COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

DEGREE IN ENGINEERING FOR INDUSTRIAL TECHNOLOGIES

BACHELOR'S THESIS

SUPERVISED LEARNING CLASSIFIERS APPLIED TO THE ANALYSIS OF FUNDAMENTAL DATA OF LISTED COMPANIES. DEVELOPMENT OF PREDICTIVE MODELS

Author: Javier Reyzabal Roig

Director: Prof. Marco Lippi

Madrid

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

Supervised Learning Classifiers applied to the Analysis of Fundamental Data of Listed Companies. Development of Predictive Models

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/2021 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.: Javier Reyzabal Roig Fecha: 26/ 08/ 2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Prof. Marco Lippi Fecha: 26/ 08/ 2021

Marco Lippi

d) Asumir la responsabilidad en el caso de que las instituciones fueran condenadas por infracción de derechos derivada de las obras objeto de la cesión.

6º. Fines y funcionamiento del Repositorio Institucional.

La obra se pondrá a disposición de los usuarios para que hagan de ella un uso justo y respetuoso con los derechos del autor, según lo permitido por la legislación aplicable, y con fines de estudio, investigación, o cualquier otro fin lícito. Con dicha finalidad, la Universidad asume los siguientes deberes y se reserva las siguientes facultades:

- La Universidad informará a los usuarios del archivo sobre los usos permitidos, y no garantiza ni asume responsabilidad alguna por otras formas en que los usuarios hagan un uso posterior de las obras no conforme con la legislación vigente. El uso posterior, más allá de la copia privada, requerirá que se cite la fuente y se reconozca la autoría, que no se obtenga beneficio comercial, y que no se realicen obras derivadas.
- La Universidad no revisará el contenido de las obras, que en todo caso permanecerá bajo la responsabilidad exclusive del autor y no estará obligada a ejercitar acciones legales en nombre del autor en el supuesto de infracciones a derechos de propiedad intelectual derivados del depósito y archivo de las obras. El autor renuncia a cualquier reclamación frente a la Universidad por las formas no ajustadas a la legislación vigente en que los usuarios hagan uso de las obras.
- La Universidad adoptará las medidas necesarias para la preservación de la obra en un futuro.
- La Universidad se reserva la facultad de retirar la obra, previa notificación al autor, en supuestos suficientemente justificados, o en caso de reclamaciones de terceros.

Madrid, a 26 de agosto de 2021

ACEPTA

Fdo. [illegible]

Motivos para solicitar el acceso restringido, cerrado o embargado del trabajo en el Repositorio Institucional:

COMILLAS
UNIVERSIDAD PONTIFICIA

ICAI

DEGREE IN ENGINEERING FOR INDUSTRIAL TECHNOLOGIES

BACHELOR'S THESIS

SUPERVISED LEARNING CLASSIFIERS APPLIED TO THE ANALYSIS OF FUNDAMENTAL DATA OF LISTED COMPANIES. DEVELOPMENT OF PREDICTIVE MODELS

Author: Javier Reyzabal Roig

Director: Prof. Marco Lippi

Madrid

Acknowledgments

I want to thank my mother, Angela. Thank you.

CLASIFICADORES DE APRENDIZAJE SUPERVISADO APLICADOS AL ANÁLISIS DE DATOS FUNDAMENTALES DE EMPRESAS COTIZADAS. DESARROLLO DE MODELOS PREDICTIVOS

Autor: Reyzabal Roig, Javier
Director: Lippi, Marco
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El proyecto que se resume a continuación tiene como principal objetivo el desarrollo de modelos de predicción en bolsa a medio plazo utilizando regresiones logísticas y maquinas de vectores de soporte kernealizadas (*K-SVM*). El trabajo concluye con la obtención de dos modelos, uno para cada clasificador utilizado, siendo el obtenido mediante *K-SVM* el que demostró ser el más eficaz, ofreciendo una exactitud del 67% y una precisión de clasificación cercana al 70%.

Palabras clave: *Machine Learning*, Aprendizaje Supervisado, Regresiones Logísticas, Máquinas de Vectores de Soporte, Finanzas, Análisis Fundamental.

1. Introducción

Hoy en día, la Inteligencia Artificial (*IA*) se utiliza en casi todos los sectores, incluido el financiero. Según un estudio llevado a cabo por analistas de *PwC*, la *IA* podría aumentar el *PIB* mundial en torno al 14% antes de 2030 [1]. Teniendo esto en cuenta, es fácil comprender por qué esta ha llegado a ser una de las ciencias que más impulso esta recibiendo en los últimos años.

Este proyecto busca aplicar técnicas de Machine Learning (*ML*), uno de los campos más importantes de la *IA*, al análisis fundamental de compañías para inversión en bolsa a medio plazo. En concreto, se utilizarán clasificadores lineales y kernealizados para tratar de obtener, a partir de una serie de datos financieros de compañías pertenecientes al *S&P500*, un modelo predictivo que permita clasificar las acciones de una compañía en función de su desempeño previsto en bolsa.

Se plantea entonces una tarea de clasificación binaria, en la que la clase positiva '1' englobará las compañías para las que se prevé, en el siguiente cuatrimestre, una subida en bolsa. Los dos algoritmos de aprendizaje automático que se utilizarán son la regresión logística (RL) y las maquinas de vectores de soporte kernealizadas (*K-SVM*).

2. Estado de la Técnica

Hay que destacar que, evidentemente, existen numerosos trabajos que buscan generar modelos predictivos en bolsa utilizando técnicas de *ML*. La mayoría de ellos, se centran en el análisis técnico, es decir, en analizar la progresión de una acción en el corto plazo.

Esto no implica que no existan proyectos de modelización en bolsa a largo plazo. Este área de la inversión en bolsa también ha sido profundamente explorada. Por ejemplo, Timor, Dinçer y Emir, utilizando *K-SVM* como algoritmo clasificador para la selección de acciones a largo plazo, obtuvieron rendimientos cercanos al 66% [2].

3. Descripción de las Tecnologías

Dentro del *ML*, existen numerosos tipos de algoritmos que pueden ser utilizados para finalidades muy diversas. Este proyecto se centra en los métodos de Aprendizaje Supervisado, concretamente en RL y SVM.

Estos algoritmos permiten entrenar clasificadores aportando para ello datos históricos ya clasificados. Su entrenamiento consiste básicamente en optimizar, a partir de dichos datos, una función de discriminación (lineal o no) que permita separar las diferentes clases en el espacio.

En cuanto a la regresión logística, cabe mencionar que se trata de un algoritmo de clasificación lineal, es decir, que es útil para lo problemas con datos linealmente separables [3]. Este tipo de clasificador busca maximizar la probabilidad de pertenencia a una determinada clase.

Por otro lado, los *SVM* también son considerados clasificadores lineales, sin embargo, en este proyecto se utilizarán los denominados *SVM kernealizados*, que permiten crear separaciones no lineales en los datos [4]. Por lo tanto, se puede decir que se utilizarán clasificadores lineales (*RL*) y no lineales (*K-SVM*).

4. La Base de Datos

Un algoritmo de ML es capaz de aprender gracias a la información que se le aporta. Por ello, para llevar a cabo un buen proyecto, se necesitará una base de datos completa y que contenga información relevante para los clasificadores.

Se seleccionaron 54 compañías del *S&P500* pertenecientes a 9 sectores diferentes, de las que se descargó la información utilizando *Bloomberg Terminal*, una de las herramientas financieras más potentes del mundo. Para cada empresa, se obtuvieron los datos correspondientes a un total de 58 variables financieras de todos los cuatrimestres desde el 1 de enero de 2010 hasta 31 de diciembre de 2019. Así, se consiguió crear una base de datos que, una vez estructurada, presentaba la siguiente forma:

	Fila	***Sector***	***Compañía***	***Cuatrimestre***	***Precio***	***ΔPrecio***	***Clase***	***Características***	
TRAINING	*1*	*BM*	*C1*	*[1,32]*	P^1_{1-32}	ΔP^1_{1-32}	*0/1*	...	...
	.		*C2*	*[1,32]*	P^2_{1-32}	ΔP^2_{1-32}	*0/1*	...	...
	.		*C3*	*[1,32]*	P^3_{1-32}	ΔP^3_{1-32}	*0/1*	...	...
	.		*C4*	*[1,32]*	P^4_{1-32}	ΔP^4_{1-32}	*0/1*	...	...
	.		*C5*	*[1,32]*	P^5_{1-32}	ΔP^5_{1-32}	*0/1*	...	...
	.		*C6*	*[1,32]*	P^6_{1-32}	ΔP^6_{1-32}	*0/1*	...	...
	.	*CD*	*C1*	*[1,32]*	*P*	ΔP	*0/1*	...	...
	.		...	...	...	...	...	...	...
	.	*CS*	*C1*	...	*P*	ΔP	*0/1*	...	...
	. *1728*	...	...	...	...	...	...	...	...
TEST	*1729*	*BM*	...	*[33,43]*	*P*	ΔP	*0/1*	...	...
	.	*CD*	...	*[33,43]*	*P*	ΔP	*0/1*	...	...
	.	...	...	...	...	...	...	...	...
	2322	...	...	...	...	...	...	...	...

Ilustración 1 – Estructura de la base de datos

Como se observa en la Ilustración 1, los datos se dividieron en dos, un conjunto de entrenamiento para obtener los modelos y un conjunto de prueba para testar su eficacia. Así, la base de datos ya estructurada contenía 2.322 filas y 58 columnas, lo que hacía un total de 185.760 celdas de datos elementales.

Por otro lado, teniendo en cuenta que tanto los algoritmos de *R*L como los de *SVM* son muy susceptibles a diferencias de escala en los datos [4], se decidió estandarizarlos para, posteriormente, comprobar si en este caso lo modelos mejoraban aplicando esta modificación en los datos.

Posteriormente, en un primer análisis visual de los datos, se comprobó que las variables incluían un gran número de *outliers*. A continuación, en la Ilustración 2, se muestra el diagrama de cajas y bigotes de dos de las variables estudiadas:

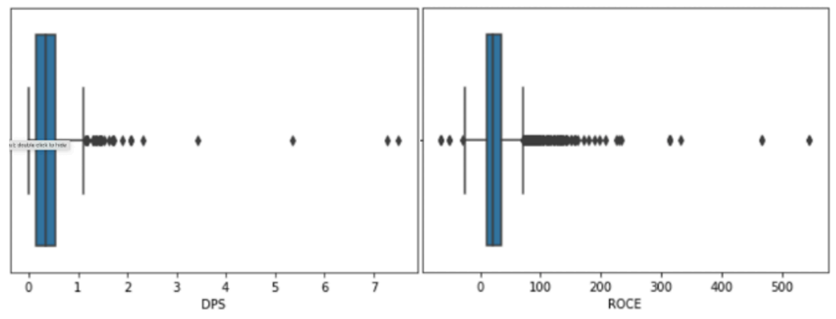


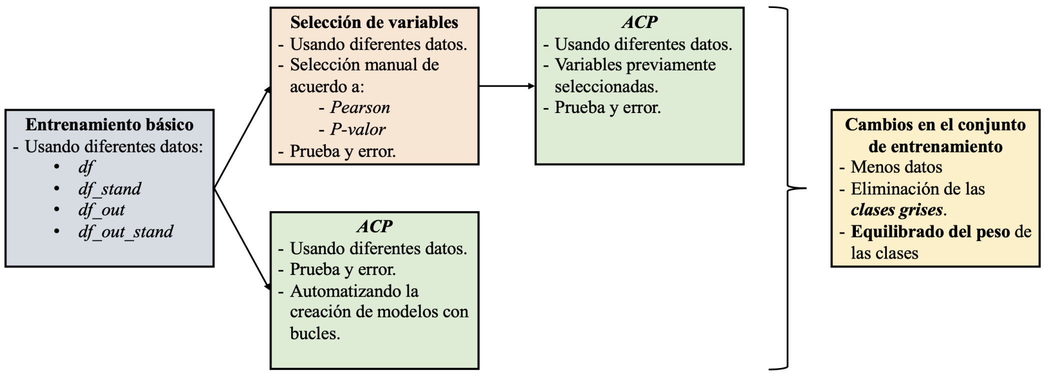
Ilustración 2 – Diagramas de cajas y bigotes de las variables ROCE y DPS

Al igual que con los datos estandarizados, se decidió crear entonces un conjunto de entrenamiento al que se le eliminaron los *outliers* más extremos, aquellos con un *Z-Score* mayor de 5. De esta forma, se comprobaría posteriormente si utilizar estos valores extremos permitiría mejorar los modelos.

Por último, se llevaron a cabo los procedimientos típicos de preprocesamiento de datos para adaptarlos a los algoritmos que se utilizarían. De esta forma, se eliminaron las filas con datos ausentes, se codificaron las variables nominales, etc.

5. Obtención de los Modelos

La metodología utilizada para entrenar los modelos se esquematiza en la Ilustración 3.



Entrenamiento básico
- Usando diferentes datos:
• df
• df_stand
• df_out
• df_out_stand
Selección de variables
- Usando diferentes datos.
- Selección manual de acuerdo a:
- Pearson
- P-valor
- Prueba y error.
ACP
- Usando diferentes datos.
- Variables previamente seleccionadas.
- Prueba y error.
ACP
- Usando diferentes datos.
- Prueba y error.
- Automatizando la creación de modelos con bucles.
Cambios en el conjunto de entrenamiento
- Menos datos
- Eliminación de las clases grises.
- Equilibrado del peso de las clases

Ilustración 3 – Metodología de entrenamiento de los modelos

Antes de comenzar a explicar la Ilustración 3, es importante mencionar que existía un desequilibrio en los datos. Con esto se quiere decir que la 'clase 1' tenía más representación que la 'clase 0', en concreto un 63% frente a un 37%. Debido a esto, modelo de decisión más básico que se podía entrenar era un modelo que clasificase todas las muestras en la 'clase 1'. Este modelo básico tendría una precisión y una exactitud del 63%. El objetivo que se buscó al entrenar los modelos posteriores fue, por tanto, superar las prestaciones de este último.

El proceso de entrenamiento se inició con las regresiones logísticas. Como se indica en el esquema, en primer lugar, se entrenaron una serie de modelos simples que utilizaban las 58 variables de la base de datos. Al comprobar la ineficacia de estos modelos, se realizó una selección manual de las características, que dio los mejores resultados con el conjunto de datos estandarizado. Se aplicaron también técnicas de reducción de la dimensión de los datos basadas en el método de Análisis de las Componentes Principales (ACP), que se basa en generar nuevas columnas preservando la varianza de los datos [5].

Los modelos obtenidos de esta forma demostraron no ser óptimos. En general, presentaban una especificidad muy baja (inferior al 30%) y una precisión cercana al 64%. Por este motivo, y con el fin de mejorar sobre todo la especificidad, se decidió buscar un método que permitiese al algoritmo reconocer mejor las clases negativas. En primer lugar, se intento reducir el espacio temporal utilizado en el conjunto de entrenamiento, sin embargo, esto empeoro aun más el modelo.

En un intento de proporcionar información más clara al algoritmo, se entrenaron nuevos modelos eliminando del conjunto de entrenamiento los 'casos grises', es decir, eliminando aquellas filas de datos cuyo diferencial de precio estuviese más próximo a 0. De esta forma, se aplicó el siguiente filtro:

$$|\Delta Precio| < \text{T}\ \%$$

Además, para buscar mejorar la especificidad, se llevó a cabo un equilibrado de las clases en el training set, eliminando las 'clases 1' sobrantes que fuesen más 'grises'. Gracias a esto, se consiguieron mejorar bastante las métricas obtenidas.

