Applications, and undefined 2011, "Using artificial neural network models in stock market index prediction," *Elsevier*, doi: 10.1016/j.eswa.2011.02.068.

[25] P. Pai, C. L.- Omega, and undefined 2005, "A hybrid ARIMA and support vector machines model in stock price forecasting," *Elsevier*, vol. 33, pp. 497–505, 2005, doi: 10.1016/j.omega.2004.07.024.

[26] Y. Zhang, L. W.-E. systems with applications, and undefined 2009, "Stock market prediction of S&P 500 via combination of improved BCO approach and BP neural network," *Elsevier*, Accessed: Aug. 18, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800852X.

[27] "Hybridization of evolutionary Levenberg–Marquardt neural networks and data pre-processing for stock market prediction," *Elsevier*, Accessed: Aug. 18, 2021. [Online].

Available: https://www.sciencedirect.com/science/article/pii/S095070511200130X.

[28] K. Hussainey and L. Khanh Ngoc, "The impact of macroeconomic indicators on Vietnamese stock prices," *J. Risk Financ.*, vol. 10, no. 4, pp. 321–332, Aug. 2009, doi: 10.1108/15265940910980632/FULL/HTML.

[29] D. P.-T. and E. D. of and undefined 2010, "Macroeconomic indicators and their impact on stock market performance in the short and long run: the case of the Baltic States," *ceeol.com*, vol. 16, no. 2, pp. 291–304, 2010, doi: 10.3846/tede.2010.19.

[30] J. Y. Huang and J. H. Liu, "Using social media mining technology to improve stock price forecast accuracy," *J. Forecast.*, vol. 39, no. 1, pp. 104–116, Jan. 2020, doi: 10.1002/FOR.2616.

[31] D. Valle-Cruz, V. Fernandez-Cortez, A. López-Chau, and R. Sandoval-Almazán, "Does Twitter Affect Stock Market Decisions? Financial Sentiment Analysis During Pandemics: A Comparative Study of the H1N1 and the COVID-19 Periods," *Cogn. Comput. 2021*, vol. 1, pp. 1–16, Jan. 2021, doi: 10.1007/S12559-021-09819-8.

[32] S. Coyne, P. Madiraju, J. C.-2017 I. 15th I. C. on, and undefined 2017, "Forecasting stock prices using social media analysis," *ieeexplore.ieee.org*, Accessed: Aug. 18, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8328513/.

[33] "LOS 17 OBJETIVOS | Sustainable Development." https://sdgs.un.org/es/goals (accessed Aug. 18, 2021).

[34] "Dogecoin: Elon Musk sacude el mercado de las criptomonedas con una encuesta en Twitter | Mercados | Cinco Días." https://cincodias.elpais.com/cincodias/2021/05/11/mercados/1620743665_091906.html (accessed Aug. 12, 2021).

[35] Y. Xu and S. B. Cohen, "Stock Movement Prediction from Tweets and Historical

Prices," *ACL 2018 - 56th Annu. Meet. Assoc. Comput. Linguist. Proc. Conf. (Long Pap.*, vol. 1, pp. 1970–1979, 2018, doi: 10.18653/V1/P18-1183.

[36] C. Hutto, E. G.-P. of the I. A. Conference, and undefined 2014, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," *ojs.aaai.org*, 2014, Accessed: Aug. 12, 2021. [Online]. Available: https://ojs.aaai.org/index.php/ICWSM/article/view/14550.

[37] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *J. Biomed. Inform.*, vol. 35, no. 5–6, pp. 352–359, Oct. 2002, doi: 10.1016/S1532-0464(03)00034-0.

[38] "Facebook gana la aprobación preliminar ante la demanda de reconocimiento facial." https://www.lavanguardia.com/redes-sociales/20200820/482914568544/facebook-gana-aprobacion-preliminar-demanda-privacidad.html (accessed Aug. 16, 2021).

[39] "Improving Deep Neural Networks: Hyperparameter Tuning, Regularization and Optimization | Coursera." https://www.coursera.org/learn/deep-neural-network?specialization=deep-learning (accessed Aug. 14, 2021).