Finalmente, se pudo obtener un modelo considerado óptimo para los intereses del proyecto. A continuación, se muestra el esquema de dicho modelo:

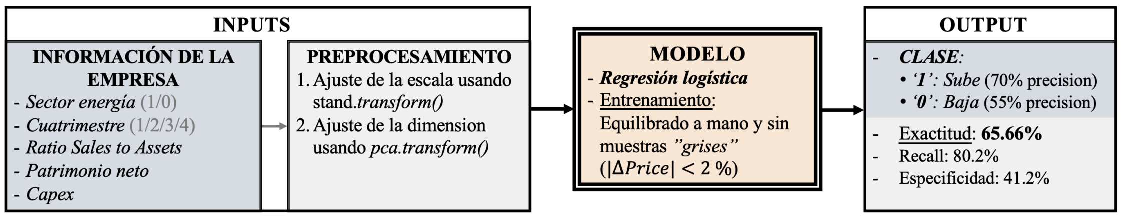

INPUTS
INFORMACIÓN DE LA EMPRESA
- Sector energía (1/0)
- Cuatrimestre (1/2/3/4)
- Ratio Sales to Assets
- Patrimonio neto
- Capex
PREPROCESAMIENTO
1. Ajuste de la escala usando stand.transform()
2. Ajuste de la dimension usando pca.transform()
MODELO
- Regresión logística
- Entrenamiento:
Equilibrado a mano y sin muestras "grises" (|ΔPrice| < 2 %)
OUTPUT
- CLASE:
• '1': Sube (70% precision)
• '0': Baja (55% precision)
- Exactitud: 65.66%
- Recall: 80.2%
- Especificidad: 41.2%

Ilustración 4 – Mejor modelo con regresión logística

En cuanto a los *K-SVM*, el proceso seguido fue el mismo. De nuevo el mejor modelo se obtuvo aplicando el equilibrado de las clases en el training set, un filtro $T = 2\%$ y una reducción de la dimensión de los datos mediante *ACP*. El esquema del mejor modelo obtenido con *K-SVM* se muestra a continuación:

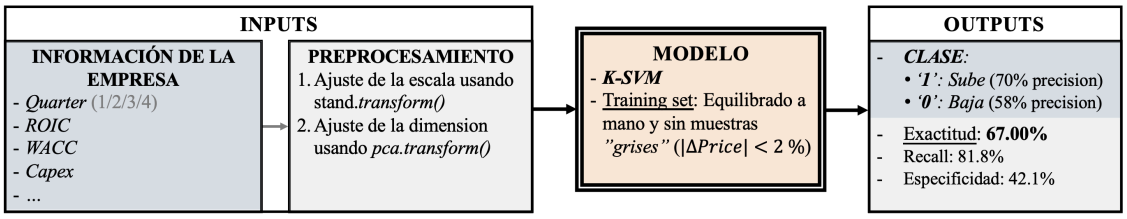

INPUTS
INFORMACIÓN DE LA EMPRESA
- Quarter (1/2/3/4)
- ROIC
- WACC
- Capex
- ...
PREPROCESAMIENTO
1. Ajuste de la escala usando stand.transform()
2. Ajuste de la dimension usando pca.transform()
MODELO
- K-SVM
- Training set: Equilibrado a mano y sin muestras "grises" (|ΔPrice| < 2 %)
OUTPUTS
- CLASE:
• '1': Sube (70% precision)
• '0': Baja (58% precision)
- Exactitud: 67.00%
- Recall: 81.8%
- Especificidad: 42.1%

Ilustración 5 – Mejor modelo con K-SVM

6. Resultados

Los dos modelos finales que se obtuvieron para cada tipo de algoritmo, demostraron mejorar las prestaciones del modelo básico del que se ha hablado anteriormente. Sus métricas de evaluación vienen recogidas en la siguiente tabla:

Modelo con *RL*		**Modelo con *K-SVM***	
Exactitud	65.7%	Exactitud	67.0%
Precisión	69.7%	Precisión	70.4%
Recall	80.2%	Recall	81.8%
F_1	74.6%	F_1	75.7%
Precisión (clase 0)	55.2%	Precisión (clase 0)	57.8%
Especificidad	41.2%	Especificidad	42.1%

Tabla 1 – Comparación de las principales métricas de los modelos

El mejor modelo obtenido fue, por tanto, el modelo entrenado con *K-SVM*. Este modelo ofrecía una exactitud del 67%, es decir, que acertaba 67 de cada 100 clasificaciones.

Conviene destacar que estos modelos pueden producir dos tipos de errores:

- Error 1: Recomendar la compra de una acción cuando luego va a caer.
- Error 2: Recomendar no comprar las acciones cuando luego van a subir.

Es evidente que el error menos deseable es el primero, por conllevar pérdidas de dinero. El segundo, sin embargo, es más tolerable, ya que solo significa una oportunidad perdida. Por lo tanto, las métricas en las que se debe basar el análisis de los resultados son la precisión y la especificidad, ya que cuanto mayor sea el valor de estas métricas, menor número de errores de clasificación del tipo 1 se cometerán.

La Ilustración 6 compara las métricas de evaluación del modelo óptimo con las del modelo básico.

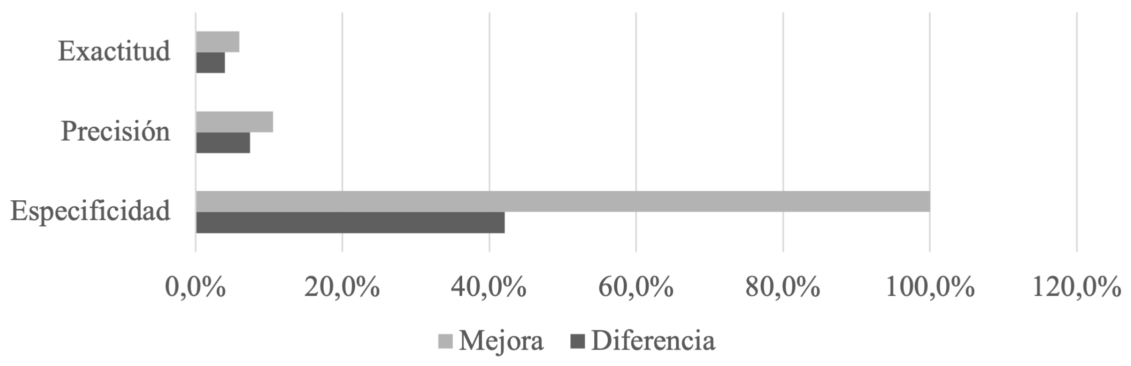


Ilustración 6 – Comparación del mejor modelo con el modelo básico

Cómo se observa el modelo que se ha logrado obtener supera al modelo básico en todos los aspectos, sobre todo en precisión de clasificación, métrica fundamental para reducir el número de errores de tipo 1.

7. Conclusiones

Al finalizar el proyecto, se logró cumplir el objetivo básico, obtener un modelo de predicción en bolsa a medio plazo que superase las prestaciones del modelo básico.

Las principales conclusiones obtenidas se recogen a continuación:

- Los algoritmos *K-SVM* han demostrado ser más eficaces a la hora de tratar los datos del problema que los de *RL*. Esto se debe seguramente a la inseparabilidad lineal de los datos con los que se ha trabajado.
- La eliminación de las clases que hemos denominado "grises" ha demostrado ser eficaz a la hora de entrenar los modelos, consiguiéndose mejoras de hasta 2 puntos porcentuales en la precisión y la especificidad.
- Los modelos entrenados con datos estandarizados demuestran tener métricas de valoración más altas con los dos algoritmos. Sin embargo, la utilización de datos sin outliers parece ser solo eficaz con los algoritmos *K-SVM*.
- Los modelos entrenados con una gran cantidad de variables tienen una elevada complejidad y no obtienen buenos resultados. Por ello, resulta fundamental seleccionar las características más adecuadas. Una técnica para hacerlo que ha demostrado ser eficaz es la selección manual en base al coeficiente de correlación de Pearson. Además, aplicar técnicas de reducción de la dimensión como la del *ACP* a estas variables ya seleccionadas mejora las características de los modelos.

8. Referencias

[1] PwC, Rao, A., & Verweij, G. (2017). *Sizing the prize. What's the real value of AI for your business and how can you capitalise?* PwC. https://www.pwc.com/gx/en/issues/analytics/assets/pwc-ai-analysis-sizing-the-prize-report.pdf.

[2] Timor, M., Dinçer, H., & Emir, S. (2012, January). *Performance comparison of artificial neural network (ANN) and support vector machines (SVM) models for the stock selection problem: An application on the Istanbul Stock Exchange (ISE) - 30 index in Turkey*. African Journal of Business Management Vol. 6(3), (pp. 1191–1198). http://www.academicjournals.org/AJBM.

[3] Murty, M. N., & Raghava, R. (2016). *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks* (SpringerBriefs in Computer Science) (1st ed. 2016 ed.). Springer.

[4] Mirjalili, V., & Raschka, S. (2020). *Python Machine Learning* (Spanish Edition) (1st ed.). Marcombo

[5] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media

SUPERVISED LEARNING CLASSIFIERS APPLIED TO THE ANALYSIS OF FUNDAMENTAL DATA OF LISTED COMPANIES. DEVELOPMENT OF PREDICTIVE MODELS

Author: Reyzabal Roig, Javier
Supervisor: Lippi, Marco
Collaborating Entity: ICAI – Universidad Pontificia Comillas

The project that is summarized below has as its main purpose the development of medium-term stock prediction models using logistic regressions and kernelized support vector machines. The work concludes with the obtention of two models, one for each classifier used, being the obtained by K-SVM the one that proved to be the most efficient, offering an accuracy of 67% and a classification precision close to 70%.

Keywords: *Machine Learning*, Supervised Learning, Logistic Regressions, Support Vector Machines, Finance, Fundamental Analysis.

1. Introducción

Nowadays, Artificial Intelligence (*AI*) is present in almost all sectors, including financial. According to a study carried out by *PwC* analysts, *AI* could increase global *GDP* by around 14% by 2030 [1]. Taking this into consideration, it is easy to understand why this has become one of the sciences that has gained the most momentum in recent years.

This project seeks to apply Machine Learning (*ML*) techniques, one of the most important fields of AI, to the fundamental analysis of companies for stock investment in the medium term. Concretely, linear and kernelized classifiers will be used in order to obtain, from a series of financial data companies belonging to the *S&P500*, a predictive model that allows classifying the shares of a company based on their expected performance on the stock market.

Basically, a binary classification task is proposed, in which the positive class '1' will encompass the companies for which a rise in the stock market is expected in the following four-month period. The two machine learning algorithms that will be used are logistic regression (*LR*) and kernelized support vector machines (*K-SVM*).

2. State of the Art

It should be noted that, obviously, there are numerous studies that seek to generate predictive models in the stock market using *ML* techniques. Most of them focus on technical analysis, that consists on analyzing the progression of a stock in the short term.

This does not imply that there are no long-term modeling projects on the stock market. This area of stock investing has also been deeply explored. For example, Timor, Dinçer and Emir, using *K-SVM* as a classifier algorithm for long-term stock selection, obtained an efficiency close to 66% [2].

3. Description of the Technologies

Within *ML*, there are multiple types of algorithms that can be used for very different purposes. This project focuses on Supervised Learning methods, specifically in *LR* and *SVM*.

These algorithms allow training classifiers by providing historical data already classified. Their training basically consists of optimizing, from said data, a discrimination function (linear or not) that allows separating the diffent classes in space.

Regarding logistic regression, it should be considered that it is a linear classification algorithm, meaning that it is useful for problems with linearly separable data [3]. This kind of classifier seeks to maximize the probability of belonging to a certain class.

On the other hand, *SVM*s are also treated as linear classifiers, however, in this project the so-called kernelized *SVM*s will be used, allowing to create non-linear separations in the data [4]. Therefore, it can be said that linear (LR) and non-linear (*K-SVM*) classifiers will be used.

4. The Database

An *ML* algorithm is capable of learning thanks to the information provided to it. Because of that, to carry out a good project, you will need a complete database that contains relevant information for classifiers.

54 companies were selected from the *S&P500* belonging to 9 different sectors, from which the information was downloaded using Bloomberg Terminal, one of the most powerful financial tools in the world. For each company, the data corresponding to a total of 58 financial variables were obtained from all the four-month periods from January 1, 2010 to December 31, 2019. Thus, it was possible to create a database that, once structured, was presented the following way:

	Row	***Industry***	***Company***	***Time (Q)***	***Price***	***ΔPrice***	***Label***	***Features***	
TRAINING	*1*	*BM*	*C1*	*[1,32]*	P^1_{1-32}	ΔP^1_{1-32}	*0/1*	...	...
	.		*C2*	*[1,32]*	P^2_{1-32}	ΔP^2_{1-32}	*0/1*	...	...
	.		*C3*	*[1,32]*	P^3_{1-32}	ΔP^3_{1-32}	*0/1*	...	...
	.		*C4*	*[1,32]*	P^4_{1-32}	ΔP^4_{1-32}	*0/1*	...	...
	.		*C5*	*[1,32]*	P^5_{1-32}	ΔP^5_{1-32}	*0/1*	...	...
	.		*C6*	*[1,32]*	P^6_{1-32}	ΔP^6_{1-32}	*0/1*	...	...
	.	*CD*	*C1*	*[1,32]*	P	ΔP	*0/1*	...	...
	.		...	...	...	...	...	...	...
	.	*CS*	*C1*	...	P	ΔP	*0/1*	...	...
	. *1728*	...	...	...	...	...	...	...	...
TEST	*1729*	*BM*	...	*[33,43]*	P	ΔP	*0/1*	...	...
	.	*CD*	...	*[33,43]*	P	ΔP	*0/1*	...	...
	.	...	...	...	...	...	...	...	...
	2322	...	...	...	...	...	...	...	...

Figure 1 – Database structure

As can be seen in Figure 1, the data was divided into two, a training set to obtain the models and a test set to test their effectiveness. Thus, the already structured database contained 2.322 rows and 58 columns, making a total of 185.760 elementary data cells.

Oppositely, taking into account that both the *LR* and *SVM* algorithms are very susceptible to scale differences in the data [4], it was decided to standardize them to later check if in this case the models improved by applying this modification to the data.

Later, in a first visual analysis of the data, it was found that the variables included a large number of outliers. Next, in Figure 2, the box-and-whisker plot of two of the variables studied is shown:

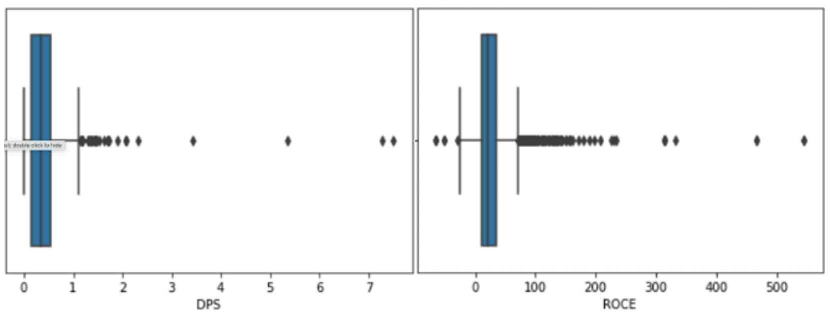


Figure 2 – Box-and-whisker plots of ROCE and DPS variables

As with the standardized data, it was then decided to create a training set from which the most extreme outliers were eliminated, those with a *Z-Score* greater than 5. In this manner, it would be checked later if using these extreme values would allow the models to be improved.

Finally, the typical data preprocessing procedures were carried out to adapt them to the algorithms that would be used. As follows, the rows with missing data were removed, the nominal variables were coded, etc.

5. Obtaining the Models

The methodology used to train the models is outlined in Figure 3:

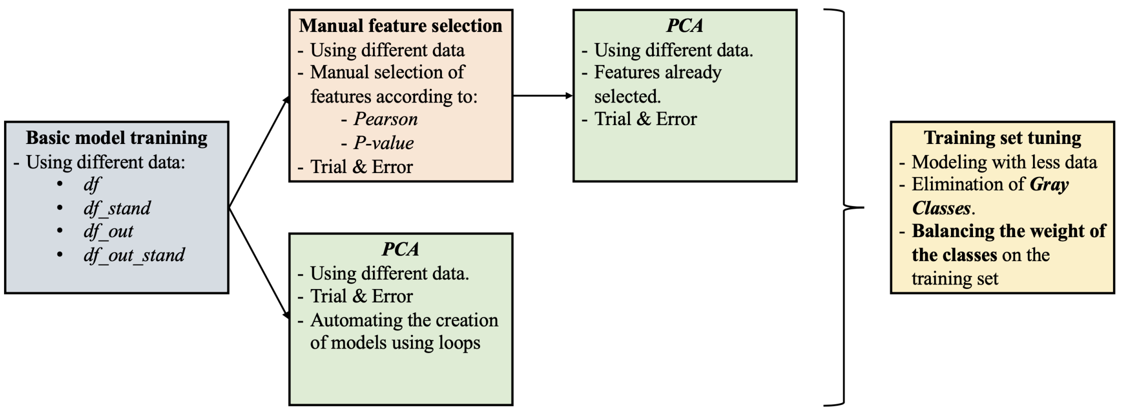

Basic model tranining
- Using different data:
df
df_stand
df_out
df_out_stand
Manual feature selection
- Using different data
- Manual selection of features according to:
- *Pearson*
- *P-value*
- Trial & Error
PCA
- Using different data.
- Features already selected.
- Trial & Error
PCA
- Using different data.
- Trial & Error
- Automating the creation of models using loops
Training set tuning
- Modeling with less data
- Elimination of ***Gray Classes***.
- **Balancing the weight of the classes** on the training set

Figure 3 - Model training methodology

Before starting to explain Figure 3, it is important to mention that there was an imbalance in the data. By this we mean that 'class 1' had more representation than 'class 0', specifically 63% compared to 37%. Because of this, the most basic decision model that could be trained was a model that would classify all samples in 'class 1'. This basic model would have a precision and accuracy of 63%. The objective that was sought when training the later models was, therefore, to exceed the performance of the latter.

The training process began with logistic regressions. As indicated in the diagram, first, a series of simple models were trained using the 58 variables from the database. When verifying the ineffectiveness of these models, a manual selection of the characteristics was carried out, which gave the best results with the standardized data set. Data dimension reduction techniques were also applied based on the Principal Components Analysis method. (*PCA*), which consists on generating new columns preserving the variance of the data [5].

The models obtained this way proved not to be optimal. In general, they had a very low specificity (less than 30%) and a precision close to 64%. For this reason, and in order to improve specificity above all, it was decided to look for a method that would allow the algorithm to better recognize negative classes. In the first place, an attempt was made to reduce the temporal space used in the training set, however, this made the model even worse. In an attempt to provide clearer information to the algorithm, new models were trained eliminating the 'gray cases' from the training set, that is, eliminating those rows of data whose price differential was closer to 0. In this way, it was applied the following filter:

$$|\Delta Price| < \mathrm{T}\,\%$$

Furthermore, to seek to improve specificity, a balancing of the classes was carried out in the training set, eliminating the remaining 'classes 1' that were 'grayer'. Thanks to this, the metrics obtained were significantly improved.

Eventually, it was possible to obtain a model considered optimal for the interests of the project. The scheme of said model is shown below:

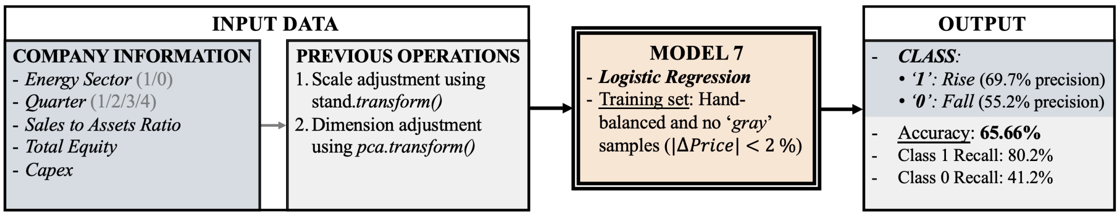


Figure 4 - Best model with logistic regression

As for the *K-SVM*, the followed process was the same. Again, the best model was obtained by applying the balance of the classes in the training set, a filter $T = 2\%$ and a reduction of the dimension of the data using *PCA*. The scheme of the best model obtained with *K-SVM* is shown below:

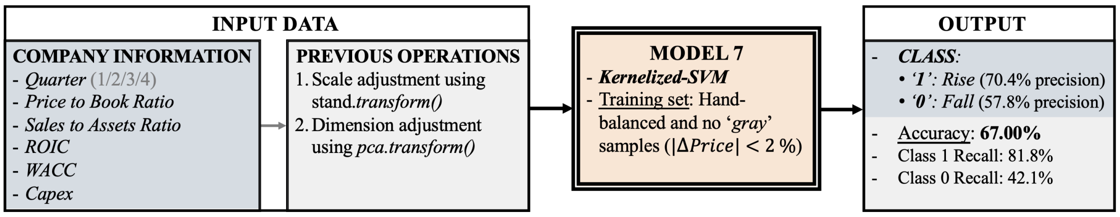


Figure 5 - Best model with K-SVM

6. Results

The two final models that were obtained for each type of algorithm, demonstrated to improve the performance of the basic model that has been discussed previously. Their evaluation metrics are collected in the following table:

Model using *LR*		Model using *K-SVM*	
Accuracy	65.7%	Accuracy	67.0%
Precision	69.7%	Precision	70.4%
Recall	80.2%	Recall	81.8%
F_1	74.6%	F_1	75.7%
Precision (class 0)	55.2%	Precision (class 0)	57.8%
Specificity	41.2%	Recall (class 0)	42.1%

Table 1 - Comparison of the main metrics of the models

The best model obtained was, therefore, the model trained with *K-SVM*. This model offered an accuracy of 67%, that is, it was correct in 67 out of 100 classifications.

It should be noted that these models can produce two types of errors:

- Error 1: Recommend the purchase of a stock when it is going to fall later.
- Error 2: Recommend not to purchase a stock when they are going to rise later.

It is obvious that the least desirable error is the first, because it entails loss of money. The second, however, is more tolerable, as it only means a missed opportunity. Therefore, the metrics on which the analysis of the results should be based are precision and specificity, since the higher the value of these metrics, the fewer type 1 misclassification errors will be made.

Figure 6 compares the evaluation metrics of the optimal model with those of the basic model.

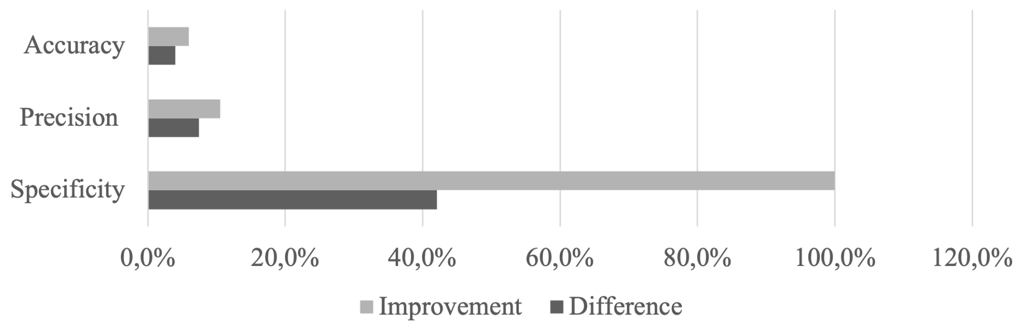


Figure 6 - Comparison of the best model to the basic model

As can be observed, the model that has been obtained exceeds the basic model in all aspects, especially in classification precision, a fundamental metric to reduce the number of type 1 errors.

7. Conclusions

At the end of the project, it was possible to meet the basic objective, to obtain a prediction model on the stock market in the medium term that exceeded the benefits of the basic model.

The most relevant conclusions are listed below:

- K-SVM algorithms have been shown to be more efficient in handling problem data than LR algorithms. This is surely due to the linear inseparability of the data with which we have worked.
- The removal of the classes that we have called “gray” has proven to be effective when training the models, achieving improvements of up to 2 percentage points in precision and specificity..
- Models trained with standardized data demonstrate higher valuation metrics with both algorithms. However, the use of data without outliers appears to be only effective with the K-SVM algorithms.
- Models trained with a large number of variables are highly complex and do not obtain good results. Therefore, it is essential to select the most suitable characteristics. One technique to do this that has proven to be effective is manual selection based on Pearson's correlation coefficient. Moreover, applying dimension reduction techniques such as the PCA to these already selected variables improves the characteristics of the models.

8. References

[1] PwC, Rao, A., & Verweij, G. (2017). *Sizing the prize. What’s the real value of AI for your business and how can you capitalise?* PwC. https://www.pwc.com/gx/en/issues/analytics/assets/pwc-ai-analysis-sizing-the-prize-report.pdf.

[2] Timor, M., Dinçer, H., & Emir, S. (2012, January). *Performance comparison of artificial neural network (ANN) and support vector machines (SVM) models for the stock selection*

problem: An application on the Istanbul Stock Exchange (ISE) - 30 index in Turkey. African Journal of Business Management Vol. 6(3), (pp. 1191–1198). http://www.academicjournals.org/AJBM.

[3] Murty, M. N., & Raghava, R. (2016). *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks* (SpringerBriefs in Computer Science) (1st ed. 2016 ed.). Springer.

[4] Mirjalili, V., & Raschka, S. (2020). *Python Machine Learning* (Spanish Edition) (1st ed.). Marcombo

[5] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media

Table of contents

Index of Figures

Index of Tables

CHAPTER 1. INTRODUCTION

Artificial intelligence has stirred the world we live in. In 1956, at a conference in Dartmouth, the term *Artificial Intelligence* (*AI*) was first introduced. Ever since, the progressions in this field have been enormous. Within 65 years we have gone from algorithms trying to defeat humans at games such as checkers, to algorithms demonstrating superior performance to humans at much more complicated and less systematic tasks [1]. Proof of this is the recent success of IBM Project Debater, which in 2019 proved to be able to debate against a champion in the field.

Nowadays, modern *AI* can be applied in almost all industries, which include healthcare, transport, safety & security, finance and many others. In fact, the impact that *AI* may have over the next decade is not precisely small. According to a study conducted by *PwC* (*PricewaterhouseCoopers*) analysts "*AI* could contribute up to $15.7 trillion to the global economy in 2030, more than the current output of China and India combined" [2]. These figures show the great importance *AI* might have in the near future, potentially increasing global GDP by around 14%. Moreover, according to the study, "the biggest potential economic uplift from *AI* is likely to come from improved productivity" [2]. As it can be observed in the figure below, approximately half of this economic growth will come from productivity improvements. Furthermore, there will also be significant improvements in time saving and in product innovation and quality.

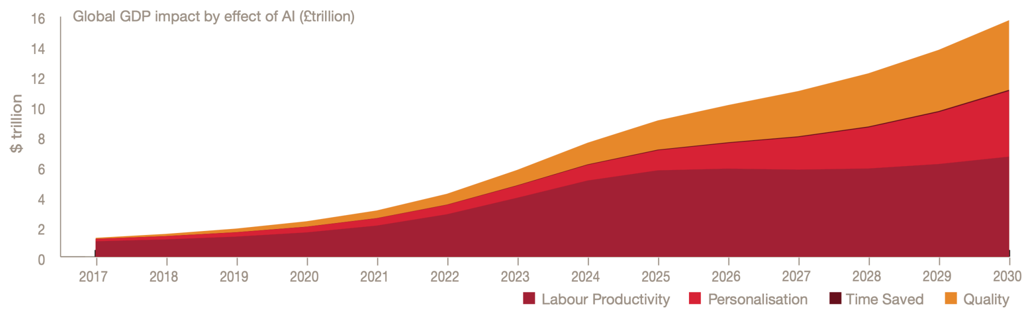


Figure 1. Where will the value gains come from with AI? (Source: PwC analysis [2])

As it can be appreciated, the economic potential of artificial intelligence over a 10-year horizon is immense. In this regard, Andrew Ng, professor at Stanford and world eminence in the field of *AI*, said during a conference at the University that "Just as electricity transformed almost everything 100 years ago, today I actually have a hard time thinking of an industry that I don't think *AI* will transform in the next several years" [3].Therefore, with *AI* being "the new electricity" [3], it is likely that, in no more than ten years, most of us will need to be aware of it, not just for our future working life, but also for life in general.

As it has been said, *AI* has a huge impact in most industries, and the financial sector is no exception. As it will be seen later in *Chapter 2*, the answer to the question of whether the finance industry has adopted *AI* and *Machine Learning* (*ML*) techniques, is yes. In fact, it has done that in many different ways, all of them motivated by the increasing of available data, only manageable by programs and models. This idea is supported by Matthew F. Dixon *et al.*, who in their book state that "the growth of machine-readable data to record and communicate activities throughout the financial system combined with persistent growth in computing power and storage capacity has significant implications for every corner of financial modeling" [4].

The reason for this project lies in the growing importance of *AI* and the incremental relevance of data in our lives. Throughout the task, we will seek to develop and compare a series of supervised learning classification algorithms, so that at the end, these algorithms will be able to predict, with reasonable accuracy, whether the value of a stock will go up or down in the medium term. For this purpose, the database used will include information from 54 companies comprehended in the *S&P500*. In order to achieve that, a fundamental analysis approach will be used. The two classifiers that will be used during the project are *Logistic Regressions* and *Support Vector Machines*. Several models of each type will be trained using more or less processed data; afterwards, the results and performance of each one of them will be compared, analyzing advantages, disadvantages, accuracy, evaluation metrics, etc.

1.1 PROJECT MOTIVATION

Today we live in a world in which traditional banks are closing their branches and completely remodeling their structure to align themselves with the *fintech philosophy*; payments are no longer made only using traditional currencies, instead, blockchain technology allows transactions using cryptocurrencies, etc. Proof of the digital transformation that the sector is undergoing is the ECB's (European Central Bank) intention to introduce a digital euro, as they showed in a press release in July 2021 [5]. ECB President Christine Lagarde said: "Our work aims to ensure that in the digital age citizens and firms continue to have access to the safest form of money, central bank money" [5]. Therefore, it can be said that even if government agencies such as the *ECB* are adapting themselves to this new *AI* era, it is because we are probably facing a paradigm shift in which, gradually, we will have to adapt ourselves.

However, *AI* has not only reached banks or currencies. Another fundamental area of application is focused on asset investment. There are algorithms of all kinds in this regard: models that quantify the risks of a given investment, models that forecast the value of real estate assets, models for fraud detection, etc. Nonetheless, this project will focus on what we call Robo-Advisors for stock investing, which are "financial advisors that provide financial advice or portfolio management services with minimal human intervention" [4].

Predicting whether a given company's stock price will rise or fall over a certain period of time has always been of great interest. Personally, my motivation to carry out this project arose during an internship in an asset management company. While working in the decision-making department, I realized how much data exists today on every traded company and how difficult it is to handle it without the assistance of computers. Therefore, researching ways to facilitate the analysis of companies, I came across *Machine Learning* and its application in these Robo-Advisors for the first time in my life.

If you are an institutional investor, you can for sure access these types of *AI*-assisted investment models and programs. However, if you are an individual investor or a student, as in my case, the access to these technologies is limited, mainly due to their high cost. It is true that Robo-Advisors can be accessed through banks or online investment platforms at low cost, but without really knowing what happens to the money invested. The investment options of this type are limited, not allowing the investor to use the program and not being able to have any kind of control in its management, so in the end, it is like investing in the traditional way. If what is sought is to be able to personally manage the Robo-Advisors, the cost skyrockets. Mainly for this reason, I consider it appropriate to carry out this project and thus contribute to the creation of open-source resources on machine learning assisted investment.

Therefore, although not reaching the same degree of sophistication as professional Robo-Advisors, the development and analysis of different models intended to facilitate decision-making in stock investment, could be of great help both for me in my learning process as a student, as well as for anyone who wants to access this work.

A second inspiration that led me to carry out this project is the possibility of providing a useful decision-making model to the student association *Comillas Investment Club* (*CIC*) to which I belong.

CHAPTER 2. DESCRIPTION OF THE TECHNOLOGIES

Once the problem to be addressed in the project has been introduced, the technologies needed to carry it out can be discussed.

The task includes two well-differentiated areas: the *fundamental analysis* (*FA*) of companies in terms of finance and *Machine Learning* (*ML*) regarding artificial intelligence and model generation. The chapter can then be divided into these two different parts. Later, in *Chapter 3,* it will be discussed how the two areas have been combined so far in the existing literature.

The objective of this section of the paper is to present the basic theoretical concepts of the two topics introduced in the previous paragraph. They will be studied in the following order:

- Initially, the basic concepts of fundamental analysis of listed companies will be presented. This analysis process allows managers to estimate the value of a company, thus being able to determine whether this value is higher than its market price.
- Subsequently, the basic concepts related to artificial intelligence and machine learning will be studied. The focus will be on supervised learning and classification algorithms.

1.2 FUNDAMENTAL ANALYSIS OF LISTED COMPANIES

It is interesting to highlight that, in the financial world, there are generally two main methods when trying to obtain wealth through the purchase and sale of equity: *investing* and *trading*. The first one tries to create wealth by identifying and investing in those companies that are considered to be undervalued by the market. That wealth creation hypothesis in investing is based on the likely alignment of price and value over the long term. On the other hand, trading aims to do that but in a much shorter period of time, usually meaning more frequent transactions.