[40] P. M. GRANITTO, P. F. VERDES, H. D. NAVONE, and H. A. CECCATTO, "A LATE-STOPPING METHOD FOR OPTIMAL AGGREGATION OF NEURAL NETWORKS," *http://dx.doi.org/10.1142/S0129065701000758*, vol. 11, no. 3, pp. 305–310, Nov. 2011, doi: 10.1142/S0129065701000758.

[41] R. M. Cantón Croda, D. E. Gibaja Romero, and S.-O. Caballero Morales, "Sales Prediction through Neural Networks for a Small Dataset," *IJIMAI, ISSN-e 1989-1660, Vol. 5, Nº. 4, 2019, págs. 35-41*, vol. 5, no. 4, pp. 35–41, 2019, Accessed: Aug. 15, 2021. [Online]. Available: https://dialnet.unirioja.es/servlet/articulo?codigo=6895714&info=resumen&idioma=ENG.

[42] N. Fatima, “79-90 eISSN: 2255-2863 A Comparative Analysis of Optimization Algorithms ADCAIJ: Advances in Distributed Computing and,” *ADCAIJ Adv. Distrib. Comput. Artif. Intell. J. Regul. Issue*, vol. 9, pp. 79–90, 2020, doi: 10.14201/ADCAIJ2020927990.

[43] F. Hutter, L. Kotthoff, and J. Vanschoren, “The Springer Series on Challenges in Machine Learning Automated Machine Learning Methods, Systems, Challenges,” Accessed: Aug. 16, 2021. [Online]. Available: http://www.springer.com/series/15602.

[44] M. Feurer, F. H.-A. machine learning, and undefined 2019, “Hyperparameter optimization,” *library.oapen.org*, Accessed: Aug. 19, 2021. [Online]. Available: https://library.oapen.org/bitstream/handle/20.500.12657/23012/1007149.pdf?sequence=1#page=15.

[45] M. Parsa, J. Mitchell, … C. S.-F. in, and undefined 2020, “Bayesian multi-objective hyperparameter optimization for accurate, fast, and efficient neural network accelerator design,” *frontiersin.org*, Accessed: Aug. 19, 2021. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fnins.2020.00667/full?report=reader.

[46] United Nations, “Sustainable Development Goals and Events,” 2015. .

[47] A. F. Agarap, “Deep Learning using Rectified Linear Units (ReLU),” Mar. 2018, Accessed: Aug. 19, 2021. [Online]. Available: http://arxiv.org/abs/1803.08375.

[48] K. Eckle, J. S.-H.-N. Networks, and undefined 2019, “A comparison of deep networks with ReLU activation function and linear spline-type methods,” *Elsevier*, Accessed: Aug. 19, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608018303277.

[49] “Searching for activation functions,” *6th Int. Conf. Learn. Represent. ICLR 2018 - Work. Track Proc.*, 2018.

ANNEX I – ALIGNMENT OF PROJECT WITH SDGS

This project is in alignment with some of the goals and targets declared by the United Nations in 2015 as the Sustainable Development Goals (SDGs). The principal sustainable development goal met by this project is Goal 4, fulfilling targets 4.4 and 4.a.

Goal 4: *"Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all."* [46]

> **Target 4.4**: *"By 2030, substantially increase the number of youth and adults who have relevant skills, including technical and vocational skills, for employment, decent jobs and entrepreneurship."*[46] No resources or tools that require additional payment or special access have been used for the realization of this project. The intention of this is to demonstrate that through the information available on the Internet it is possible to create highly effective models.

> **Target 4.a**: *"Build and upgrade education facilities that are child, disability and gender sensitive and provide safe, non-violent, inclusive and effective learning environments for all."*[46] All codes and databases used for the realization of this project will be uploaded to the opensource platform GitHub, the purpose of this is to help others to continue with the research done in this field of study and to encourage the use of these platforms to continue learning in a friendly environment.