Investing and trading can therefore be considered as two different approaches to building a stock portfolio. The main difference between them lies in the period of time the stock is intended to be held. Therefore, it is logical to think that there are also two methodologies for stock analysis:

- On one hand, for *trading*, the most widespread method is technical analysis, which consists of predicting stock prices based on their past behavior. To do so, analysts use statistical methods to try to find similarities between the company that is being analyzed and historical stock market circumstances, in the hope that a similar situation will arise in the future [6]. It could be said that, despite having no financial basis, technical analysis works because many people apply it. Generally, people who operate in this way in the stock market use the same guidelines when buying or selling stocks; this causes buying and selling volumes to vary and, as a consequence of the Law of Supply and Demand, the price.
- On the other hand, for *investing*, a fundamental analysis of the company to be included in the portfolio is usually carried out, along with industry economic cycle analysis. This type of analysis has a much more financial and less statistical approach. Its aim is to determine the target price of the share based on the fundamentals provided by macroeconomics, microeconomics, business strategy, accounting, and the analysis of stock market ratios [6].

Based on this, many professional investors, when assessing whether to invest or not in a certain company, use both methods. A first fundamental analysis allows them to try and understand if the company is undervalued or overvalued, followed then by a technical analysis that determines whether it is the right time to buy the stock. In this way, for example, if after the fundamental analysis the investor concludes that the company may be undervalued, but the technical analysis does not support the purchase in the short period, perhaps the investor should wait to incorporate the company into the portfolio.

The project links finance with machine learning. For this reason, it focuses on investing techniques and fundamental analysis. As it will be seen in the following chapters, the algorithms developed will be based on present fundamental indicators of the selected companies, without using technical indicators or future performance estimates.

1.2.1 Stock Analysis

Fundamental analysis is only one part of the company analysis procedure. In general, this process is much broader, consisting, as shown in the image below, of three main phases: economic analysis, industry analysis and fundamental analysis. Following the order indicated in the image, a top-down analysis would be applied, i.e., an analysis that starts with the overall picture and ends with the details of the company. A bottom-up analysis is also feasible, with an inverted order; however, it is not as common as the top-down [7].

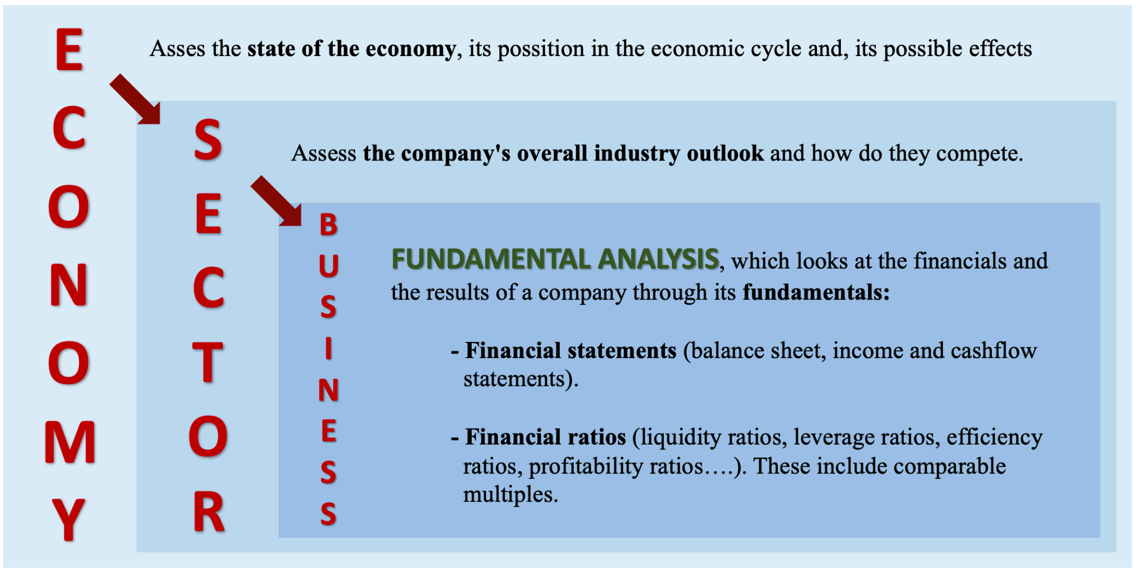

ECONOMY
Asses the **state of the economy**, its possition in the economic cycle and, its possible effects
SECTOR
Assess **the company's overall industry outlook** and how do they compete.
BUSINESS
FUNDAMENTAL ANALYSIS, which looks at the financials and the results of a company through its **fundamentals:**
- **Financial statements** (balance sheet, income and cashflow statements).
- **Financial ratios** (liquidity ratios, leverage ratios, efficiency ratios, profitability ratios….). These include comparable multiples.

Figure 2. Top-down company analysis process

Generally, to arrive at the value of a company, fundamental analysis techniques such as financial ratio analysis are usually combined with intrinsic valuation models based on fundamentals, such as the *discounted cash flow* modeling (*DCF*). *DCF* models assign an intrinsic value to the company by assessing the expected future performance of the business and valuing at the present day, i.e., discounting a company's future cash flows in a single figure [8]. Yet, as these models are based on predictions of the future performance of the company, they are not of interest to the project, which will focus exclusively on current fundamentals.

Figure 2 shows the three phases that should make up any business analysis. By following these steps, it should be possible to obtain an accurate representation of the company to help decide whether it is undervalued. As it has been said, although the first two steps are crucial to carry out a good analysis, this project will focus on the analysis of the indicators and ratios used in fundamental analysis.

1.2.2 Classification of the Financial Variables Used in the Project

The fundamental analysis of a stock draws its conclusions from the study of financial statements, expansion plans, sales, future expectations of the company, etc. [6]. Therefore, the information can be divided into two classes: qualitative or quantitative information. Qualitative information would be, for example, the sector, the quality of the management team, the goodwill, the ESG criteria used... On the other side, quantitative information includes everything related to the performance of the company: values of the financial statements (EBITDA, sales...), ratios that emerge from them (price to earnings ratio, financial leverage, price to sales ratio, EBITDA margin...). The project will use all types of information, although mainly quantitative.

According to its purpose, during the project, data will be classified into different categories, as shown in the following table.

Data class name		*Description*	*Examples*
Stock information data		Information related to the company's shares	*Book value, market capital, eps…*
Financial statement data		Information obtained directly from the financial statements	*EBITDA, Assets, CAPEX…*
Ratios	***Leverage ratios***	They focus on the company's financial structure and indicate the amount of debt being used to support the operations [7]	*Debt-equity ratio, free cash flow yield, leverage…*
	Liquidity ratios	They measure the company's ability to meet operating expenses as they fall due [7]	*Current ratio* and *quick ratio*
	Profitability ratios	Relate the returns of a company to its sales, assets, or equity. It can be an indicator of success [7]	*ROE, ROIC, EBITDA margin, gross margin…*
Comparable multiples		Multiples used to compare a company with others, generally in the same sector.	*Price to earnings, EV/EBITDA, price to sales…*
WACC ratios		Information and ratios related to the company's WACC	*WACC, CoE to WACC ratio…*
Other ratios		Ratios that do not fit into the above categories	*Working capital to sales ratio…*

Figure 3. Classification of the financial variables used in the project

All the financial variables used in the project are classified in one or another category depending on their use; for example, if a certain variable is used to determine whether the company will be able to meet its short-term payments, it will be classified as a *liquidity ratio*.

Every financial fundamental that will make up the database used to train the models is listed in *Appendix A*.

1.2.2.1 The WACC

Many types of variables will be used. These range from typical ratios such as price to earnings (*PER*) to others used in different circumstances such as *WACC* and derivative ratios.

The *WACC* (weighted average cost of capital) is the value at which cash flows are generally discounted, for example, in *DCF* models [9]. As this type of modelling methods for calculating the intrinsic value of a share by discounting its future cash flows is not the objective of this project, the $WACC$ will not be described in depth. Only the basic notions will be explained, so that one can clearly understand what it is and what it is used for in valuation. For this reason, this paper will not explain the theory behind the calculation of this value, but its usefulness.

The $WACC$ is an essential value for all those who invest in a company, including shareholders. Its value represents the minimum return that the company for which it is calculated must earn to satisfy its investors. It can be said that this value is typically an indicator of the extra risk that investors take on when entering a company. The higher the $WACC$ is, the higher the minimum required to return by the investor will be, who will thus try to neutralize the risk assumed [9].

Its calculation comes from the sum of the cost of the company's debt and the cost of its equity. The general formula for calculating the $WACC$ is presented below [8].

$$\text{E. 1} \qquad WACC = \frac{D}{V} k_d (1 - T_m) + \frac{E}{V} k_e$$

Where:

D is the value of the company's debt. Includes both long-term and short-term debt.

E is the value of the company's equity.

V is the sum of the market value of the company's debt and equity ($V = E + D$).

k_d is the cost of the company's debt. It can be considered as the effective interest rate at which a company pays its debt.

k_e is the cost of the company's equity. It can be considered as the cost assumed to finance the company with equity (shares).

T_m is the corporate income tax rate.

It is important to mention that in the WACC formula, $\frac{D}{V}$ and $\frac{E}{V}$ denote respectively the percentage of debt and equity respectively. Thus, the sum of both values should be 1.

All the variables involved in the equation *E. 1*can be obtained directly using the company's financial statements except for the cost of equity (k_e) and the cost of debt (k_d). Both can be estimated using the Capital Asset Pricing Model (*CAPM*) as shown in A This will be the method used in the project. Nevertheless, it should be noted that there are other ways of estimating these two values, such as, for example, through the *Dividend Capitalization Model* [9].

1.3 MACHINE LEARNING AND SUPERVISED LEARNING

1.3.1 ARTIFICIAL INTELLIGENCE

The other key area of the project is *Artificial Intelligence* (*AI*) and *Machine Learning* (*ML*).

Elaine A. Rich defined *AI* in 1983 as "the study of how to make computers do things at which, at the moment, people are better" [10]. Although many *AI* computers are already capable of outperforming humans when performing certain tasks, this definition is still true, since in many other fields people are and will continue to be better than machines. Therefore, according to Rich's definition, these are the tasks that *AI* should address [1].

This project fits perfectly with the above definition. Currently, most of the process of analysing companies is carried out by humans; in this sense, *AI* is generally used only to support the decision-making process.

AI is a very broad subject of great interest in the modern world. As it can be seen in the figure below, this science addresses many different areas in which there are challenges or tasks suitable for artificial intelligence. For example, in the *Computer Vision* area the goal is "to obtain visual understanding of the world" [11].

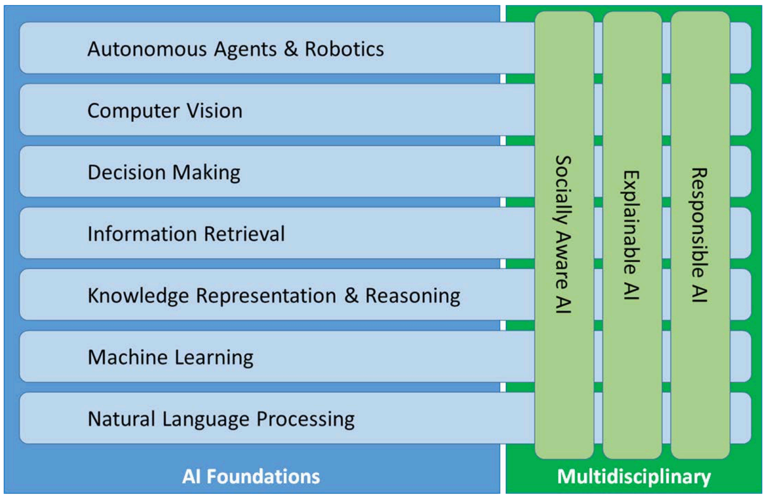


Figure 4. AI foundational areas (Source: Dutch Artificial Intelligence Manifesto [11])

The areas shown in the image above have been identified in the *Dutch Artificial Intelligence Manifesto*, developed by *The Special Interest Group of AI* (*SIGAI*). As it can be observed, there are 7 different areas, among which the ML is, on which this research project is focused.

1.3.2 MACHINE LEARNING

The *SIGAI* defines *ML* as the process of learning from data, using statistical techniques and/or neuronal networks [11]. Therefore, we can say that the purpose of *ML* is to learn from past situations (using data) to try to predict what will happen in the future or to classify or evaluate something in the present.

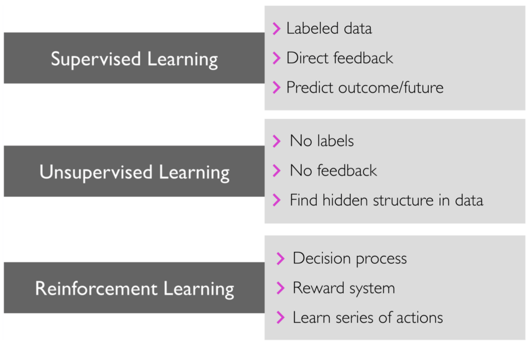


Figure 5. Types of Machine Learning (Source: Book [12])

The image above shows how many different types of *ML* algorithms exist. These fall into one class or another depending on factors such as the type of data available, whether they are labelled or not, the type of problem to be addressed, etc.

1.3.3 Classification Tasks in Supervised Learning. The Gradient Descent Method

What this project aims to do is to classify companies into groups depending on whether the algorithm expects the share price to increase or decrease in the medium term. In other words, it is a future prediction activity. For this reason, and according to the classification shown in Figure 5, *Supervised Learning* (*SL*) can be used.

The main characteristic of this type of *ML* is that it always requires labelled data. For example, training an algorithm to classify emails as "spam" or "non-spam" requires already classified emails from which the model will be further developed. Knowing this, it is easy to associate the word "supervised" to the fact that the output signals of the algorithm are already known [12].

As they require already classified data, the type of problems for which supervised learning can be applied is restricted to two [12]:

- Classification problems, which result in the inclusion of the sample in a certain discrete class. An example would be the mail classifier discussed above. These algorithms are generally referred to as *classifiers*.
- Regression problems, where, instead of falling into different classes, the result is a continuous value, just like the label. A case could be an algorithm that assesses the market value of a real estate asset.

Classifiers are used in all fields, from medicine to finance. All of them use as input a vector of unseen data $\bar{x}_{kj}$ called *vector of characteristics*. The result is a value $\bar{y}_k$ that corresponds to the class in which the sample falls.

The objective of training this type of model is to find the relationship between a *matrix of features* $X \in \mathbb{R}^{ixj}$ (which includes all the exogenous variables that may or may not explain the future model) and that of targets $Y \in \mathbb{R}^{ix1}$ (which includes all the classes to which each observation belongs) [12]. Thus, for this type of algorithm, the data set used for training and validation must follow the following structure:

- Matrix of features: $X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1j} \\ x_{21} & x_{22} & \cdots & x_{2j} \\ x_{31} & x_{32} & \cdots & x_{3j} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ij} \end{bmatrix} \in \mathbb{R}^{ixj}$
- Matrix of targets: $Y = \begin{bmatrix} y_{11} \\ y_{21} \\ y_{31} \\ \vdots \\ y_{i1} \end{bmatrix} \in \mathbb{R}^{ix1}$

In the end, a classification task can be reduced to finding a relationship between these two matrices.

A *classifier* can be defined as the non-linear algorithm A that relates X to Y, such that a prediction (y_k) can be made applying A to a sample vector of characteristics (x_{kj}) [4]:

$$y_k = A(x_{kj})$$

Algorithm A poses the following transformation [4]:

$$A\colon\ X \rightarrow G, \text{where } G \in \mathcal{M} \coloneqq \{0, 1, \dots, 1 - k\}, \text{being } k \text{ the number of classes}$$

There are many types of classifiers. The most commonly used in research are: *K-Nearest Neighbour* classifier, *Logistic Regressions*, *Decision Trees*, *Random Forests* and *Support Vector Machines* (*SVMs*). All of them are based on statistical techniques, therefore, both the matrix of features (X) and the matrix of targets (Y) must be purely numerical. This does not mean that qualitative data cannot be used. There are several techniques that allow us to adapt this type of data to numerical values, such as *one-hot encoding*. As we will see in *Chapter*

5, these strategies are usually implemented in the data pre-processing, before training the models.

When generating the models, a fairly generalized process is usually followed. This process can be divided in 5 steps [12]:

1. Feature selection and obtainment of a complete database.
2. Selection of a classification algorithm and optimization of the classifier.
3. Data pre-processing.
4. Training of the algorithm using the training set.
5. Performance evaluation using the test set.