Goal 8: *"Promote sustained, inclusive and sustainable economic growth, full and productive employment and decent work for all."*[46]

> **Target 8.2**: *"Achieve higher levels of economic productivity through diversification, technological upgrading and innovation, including through a focus on high-value added and labour-intensive sectors"*[46] Just as the codes and databases used will be shared, the matrices (already trained) of the weights and biases obtained in the best

model will also be published on GitHub. By doing this, one allows others to use these matrices to make their predictions and to invest in a more controlled way (before using them, it is recommended to test them and ensure their effectiveness).

Goal 10: *"Reduce inequality within and between countries."* [46]

Target 10.3: *"Ensure equal opportunity and reduce inequalities of outcome, including by eliminating discriminatory laws, policies and practices and promoting appropriate legislation, policies and action in this regard."*[46] By publishing all the material used and the weights and bias matrices on platforms such as GitHub, indiscriminate learning is encouraged since these are opensource platforms. Furthermore, during the project there was an active participation in discussion forums such as Stack Overflow, participation in these forums is always a very useful way to help others, since the resolution of someone's problem can surely help others to solve their own ones.

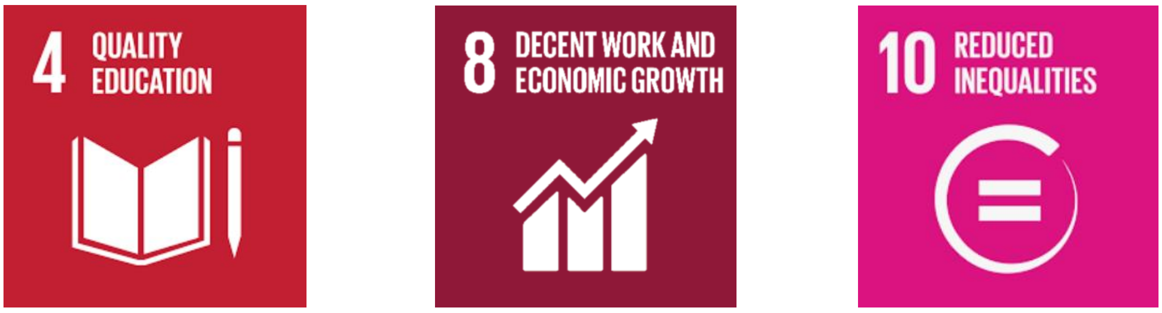


Figure 0-1: Sustainable Development Goals 4, 8 and 10. Source: United Nations

ANNEX II – TANH VS. RELU VS. LEAKY RELU FUNCTIONS

It has been explained before that when working with Deep Neural Networks it is necessary to use, in the intermediate layers, functions that do not reduce the value of the derivatives in order to avoid problems related to vanishing gradients. It has also been explained that there are several activation functions that meet these requirements (hyperbolic tangent, ReLU and Leaky ReLU), but what has not been explained is why it has been decided to use the ReLU function instead of the others. The purpose of this annex is to shed light on this issue.

The truth is that all three functions can be used indifferently and adequate results will be obtained. The reason why there is a tendency to use the ReLU function and why this project has also opted for it is that when one is computing the algorithm, for the optimization of the parameters it is necessary to program the gradient of the loss function and apply the chain rule for the different layers; this means that it is necessary to compute the derivative of the activation functions used. Taking into account that the ReLU function is a combination of a line whose derivative is 0 and another whose derivative is 1, the creation of the derivative is very simple. Meanwhile, the derivative of the cost tanh is much more difficult and there is the added risk that is the value of x exceed the interval [-2, 2], then the derivative becomes very small and the network might not work due to the incapacity to converge.

As to why the Leaky ReLU is not used, the reason is because the ReLU function is the one that is most recommended by experts [47], [48]. However, recent studies have proved that, in neural networks that use a lot of hidden layers, the use of the Leaky ReLU activation function provides allows better results to be obtained [49]. Since this project does not consider the use of models with too many hidden layers, it was decided to follow the advice of experts and use the ReLU function.