These steps will be explained one by one during the following chapters.

Raschka and Mirjalili simplify these stages very well in their book. They propose to use the following workflow to train the classifiers:

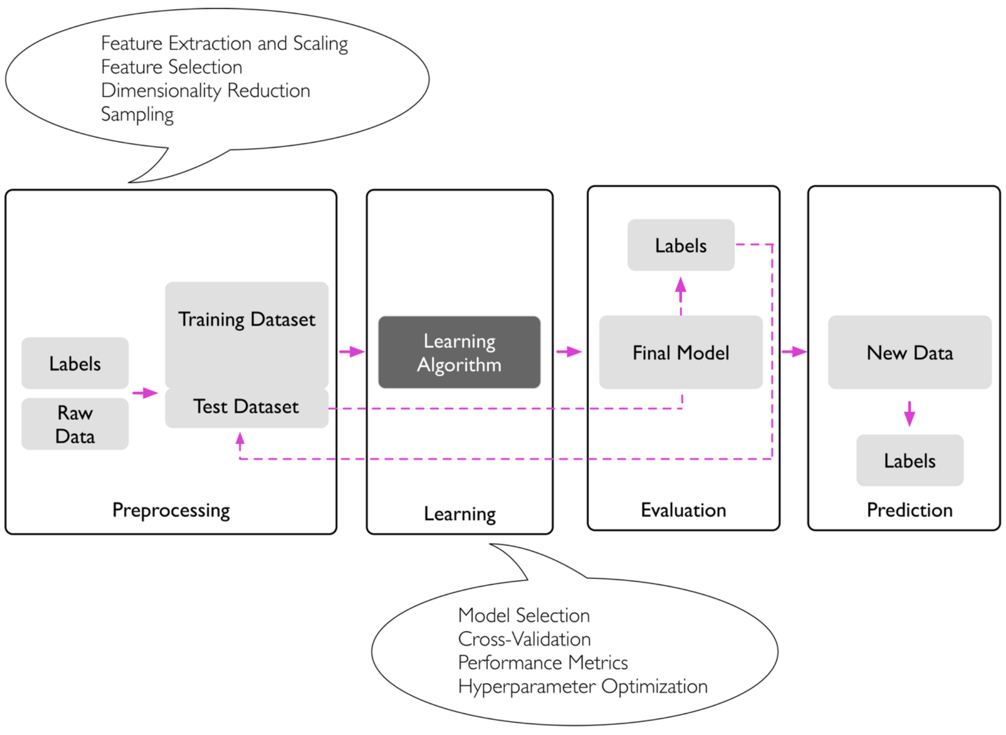


Figure 6. Classifier training workflow (Source: Book [12])

1.3.3.1 Linear Classifiers and Optimization Algorithms

So far, the fundamental characteristics of the classification algorithms have been presented. Now, two simple classifiers will be introduced: the *Perceptron* and the *Adaptive Linear Neuron* (*Adaline*). Understanding the theory behind these algorithms provides the basis for understanding more complicated algorithms later on.

These two algorithms are linear classifying algorithms. This means that they divide the space using hyperplanes, as shown in the figure below [13].

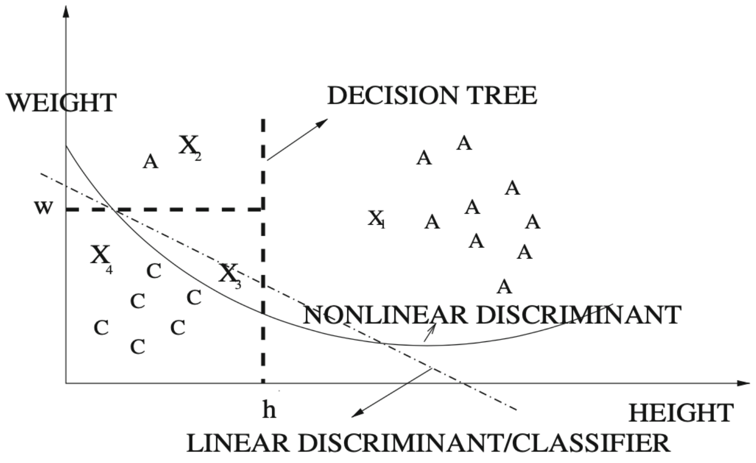


Figure 7. Linear and Non-Linear Classifiers (Source: Book [14])

The figure shows a two-dimensional space, where the hyperplane is a straight line, but, obviously, this can be extrapolated to higher dimensions.

A linear classifier is, then, characterized by a *linear discriminant function* [13]. This function, g, can be expressed as follows:

$$g(\bar{x}) = \bar{w}^T \bar{x} + b, where\ \bar{w}, \bar{x} \in \mathbb{R}^l\ and\ b \in \mathbb{R}, being\ l\ the\ dimension\ of\ the\ space$$

This equation can be rewritten as follows:

$$E.\ 2 \qquad g(\bar{x}) = z = b + \sum_{j=1}^{l} w_j\, x_j = b + w_1 x_1 + \cdots + w_l x_l$$

For this type of algorithm, the function $g(\bar{x})$ characterizes the decision boundary, i.e., the straight line (in two dimensions) shown in Figure 7 [13].

All linear learning algorithms for binary classification work in the same way. They learn a linear discrimination function $\hat{g}(\bar{x})$ or $\hat{z}$ such as the one in equation *E. 2*. commonly using iterative optimization algorithms such as gradient descent, which will be discussed later [13]. This process can be summarized in three steps:

1. Generally, to learn $g(\bar{x})$, the weights are initialized with small random numbers. Once this is done, the linear discrimination function is evaluated using an *activation function* $\phi(z)$. This function $\phi(z)$, provides reference values that, introduced in a previously defined *error function J(w)*, serve as feedback to the model to optimize or not the weights, depending on the difference between the predicted class values and the real ones. This process is repeated iteratively as many times as necessary Once an acceptable error has been reached, the linear discrimination function will no longer be updated.
2. With the previous step, a learned linear discrimination function for the model, $\hat{g}(\bar{x})$, is obtained. This, again, goes through the activation function, but, producing an already acceptable error, does not give feedback to the model to re-optimize the weights. It is at this point when the model is trained and can be used to classify samples.
3. To classify a sample x^n, the algorithm must first evaluate $\phi\left(\hat{g}(\bar{\bar{x}}^n)\right)$. As the error function will no longer come into play, the above value is evaluated directly with a *threshold function*. Depending on the value obtained for the sample with the linear discrimination function, the threshold function decides whether the sample belongs to one class or the other. [12]. An example of binary classification could be:

$$y = \begin{cases} 1 \; if \; \phi\left(\hat{g}(x^n)\right) > k \\ 0 \; if \; \phi\left(\hat{g}(x^n)\right) \leq k \end{cases}, \text{where } z = \hat{g}(\bar{x})$$

Everything explained in the previous steps is schematized in the figure below.

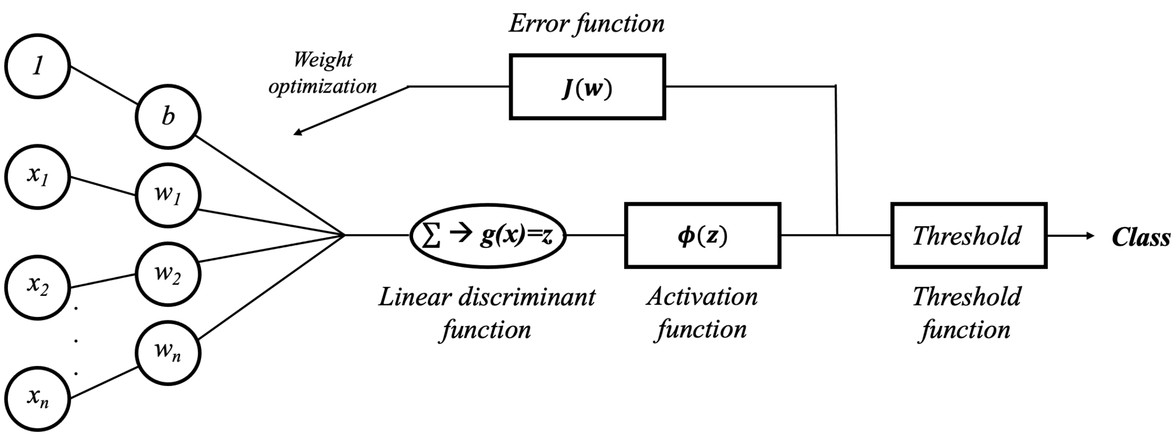


Figure 8. Scheme of a Linear Classification Algorithm

The Perceptron

The perceptron is considered to be the basic linear classifier. The algorithm learns a linear discriminant function such as the one introduced in equation *E. 2*. Specifically, it learns the weight values (w_j and b, which can be considered the value of w_j for $j = 0$ and $x_{j=0} = 1$).

To obtain the function $\hat{g}(\bar{x})$, a second, usually iterative, algorithm, called the *classifier optimization algorithm*, must be applied. In the case of the perceptron, the *perceptron learning rule* is applied. For this purpose, the following steps are implemented [12]:

1. The weights (w_j and b) are initialized to random values close to zero.
2. For each training sample $\bar{x_i} \in \mathbb{R}^{1xl}$, where $i \in \{1, \dots, m\}$, being m the number of samples:
   a. Calculate the predicted class label $\hat{y}_i$.
   b. If the predicted value is not as expected, he weights (w_j and b) should be updated. Then step 2 is implemented again.

The update of the weights, in case of failed prediction, works as follows:

$$E.\ 3 \qquad w_j \coloneqq w_j + \Delta w_j$$

To calculate the increase in weights (Δw_j), the following formula is usually used:

$$E.\ 4 \qquad \Delta w_j = \eta(y_i - \hat{y_i})x_{ij}$$

According to equation E. 4, the weights are only updated if the predicted label is incorrect.

In this equation, η is a *hyperparameter*. These are not normal parameters, but their values are chosen before learning the model and are not part of it. Hyperparameter tuning is a fundamental aspect of classifier training. In this case η is called *learning rate* and is a value generally between 0 and 1. The higher the learning rate is, the bigger the weight update will be and the faster the sample will be labeled correctly. However, the convergence of the algorithm is only guaranteed for sufficiently small values of η and if the classes are linearly separable [12].

The image below shows a block diagram representing the general perceptron concept. This image has been taken from the *Python Machine Learning* book [12].

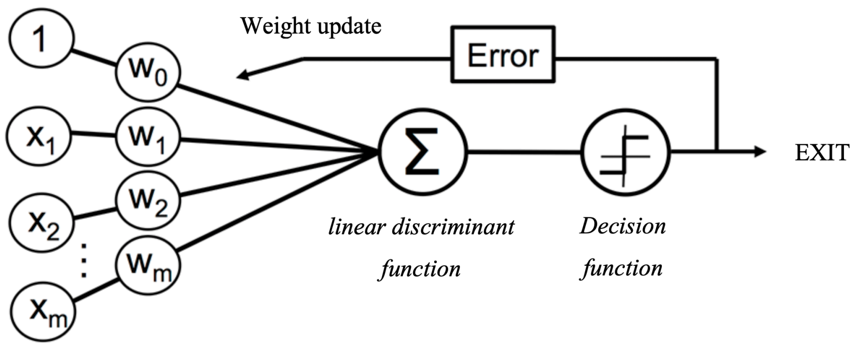


Figure 9. The Perceptron (Source: Book [12])

As can be appreciated in this last figure, the activation function in the perceptron is as follows:

$$\phi(z) = \begin{cases} 1\ if\ z \geq 0 \\ -1\ otherwise \end{cases}$$

By subsequently evaluating the learned linear discriminant function $\hat{z}$ or $\hat{g}(\bar{x})$ for our sample $\bar{x}$ at $\phi(z)$, the sample can be classified according to:

$$y = \begin{cases} 1\ if\ \phi(z) \geq 0 \\ 0\ if\ \phi(z) < 0 \end{cases}, \text{where } z = \hat{g}(\bar{x})$$

For this algorithm it is not necessary to use a threshold function, since the activation function itself, being a unit step, serves to classify the samples.

Adaptive Linear Neuron (Adaline)

This algorithm lays the foundation for more complex algorithms such as logistic regression or SVM. The structure is very similar to that of the perceptron, but in this case, the weights are updated on the basis of a linear decision function$\phi(z)$and not on the basis of a unit step as was the case with the perceptron [12]. The new activation function is then:

$$\phi(z) = \overline{w}^T \overline{x} + b$$

In this way, the *Adaline* algorithm compares the real labels with the continuous values provided by this new decision function to calculate an error [12]. Most classifying algorithms define an *objective function* to be optimized. In this case, it is the *Sum of Squared Errors* (*SEE*) which is defined as:

$$J(w) = \frac{1}{2}(y_i - \phi(z_i))^2$$

This *SEE* is the *error function* and must be minimized.

For the perceptron, as the error was not continuous but discrete in the form of a unit step, there was no need to implement any complex optimization algorithm. Now that the objective function is more complicated, more powerful algorithms are required to deal with its optimization.

Minimizing cost functions with gradient descent

The main advantage of this linear decision function is that the new cost function $J(w)$ is now differentiable. This fact allows us to apply powerful algorithms to minimize the error, such as the *gradient descent algorithm* (since J is differentiable, its gradient can be calculated). This optimization algorithm is based on taking small steps in the opposite direction of the gradient until the local minimum is reached [12]. The following figure schematizes the process:

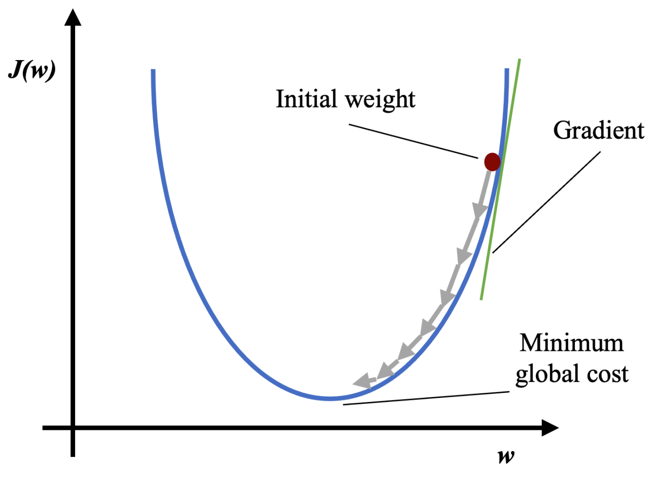


Figure 10. The Gradient Descent Algorithm

Again, the weights are updated using the formula *E. 3*.

However, the increments of the weights (Δw_j) are calculated this time using the gradient:

$$E.\ 5 \qquad \Delta w_j = -\eta \nabla J(w) = -\eta \frac{\delta J}{\delta w_j} = -\eta \sum_i \big(y_i - \phi(z_i)\big) x_{ij}$$

Using this method, the update of the weights can be calculated on the basis of all samples rather than on a sample-by-sample basis [12].

Using this method, the update of the weights can be calculated based on all samples rather than on a sample-by-sample basis. However, the algorithm no longer converges directly as before. In this case, a second hyperparameter, the number of iterations or number of epochs, must be defined (*epochs*). If we assign a very low value to this hyperparameter, it is likely that the algorithm will not converge; if the value is very high, for a large amount of data, it may require a lot of time to converge. Therefore, it is again important to assign a good value to it in order to find the balance between accuracy and cost.

As mentioned at the beginning of this section, by understanding these two algorithms, the rest can be easily understood. Therefore, the theory of the other classifiers will be introduced as they emerge in the project.

1.3.4 OVERFITTING AND L2 REGULARIZATION

One of the main objectives when modeling a classification algorithm is to achieve a good fit between the model and the data. If this is not achieved, it is possible to fall into one of the most dangerous traps of ML, *overfitting*.

Overfitting is a problem that generally occurs when the degree of complexity of the models is very high. Classifiers with this problem generally lose the ability to generalize for unseen data, i.e., they predict the training set well, but fail to predict the test set properly [23]. The following graph shows the trade-off between the complexity of the model and its ability to generalize.

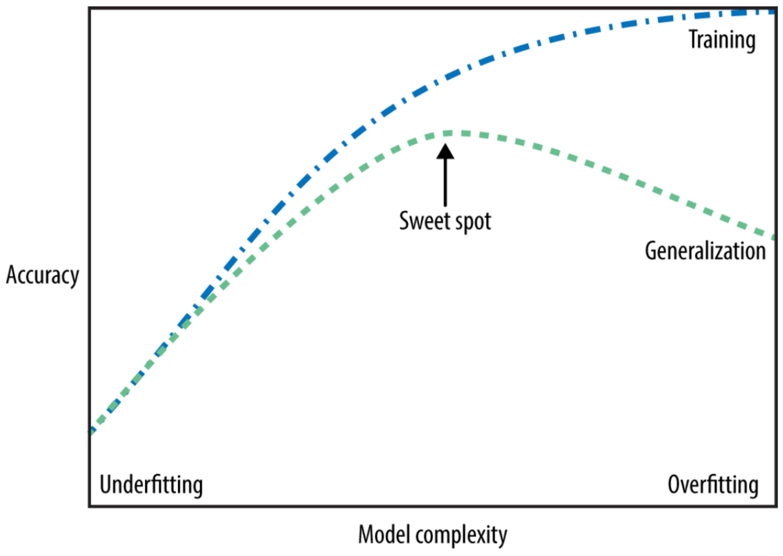


Figure 11. Model Complexity Against Accuracy (Source: Book [23])

Overfitting is especially strong when there are many potential outliers. As we will see below, the data that will be used presents many these values. It is therefore important to have a technique to avoid this problem. The method to be used is the regularization technique, which makes it possible to find the *Sweet Spot* shown in Figure 11.

"Regularization means explicitly restricting a model to avoid overfitting" [23]. This is usually done by adding a term to the cost function (J). The purpose of this new term is to present additional information so that the weight of the most extreme values is penalized. There are two types of regularization. The most commonly used is the so-called *L2 regularization*, which is applied by adding the following term to the cost function [12]:

$$L2_{regularization}term\colon \frac{\lambda}{2}\sum_{j=1}^{l} w_j^2 \,, \forall\, \lambda = \frac{1}{c}$$

It should be noted that the regularization parameter used in Scikit-learn is c. As c increases, the regularization of the model will decrease.

CHAPTER 3. **STATE OF THE ART**

Regarding the influence of *AI* on the stock picking process, it should be said that the development of Machine Learning techniques in recent years has been focused mainly on trading and technical analysis. This does not imply that algorithms and predictive models based on fundamental analysis do not exist. Of course, they do, especially at an institutional level.

Literature regarding high-frequency trading *ML* algorithms is vast nowadays. These algorithms can assess almost any trading-problem to be solved or optimized, which go from optimizing the time a stock must be held on the portfolio, to forecasting the tendency of a stock to rise or fall in the short term [14]. Furthermore, there are different approaches to this, which include a wide variety of techniques and diverse ML algorithms to address these problems.

Nonetheless, the literature is not as extensive as that dedicated to technical analysis when it comes to fundamental analysis. In this case, it focuses mainly on predictive models which based on certain financial indicators and ratios, seek to forecast the future behavior of the stock. As most of these algorithms are institutional, access to them by non-professional investors (as in the case) is quite limited if not impossible.

As mentioned in *Chapter 1*, there is an intermediate solution between technical analysis and fundamental analysis, Robo-Advisors (*RAs*). These are complex automated investment programs that usually combine all kinds of artificial intelligence and financial techniques. *RAs* have been extensively studied in recent years. For instance, Alexander Rühr studies in his paper the *performance-control dilemma* that exists in this new investment stream [15]. However, the literature on the analysis of their performance and operation is very scarce, mainly due to the interest of the proprietary companies in not making their algorithms public.

Focusing on much smaller scale investment decision algorithms, specifically on classification algorithms such as the ones this project intends to develop, it should be emphasized that there are other similar, but not identical, existing papers.

Nikola Milosevic, in his essay "Equity forecast: Predicting long term stock price movement using machine learning", studies how to classify stocks according to the expected price movement in the long term (one year). Using several classification algorithms, Milosevic manages to obtain accuracies of over 60%; in particular, he achieves an accuracy of 64% with logistic regressions and 75% with random forests [16]. In a similar study carried out by Manish Kumar, comparable results were obtained, resulting in an accuracy of 59.6% for his model with logistic regression, 67.4% using random forests and 68.44% with support vector machines (*SVM*) [17].

In contrast, Timor, Dinçer and Emir, using *SVMs* as a classifier algorithm for stock selection, obtained performances close to 66% [18].

Finally, in the paper "Prediction of Stock Market Index Movement by Ten Data Mining Techniques" carried out by Phichhang Ou, it is said that the experimental results of the research show that *SVMs* classifiers are the most suitable for this type of task [19].

In addition to classification algorithms, there are many other essays that study the application of different *ML* technologies to other areas of finance, such as risk quantification or the valuation of real estate assets.

CHAPTER 4. **PROJECT DEFINITION**

As already introduced in *Chapter 1. Chapter* **1.** Introduction, this project will seek to develop, through the most widely used Supervised Learning classification algorithms, models that help with the selection of 54 *S&P500* listed stocks in which to invest. There are an infinite number of classifiers, but this work focuses on will focus on linear and kernelized classifiers, specifically in *Logistic Regressions* and *Kernelized Support Vector Machines* (K-*SVM*).

1.4 JUSTIFICATION

In view of the previous chapter, it is clear that the task of this project has already been investigated by several authors. However, neither the databases nor the source codes used in those papers are available. Since only the results obtained in those essays and their conclusions are accessible, it is impossible to use the models, even for studying purposes.

As there is no accessible model (source code) that, by means of Machine Learning techniques, allows recognizing the stocks with the highest revaluation potential, this project will attempt to develop one and make it available to anyone interested in it.

1.5 OBJECTIVES

The main goal of this project is to create a predictive model that, based on a series of fundamental data about a company belonging to the *S&P500*, allows the company to be classified according to its future performance. With this always in mind, the following objectives will be pursued:

- Using the main linear and kernelized classifiers, a series of models will be obtained. These should make it possible to classify the shares of a series of companies according to whether or not they will rise in the following quarter.

- At least one model that outperforms another model that classifies all shares in a single class must be obtained.
- For each model obtained, its optimization will be sought using hyperparameter tuning techniques.
- The idea is to create a single script in which all the source code is contained and from which any changes can be made.
- In order to obtain good models it is essential to have good data. Therefore, an attempt will be made to create a good database containing sufficient information. This database will be preprocessed and modified so that the models can be trained correctly.
- The development of a suitable methodology to obtain models of this type will be sought.

1.6 WORK METHODOLOGY

This project is mainly divided into two sections, both essential to the goals of this project:

1. Data collection and preprocessing
2. Model training and analysis.

The first of them lays the foundations for later being able to obtain good models. In this section the aim is to generate a good database. A first analysis of the intended purpose will be carried out, and then, based on this analysis, the desired data will be downloaded from *Bloomberg*. Once the data has been structured in an Excel sheet, the labeling of the data according to stock market performance and the calculation of derived variables will begin. Afterwards, preprocessing and cleaning of the data will be performed (missing values will be treated, nominal variables will be coded, etc). In addition, a detailed study of outliers and their treatment will be made. To conclude this first part, the data frames to be used in the next stage (training sets and test sets) will be prepared and their composition will be analyzed.

The second part consists of developing predictive models on the basis of the previous data. For this purpose, a trial-and-error based methodology will be applied. The models will be trained, optimized and their predictive capacity will be assessed together with their evaluation metrics. The intention is to create several models, so that their performance will improve until the best one is obtained.

1.7 ALIGNMENT WITH THE SUSTAINABLE DEVELOPMENT GOALS

This project is fundamentally aligned with the eighth *Sustainable Development Goal* (*SDG*): "Decent work and economic growth" [20]. As an instrument dedicated to facilitating investment, a priori, it would require less knowledge on the part of the investor, which could facilitate the arrival of new non-professional investors, thus injecting (on a small scale) more money into the stock market system. This would have a positive impact on the economy, allowing companies to develop projects, creating jobs...

On the other hand, it also supports *SDG* goal number 9: "Industry, innovation and infrastructure". [20]. The incorporation of *AI* in finance is aimed at modernizing the sector and foster innovation.

It can also be said that the project is aligned with the tenth *SDG* goal: "Reduced inequalities" [20], in the sense that, since less financial knowledge is required, more people will be able to access the financial system and obtain income from it.

1.8 RESOURCES USED

To carry out the project, two main platforms will be used.

The first one is *Bloomberg Terminal*, which is one of the world's most recognized financial data platforms. It will be used for the configuration of the database. The other platform is *Jupiter Notebook*, which will be used to create a *Python* script to work with. The library that will be used for model development will be *Scikit-Learn*.

CHAPTER 5. THE DATA BASE

ML is the process of learning from data [11]. Therefore, having a good database is essential to obtain accurate and precise models.

Moreover, it is important that this data is presented in a suitable form so that the algorithms can process it. This can be achieved with data preprocessing techniques.

1.9 THE DATABASE

The task of predicting whether a stock will rise during the next quarter is very broad. To simplify it a bit, we will only work with 54 selected companies from the *S&P500* index.

1.9.1 COMPANY SELECTION

The *S&P500* is considered the most representative index of the stock market. It includes 500 of the most important companies in the USA, belonging to 11 different sectors.

As mentioned above, to facilitate the creation of the database, only 54 companies have been selected. The process followed to select them was:

1. Newer companies (those that went public after 2010) were discarded to avoid missing data.
2. The companies with the largest market capitalization were prioritized. These tend to be the most studied and therefore tend to have the most available data.
3. Companies in the *financial* and *real estate* sectors were excluded. For these sectors, different valuation methods and fundamentals are often used, which could distort the model.
4. Six companies from each of the remaining nine sectors were selected (ensuring proportionality in the data). Not all data for all companies is available, so, through a

visual inspection and prioritizing the companies with the highest data availability, the 54 companies were finally selected.

The following table shows the ticker of the 54 selected companies and the sectors they belong to.

SECTOR	CODE	COMPANIES
Basic Materials	*BM*	*ECL, FCX, NEM, FAST, IP, CE*
Consumer Discretionary	*CD*	*AMZN, BKNG, NFLX, MCD, SBUX, COST*
Consumer Staples	*CS*	*PG, CVS, MDLZ, MCK, PEP, KMB*
Energy	*E*	*XOM, CVX, SLB, WMB, HES, COP*
Health Care	*H*	*ABT, MRK, AMGN, BMY, ANTM, GILD*
Industrials	*I*	*UPS, HON, BA, CAT, RTX, MMM*
Technology	*TEC*	*AAPL, NVDA, INTC, ORCL, INTU, QCOM*
Telecommunications	*TEL*	*CMCSA, VZ, T, TMUS, LUMN, JNPR*
Utilities	*U*	*D, WM, EXC, SRE, XEL, RSG*

Table 1. Companies and sectors making up the database

Once the companies to work with have been selected, the database can be constructed.

1.9.2 DATA RETRIEVING USING BLOOMBERG

The source of the data used in the project is the *Bloomberg Terminal*, which allows creating very complete databases in Excel. Using this tool, almost any financial database can be created.

To train the algorithms, quarterly historical data will be needed for each of the selected companies.

The first question that arises is: Which time interval should be chosen? To avoid possible distortions in the models due to the 2008 financial crisis and the *Covid-19* crisis, data from

2010 to 2019 will be used. The database will then contain, for each of the 54 companies, data corresponding to 44 quarters.

Once the companies and the time intervals to work with are known, the data can be obtained from Bloomberg. The database includes:

- The date to which each row of data corresponds.
- The quarter to which each row of data corresponds.
- The sector to which the company belongs (included in each row of data).
- The share price for each quarter.
- All the data explained in Chapter 2 and included in Appendix A.

1.9.3 DATA LABELING

Classification algorithms, as they belong to the Supervised Learning category, always require labeled data. Obviously, the data provided by *Bloomberg* is not intended to be used to train classifiers; therefore, the class to which each observation belongs is not provided directly.

Excel is used to label each row of data. Since a company's share price is available for both quarter n and quarter $n+1$, the value of the increase in the share price can be easily calculated using the equation *E. 6*.

$$E.\ 6 \qquad \Delta Price_n = \frac{Price_{n+1} - Price_n}{Price_n}$$

This increment allows the sample in question to be labeled according to the following rule:

$$Label_n = \begin{cases} 1\ if\ \Delta Price_n > 0 \\ 0\ f\ \Delta Price_n \leq 0 \end{cases}$$

This way of manually classifying the data has a drawback: since the prices for quarter 45 are not available, the row of data corresponding to quarter 44 for each company is lost. Therefore, 54 samples in our database become unusable because they cannot be labeled. To avoid clutter, once the samples have been sorted, all rows 54 are removed.

As can be observed by looking at the labeling rule, a binary classification task is proposed, being the classes:

- Class '0': The share price will fall or remain stable over the next four months.
- Class '1': The share price will rise over the next four months.

1.9.4 PROCESSING OF THE COMPARABLE MULTIPLES

On the other hand, it should be recalled that the database includes some ratios that, in the financial world, are usually used by comparing them with those of other companies in the same sector (comparable companies). These are the so-called *Comparable Multiples* in the classification established in Figure 3.

In this sense, it may seem useful to have a value for these multiples that compares the multiple of the company with the sector average at each date. A new feature or variable '*multiple_industry$_n$*' is thus created containing the industry average in each quarter for each multiple. It should be noted that, for quarter n, the six companies of the same sector will have the same value for this new variable. Therefore, as is logical, this new feature cannot be used to train the model because it will not be linearly independent and would not be useful for learning anything.

Once the average of each sector has been calculated, the feature of interest '*multiple_DIF*' can be computed by means of the following equation:

$$multiple_n^{DIF} = \frac{multiple_n^{company} - multiple_n^{industry}}{multiple_n^{industry}}$$

This feature, together with the multiple itself, has the necessary characteristics to be included in the models.

1.9.5 DATABASE STRUCTURING. DATA SETS

Once all the necessary preliminary operations have been performed, the data can be formatted in its final version.

Before accumulating all the sectors in a single spreadsheet, it is important to know that, for future algorithm training, two different datasets will be needed [4]:

- The *training set*, which will serve to train the model and to learn the weights of the discrimination functions.
- The *test set*, which will be used, once the model has been trained, to test it with unseen data. It helps to understand its performance.

Generally, 70% to 80% of the data is allocated to the training set and the rest to the test set when the database is not particularly large. For larger data sets, it is considered enough to allocate approximately 5% of the data to the test set [12].

Sometimes, a third data set called the *validation set* can be used. This allows the best model to be selected, for example, when training the same algorithm, but with different inputs. However, when the data set is not that large, a common practice is to obtain the validation set from subsets of the own training set.

For the purposes of the project a ratio of 75:80 will be used for the training set and validation set respectively.

Basically, in Machine Learning, these sets are chosen randomly by setting the desired proportion. Though, for temporal data, doing so does not make sense, since it could be the case that a data sample at $t = d$ is used to validate the model generated by data samples at $t > d$. This situation would be something like using the future to predict the past, which is unreasonable.

In order to avoid making this mistake, the datasets will be generated in temporal order rather than randomly. This ensures that past data is used to predict the future.

A threshold value can then be set to allow the database to be split according to the desired ratio (75:25). In this case, 'December 31, 2016' could be used as that cut-off value, so that all samples between 2010 and 2016 fall into the training set, and those between 2017 and

2019 end up in the test set. In this way, rows 1 to 32 of each company will be assigned to the training set and rows 33 to 43 to the test set.

Once the sets have been created, it is important to make sure that they both contain a sufficient representation of each of the classes. In time series, for the reasons just mentioned, it is practically impossible to guarantee equal distribution in both sets. In these cases, it is sufficient to ensure that both classes are somewhat represented in each set (less than 25% cannot be accepted).

To see the proportion of each class, once the data sets are divided, the following code, which has been taken from the source code, is applied:

```
#  To execute this code it is necessary to have previously executed <import
   pandas as pd>.

train_p = (y_train.sum() / y_train.shape[0])*100
      print('The class 1 ratio in the training set is: ',train_p,'% of class 1')

test_p = (y_test.sum() / y_test.shape[0])*100
      print('The class 1 ratio in the test set is:',test_p, '% of class 1')
    -Output:
        The class 1 ratio in the training set is: 63.6574% of class 1
        The class 1 ratio in the test set is: 62.7946% of class 1
```

From the output of the above program, it is noticeable that the two datasets have a very similar proportion of classes. About 63% of the samples in each set belong to 'class 1', while the rest belong to 'class 0'. The fundamental reason why this happens lies in the fact that the *S&P500* has been characterized by large rises in recent years; therefore, it is logical to have more samples for which the share price will rise in the following quarter.

To make subsequent programming easier, the database is structured in such a way that the division into training and test set can be carried out directly. In a new spreadsheet, all rows to be included in the training set are collected first (one after the other), followed then by the test set rows.

The database will finally have the structure shown in Figure 10.

	Row	***Industry***	***Company***	***Time (Q)***	***Price***	***ΔPrice***	***Label***	***Features***	
TRAINING	*1*	*BM*	*C1*	*[1,32]*	P^1_{1-32}	ΔP^1_{1-32}	*0/1*	...	...
	.		*C2*	*[1,32]*	P^2_{1-32}	ΔP^2_{1-32}	*0/1*	...	...
	.		*C3*	*[1,32]*	P^3_{1-32}	ΔP^3_{1-32}	*0/1*	...	...
	.		*C4*	*[1,32]*	P^4_{1-32}	ΔP^4_{1-32}	*0/1*	...	...
	.		*C5*	*[1,32]*	P^5_{1-32}	ΔP^5_{1-32}	*0/1*	...	...
	.		*C6*	*[1,32]*	P^6_{1-32}	ΔP^6_{1-32}	*0/1*	...	...
	.	*CD*	*C1*	*[1,32]*	*P*	*ΔP*	*0/1*	...	...
	.		...	...	...	...	...	...	...
	.	*CS*	*C1*	...	*P*	*ΔP*	*0/1*	...	...
	. *1728*	...	...	...	...	...	...	...	...
TEST	*1729*	*BM*	...	*[33,43]*	*P*	*ΔP*	*0/1*	...	...
	.	*CD*	...	*[33,43]*	*P*	*ΔP*	*0/1*	...	...
	.	...	...	...	...	...	...	...	...
	2322	...	...	...	...	...	...	...	...

Figure 12. Database Final Structure

The database, structured in this way, contains 2,322 rows and 80 columns. Thus, 185,760 elementary data cells are available.

To work with the *pandas* library, it is important to have a database converted to *Comma Separated Values* format (*csv*). The database will therefore be named as '*_DATA_.csv*'.

1.10 DATA CLEANING AND PRE-PROCESSING

Starting from an already structured database, it is possible to start working with Python. First of all, the data file is imported into the *Jupiter Notebook* under the name *raw_df*. For this purpose, the following code is executed:

```
import pandas as pd
raw_df = pd.read_csv('/Users/javierreyzabal/Desktop/TFG/DATA/FINAL_DATA/__DATA__.
   csv', sep=';', decimal=',')
```

First of all, we proceed to remove all those columns and rows that, as previously indicated, would not be useful for the learning process of the algorithms. These are:

- The rows corresponding to each company's quarter number 44.
- The columns '*Date*' and '*Time*', which were used exclusively for structuring the data.

- The '*multiple_industry*$_n$' columns, created to compute '*multiple_DIF*' features.
- The column '*REVENUE_SEQUENTIAL_GROWTH*'. This column is deleted because it is inconsistent with the models that are intended to be created. The aim is to analyze the quarterly growth and the variable measures the growth year over year (*YoY*).

Another common practice in data cleaning is to put the column containing the data labels (T) in the last place, in order to facilitate the visualization of the data. Doing so will simplify the subsequent work.

1.10.1 DEALING WITH MISSING VALUES

Most algorithms, such as logistic regressions, only accept numerical values as input. For this reason, it is essential to make sure that there are no empty values or '*N/A*' values in the data frame (*df*). The following code returns the proportion of missing values present in the *df*:

```
raw_df2.sample()
missingvalues_count = raw_df2.isnull().sum()
missingvalues_count[:]
total_cells = np.product(raw_df2.shape)
total_missing = missingvalues_count.sum()
missing = (total_missing/total_cells)*100
print('The proportion of total missing values is:',missing, '% of the data')
    -Output:
        The proportion of total missing values is: 0.0 % of the data
```

As there are no missing values in the *df*, no further actions are required.

1.10.2 DEALING WITH NON-NUMERICAL VALUES

It has already been mentioned that many of the classifiers do not support nominal data. It is therefore necessary to code these features. In this case, there is only one non-numeric variable, the one related to the sector ('*INDUSTRY*').

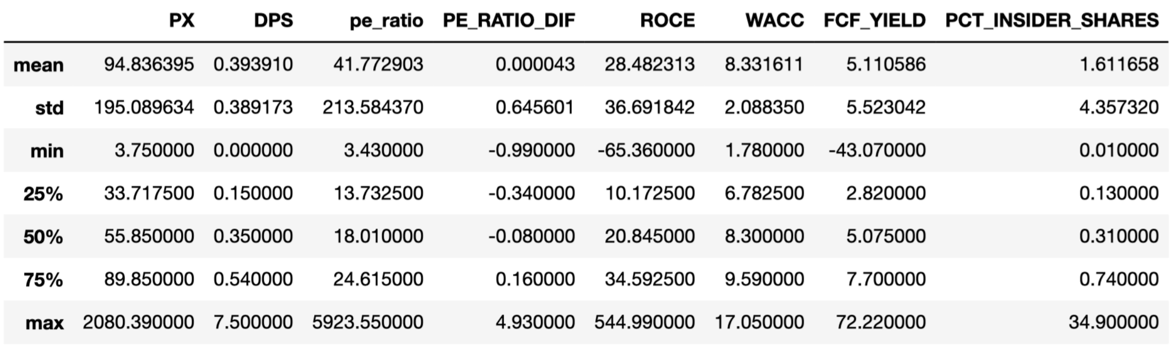
The *One Hot Encoding* technique will be applied to process this data. This technique codes *n* classes by creating *n* different binary classification columns (one for each class) and eliminating the first of them (to avoid multicollinearity). With this method, no information is lost [12].

Thus, if there are nine different industries, eight new columns will be created to represent the last eight sectors. A 0 in all the eight new columns would then indicate that the row belongs to the first sector; a 1 in the first column would mean that it belongs to the second sector, and so on. Once the variable 'INDUSTRY' is coded, the number of columns in the *df* will increase to 73.

1.11 EXPLORATORY DATA ANALYSIS (EDA) AND OUTLIERS

EDA is considered a crucial step in *ML*. It "the stage where we actually start to understand the message contained in the data".

In a first approach to *EDA*, it is always advisable to perform an analysis of descriptive statistics. As we have many features, we decide to analyze only some of the most representative variables. The results are shown in the following table.



	PX	DPS	pe_ratio	PE_RATIO_DIF	ROCE	WACC	FCF_YIELD	PCT_INSIDER_SHARES
mean	94.836395	0.393910	41.772903	0.000043	28.482313	8.331611	5.110586	1.611658
std	195.089634	0.389173	213.584370	0.645601	36.691842	2.088350	5.523042	4.357320
min	3.750000	0.000000	3.430000	-0.990000	-65.360000	1.780000	-43.070000	0.010000
25%	33.717500	0.150000	13.732500	-0.340000	10.172500	6.782500	2.820000	0.130000
50%	55.850000	0.350000	18.010000	-0.080000	20.845000	8.300000	5.075000	0.310000
75%	89.850000	0.540000	24.615000	0.160000	34.592500	9.590000	7.700000	0.740000
max	2080.390000	7.500000	5923.550000	4.930000	544.990000	17.050000	72.220000	34.900000

Table 2. Descriptive Statistics

For the '*PX*' variable, which represents the price of the stock, it is logical to expect outliers, since only a few companies trade above $250. This fact is shown in the following box-and-whisker plot.

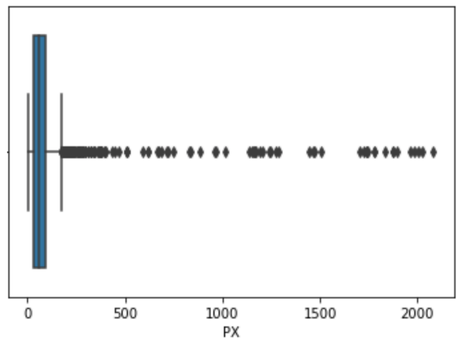


Figure 13. 'PX' Box-and-Whiskers Plot

In the diagram above, every mark beyond the right whisker is a company that, in terms of share price, can be considered as an outlier. This is not only true for the '*PX*' variable, but, in general, for the other variables as well.

The same applies to the variables '*DPS*' (dividends per share) and '*ROCE*' as shown in the following figure.

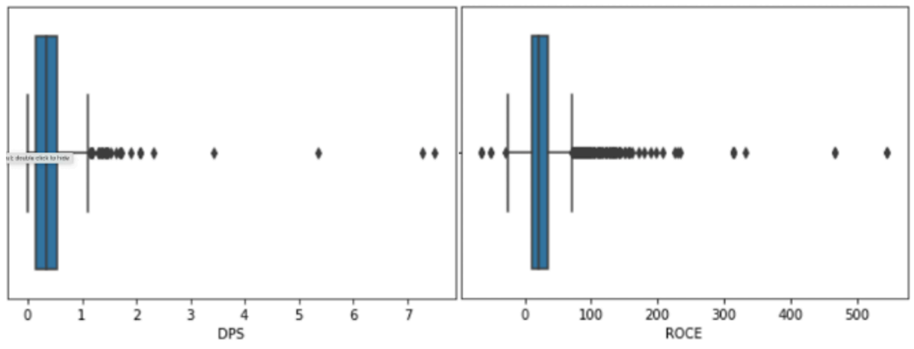


Figure 14. 'DPS' and 'ROCE' Box-and-Whiskers Plot

For comparable multiples, the number of outliers in them skyrockets. However, the '*multiple_DIF*' variables created earlier help to give the data a shape more similar to a standard normal distribution, also reducing the number of outliers.

The following figure shows what has just been said, exemplified for *P/E* variables.

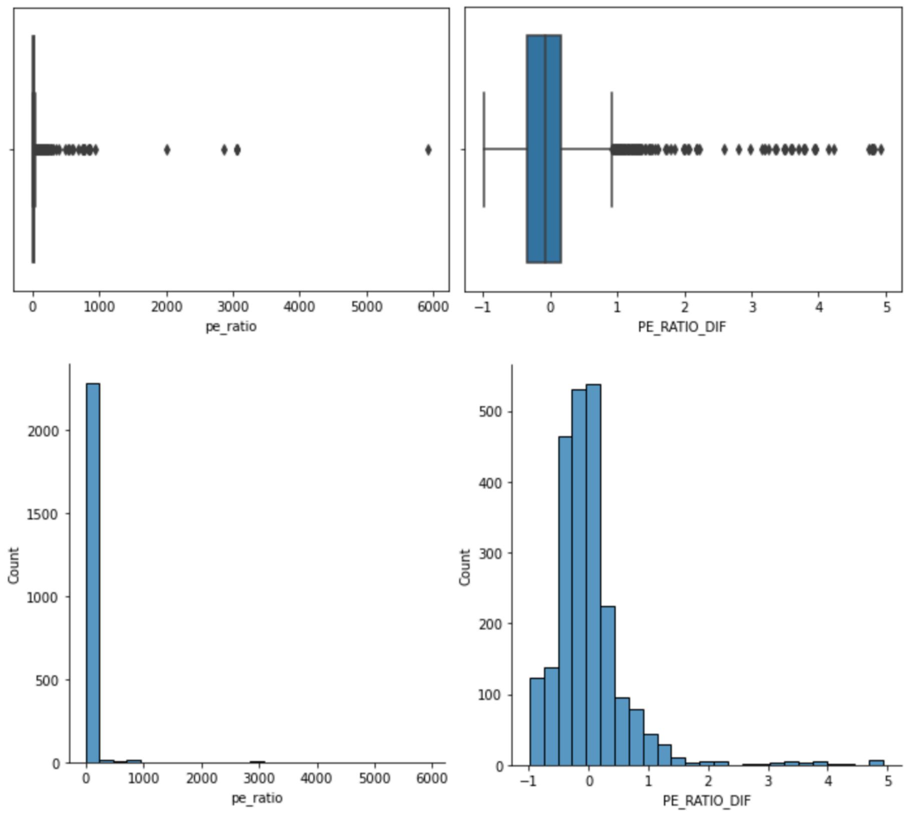


Figure 15. Analysis of the Outliers of Comparable Multiples

It has been demonstrated that there are numerous outliers in the data sample. Generally, these outliers (when few in number), can be treated, eliminated or replaced. However, as in this case there are so many, applying these techniques could lead to the creation of an erroneous model that may not represent the reality.

It can be said then that, as there are so many extreme values, they can almost be considered as normal values for the task proposed in this project. For this reason, it has been decided to keep at least most of them in the *df*, since they are an important part of the data.

The models to be used are not very robust to outliers, especially the logistic regression model. Therefore, this decision to keep all the outliers could lead to models that suffer from overfitting, i.e., models that, because of the high variance present in the training data, do not generalize well with the unseen data. These overfitted models are generally very complex

[12]. To fight against complexity and avoid overfitting, some statistical techniques can be used, such as regularization. These techniques can help to generate good models using data containing outliers.

In addition, the existing literature supports the decision of maintaining at least part of the outliers in the *df*. For example, a study carried out by professors at the University of Seville concluded that the total elimination of outliers does not achieve significant improvements with any classifier. Nevertheless, this study also concludes that the partial elimination of outliers does significantly improve the results obtained with the original data sets [22].

1.11.1 OUTLIERS MANAGEMENT

Let's see what happens if part of the outliers are discarded. By executing the following code:

```
from scipy import stats
z = np.abs(stats.zscore(df))
df_try_z2 = df[(z<=2).all(axis=1)]
print('By eliminating the values with a z-score greater than 2, we are left with
   ',df_try_z2.shape[0], 'rows')
```

```
    -Output:
        By eliminating the values with a z-score greater than 2, we are left with
        121 rows
```

This program removes from the data frame the rows of all those samples that have a *Z-Score* greater than 2, i.e., that are more than 2 standard deviations away from the population mean. This would leave only 121 rows of data, which is not enough. If the same code is ran, but for higher *Z-Score* values, the results shown on the table below are obtained.

Elimination condition	Remaining Rows	Remaining Rows in the df (%)
$Z - Score > 1$	10	0.43 %
$Z - Score > 2$	121	5,21 %
$Z - Score > 3$	1,483	63,87 %
$Z - Score > 4$	1,775	76,44 %
$Z - Score > 5$	1,977	85,14%

Table 3. Possible Scenarios for the Outlier Elimination

When selecting which filter to apply, it should be kept in mind that, by eliminating a large number of outliers, the model becomes less realistic but theoretically more effective. Therefore, taking this into account, it seems logical to discard only the most extreme values (those that may be caused by errors).

According to what has been said, the criterion of eliminating those values with a *Z-Score* greater than 5 seems to be appropriate. In this way only 14.8% of the data would be lost.

Hence, outliers that are more than 5 standard deviations away from the population mean will be eliminated, thus creating a new data frame called *df_out*. However, the original data set*df* is preserved to check later whether this correction of the data improves the performance of the models.

It is important to check that, after this reduction, the proportion of data in the training set and in the test set for this new *df_out* is maintained, as well as the proportion of classes in each. The following code is therefore executed:

```
prop_out = (df_training_out.shape[0] / (df_training_out.shape[0] +
   df_test_out.shape[0]))*100
print(prop_out, '% of data rows are in the training set')
training_out_prop = (df_training_out.iloc[:,72].sum() /
   df_training_out.shape[0])*100
print('The class 1 ratio in this new training set "df_out"is:',training_out_prop,
   '% of class 1')
test_out_prop = (df_test_out.iloc[:,72].sum() / df_test_out.shape[0])*100
print('The class 1 ratio in this new test set "df_out" is:',test_out_prop, '% of
   class 1')
```

```
    -Output:
        75.61962569549823 % of data rows are in the training set
        The class 1 ratio in this new training set "df_out" is: 63.81% of class 1
        The class 1 ratio in this new test set "df_out" is: 62.44% of class 1
```

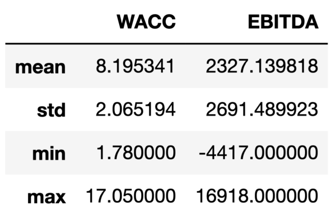
It is worth mentioning that, for the above code, *df_out* had already been split, forming the training set '*df_training_out*'. The test set, however, should not be modified. The usual one will then be used to check the true efficiency of the models.

The output of the previous code shows that the data structure and the representation of the classes are practically unchanged.

1.12 FEATURE SCALING

Feature scaling is another fundamental step to take before starting to train models.

Most algorithms (especially linear classifiers) are very susceptible to different data scales. The variables handled in this project have very different values. The maximum and minimum values vary significantly from one characteristic to another. The following table shows how that difference can be enormous.



	WACC	EBITDA
mean	8.195341	2327.139818
std	2.065194	2691.489923
min	1.780000	-4417.000000
max	17.050000	16918.000000

Table 4. 'WACC' and 'EBITDA' Value Ranges

It can be seen that the scale of one variable is a hundred times larger than the other. If these values are used to train a linear classifier, the optimization algorithm will probably give more importance to the '*EBITDA*' variable, since it will optimize the weights of the discrimination function by prioritizing the largest errors [12].

For this reason, it is convenient to transform the variables so that they all have ranges of values on the same scale. To do this, there are two different approaches: normalization and standardization. The first one generally rescales the features into a range of values from 0 to 1. This type of scaling causes much of the useful information about the outliers to be lost.

Alternatively, by standardization, the data take the form of a standard normal distribution. Furthermore, this type of scaling retains a bit more the information about atypical data and makes the algorithm less sensitive to them [12].

According to this, standardization should be better for the objectives of this project, since it maintains the information of the outliers. To standardize a data sample, the equation E.7 must be applied.

$$E.\ 7 \qquad x_i^{std} = \frac{x_i - \mu_x}{\sigma_x}$$

However, it can be noticed how, in practice for our database, both methods produce almost identical results. The figure below exposes this.

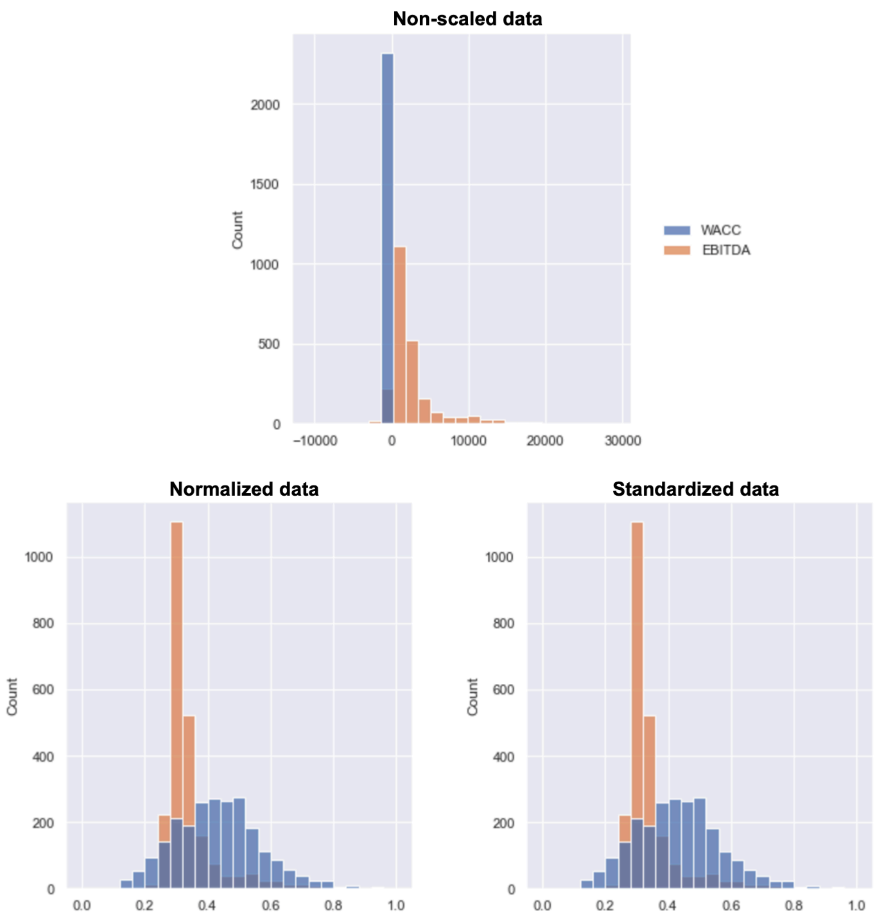


Figure 16. Normalization and Standardization Comparison

For these reasons, it was decided to apply standardization to all the data in our df. To do so, the following code is used:

```
from sklearn.preprocessing import StandardScaler
stand = StandardScaler()
X_train_stand = stand.fit_transform(X_train)
X_test_stand = stand.fit_transform(X_test)
X_train_out_stand = stand.fit_transform(X_train_out)
X_test_out_stand = stand.fit_transform(X_test_out)
```

To sum up what has been presented in this chapter, it can be concluded that four types of data sets have been obtained, all of them ready to be used with the algorithms:

- df data.
- df_out data, consisting of the *df* data without the most extreme outliers (those with a *Z-Score* greater than 5).
- df_stand data, composed of the standardized *df* data.
- df_out_stand data, composed of the standardized *df* data without the most extreme outliers.

CHAPTER 6. MODEL GENERATION WITH SUPERVISED LEARNING CLASSIFIERS

Once the data has been cleaned and preprocessed, the generation of the models can begin. In this chapter two of the most widely used linear classifiers will be implemented: *Logistic regressions* (*LR*) and *Supported Vector Machines* (SVM).

1.13 LOGISTIC REGRESSIONS

1.13.1 DESCRIPTION OF THE TECHNOLOGIES

Based on what was explained in Chapter 2 about linear classifiers such as *Adaline*, it is easy to describe how logistic regressions work.

To arrive at the activation function used for *LR*, which will be the sigmoid function, it is necessary to understand what the logit function is:

$$logit\ (p) = \ln\left(\frac{p}{1-p}\right), \forall\ p \in [0,1]$$

In this function, p indicates the probability of occurrence of a particular event (e.g., that a sample belongs to a certain class). The logit function takes then as input a value between 0 and 1 and transforms it into a number belonging to the whole real scale [12]. This property can be used to relate the linear discrimination function *z* with the probability that a sample belongs to the positive class (class '1):

$$E.\ 8 \qquad logit\big(p(y=1|\bar{x})\big) = ln\left(\frac{p}{1-p}\right) = \sum_{j=0}^{l} w_j\, x_j = z$$

The above equation takes as input the probability $p(y=1|\bar{x})$. However, for classification, what matters is to have a function that calculates this value, given $z(w,x)$.

Therefore, inverting the logit function in equation E. 8 may help. By doing so, we obtain the following expression [12]:

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

This function, called *sigmoid function*, is useful for the purpose of a classifier. Once z is trained, it returns the probability that a sample of $\bar{x}$ features belongs to 'class 1'.

Consequently, the logistic regression will have a sigmoid function as activation function $\phi(z)$, which will have the following shape:

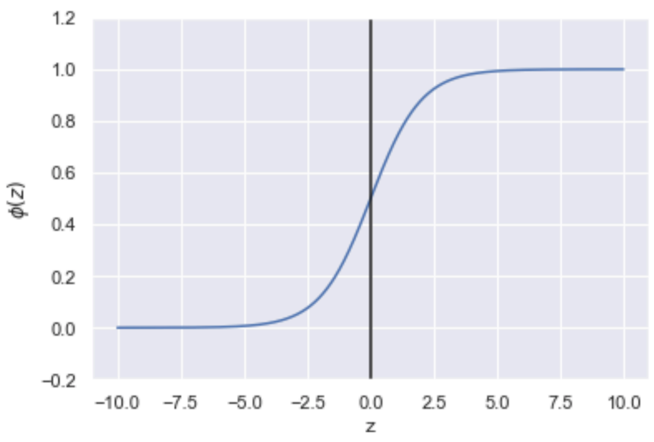


Figure 17. The Sigmoid Function

The threshold (or decision) function will again be a step function depending on the probability predicted by ϕ. It will have the following form [12]:

$$\hat{y} = \begin{cases} 1 \; if \; \phi(z) > 0.5 \\ 0 \; otherwise \end{cases}$$

1.13.2 MODEL GENERATION

In this section different models will be trained. The main objective will be to improve the basic model using techniques such as regularization or feature selection.

1.13.2.1 Basic Models

To begin with, a first model is generated. This model uses all the characteristics of the *df*. Obviously, many of the variables will be of no importance in predicting whether a stock will rise. In fact, many variables will only add noise to the model. The code used is as follows:

```
from sklearn.linear_model import LogisticRegression
model1 = LogisticRegression(C=10, max_iter=100000)
model1.fit(X_train, y_train)
pred1 = model1.predict(X_test)
print(classification_report(y_test,pred1))
```

This first model is obviously a poor model. The confusion matrix, which represents the different types of prediction errors and hits, is as follows:

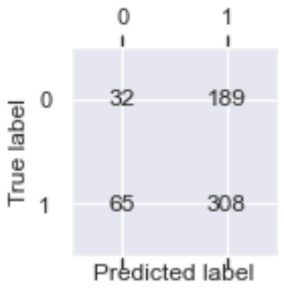


Figure 18. Confusion Matrix of LR Model 1

The evaluation metrics for *Model 1* are shown in the following table:

Logistic Regression *Model 1*	
Accuracy (***ACC***)	0.61
Precision (**PRE**)	0.62
Recall (**REC**)	0.83
$\boldsymbol{F_1}$	0.71
Negative Recall (**N-REC**)	0.14
Negative Precision (**N-PRE**)	0.20

Table 5. Evaluation Metrics of LR Model 1

As can be seen, this first model classifies 61% of the samples correctly. This is obviously not optimal.

A theoretical objective that an acceptable model should achieve is to have accuracy and precision metrics greater than 63%. This is because, knowing that 63% of our observations are '*class 1*' data, i.e., stocks that go up in the stock market the following quarter, we could establish a second model (*Model 0*) that classifies all stocks in category 1. Thus, this new model would ensure an accuracy of 63%.

Model 0 would have the following evaluation metrics:

Model 0	
Accuracy (***ACC***)	0.63
Precision (**PRE**)	0.63
Recall (**REC**)	1
$\boldsymbol{F_1}$	0.77
Negative Recall (**N-REC**)	0
Negative Precision (**N-PRE**)	-

Table 6. Evaluation Metrics of Model 0

Another negative aspect of *Model 1* is that it has barely learned how to predict which stocks will fall in the future. Taking a look at the negative recall (*N-REC*), we can see that the model is only able to correctly predict 14% of samples belonging to the '*0 class*'. Moreover, every time it predicts a '*class 0*', it is only correct in 20% of the cases.

On the other hand, as a positive aspect, it should be emphasized that the Recall (*REC*) of the model is quite good. Specifically, the model correctly predicts 83% of the samples belonging to 'class 1'. However, this is useless considering that its precision (*PRE*) is 62%, i.e., every time it predicts that a stock is going up, it is only correct in 62% of the cases.

It can be said then that *Model 0*, having higher accuracy and precision (both around 63%), is in the end, better than *Model 1*.

If the aim is to predict which stocks will fall the following quarter, a third simple model named *Model 0-bis* would still be more appropriate. This model would predict that all stocks would go down, having a negative accuracy and precision close to 37% (higher than those of Model 1).

Models of this type have also been trained for the other *df* (those that contain fewer outliers and those that are standardized). The results were worse in all aspects compared to those obtained for *Model 1*. For this reason, they are not shown, as they would not provide any improvement.

1.13.2.2 Basic Models using Regularization

The objective now is to improve the metrics obtained with the previous models.

Knowing that *Model 1* is very complex (because it takes into account all the characteristics), it is probably overfitted. Let us remember that by means of the *L2 regularization*, it is possible to restrict the model by making it los overfitted [23].

The following figure represents the *validation curves* of Model 1.

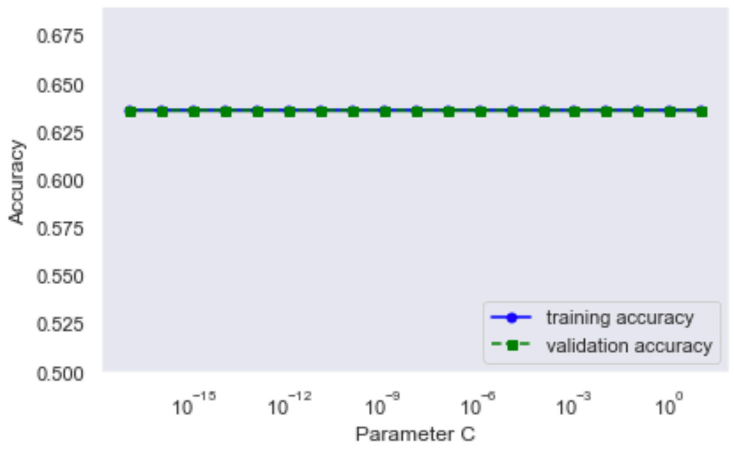


Figure 19. Model 1 Validation Curves

These curves are used to optimize the value of the hyperparameters (in this case, the value of *c*). They are constructed by training the model for different values of *c*, using cross-validation in each training, and subsequently calculating the average accuracy of the trained models for each value of *c*. It should be remembered that cross-validation consists of dividing the training set into *k* different sets, using *k-1* of them to train the model and the rest to validate it; this process is repeated *k* times so that all the sub-sets are used at some point as validation-sets, thus generating k models [12].

It should be noted that cross-validation techniques are not usually used with time series, since models may be validated with older data than that used to train them. However, it can be a good way to intuit the value of *c* that optimizes the model. From now on, the validation curves will only be used for that, to intuit the optimal hyperparameters.

From Figure 19 it can be concluded that even by increasing the regularization to extremely low values of *c*, the efficiency of the model does not increase. However, it does become simpler. In fact, it can be verified that for a value of $c = 10^{-10}$, the trained model would match *Model 0* (it would classify everything as '*class 1*'). The following code proves this:

```
model1b = LogisticRegression(C=0.00000000001, max_iter=1000000)
model1b.fit(X_train, y_train)
pred1b = model1b.predict(X_test)

from sklearn.metrics import confusion_matrix
confmat1b = confusion_matrix(y_true=y_test, y_pred=pred1b)
```

This confusion matrix (*confmat1b*), well presented, would look like this:

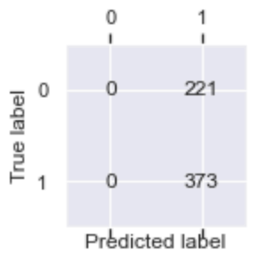


Figure 20. Confusion Matrix of Model 0 and Model 1B

1.13.2.3 Models with Manual Feature Selection According to their Correlation with the Class

Now, to improve the model, the variables to be included in will be manually selected. In this case, *Pearson's correlation coefficient* (ρ) will be used as the decision variable.

This coefficient takes a value between -1 and 1 and indicates whether there is a direct (when close to 1) or inverse (when close to -1) linear dependence between two variables [24].

Applying this to the '*TARGET*' variable with respect to the rest of the other features, it will be possible to select those variables on which the class is more dependent. The following table shows the characteristics that are most linearly dependent on the class label.

Positive Correlation		**Negative Correlation**	
Variable	**ρ**	**Variable**	**ρ**
Quarter	0.079	*Capex*	-0.074
Sales_To_Assets	0.070	*Industry_E*	-0.073
Industry_CD	0.055	*Total_Equity*	-0.071
Fcf_Yield	0.053	*Assets*	-0.057
Cost_Equity_WACC	0.048	*Cost_Debt_WACC*	-0.049
Cash_From_Inv_Act	0.046	*Book_Val_Per_Sh*	-0.046
Industry_I	0.038	*Cur_mkt_cap*	-0.046
Sales_To_Assets_DIF	0.037	*Ev_To_Sales*	-0.044
ROIC	0.034	*Net_Debt_To_EBITDA*	-0.043
WACC	0.033	*Liabilities*	-0.040

Table 7. Features-Labels Correlation

We test with the four different *df*, although, a priori, the results of the models trained with standardized data should be better.

A model is then trained using the above variables (all 20). This model, once *c* is adjusted, does not improve the metrics of the first one. Therefore, several models will be generated by combining the variables shown in Table 7.

After different attempts and combinations of features, it has been observed that the best one is a model that includes the 5 most correlated features (with the class variable), i.e., a model that includes variables such as the quarter to which the sample belongs, the sales to assets ratio or the Capex. It is worth mentioning that the metrics of the already optimized model improve when the standardized dataset is used. The regularization of the model has been optimized using the following validation curves:

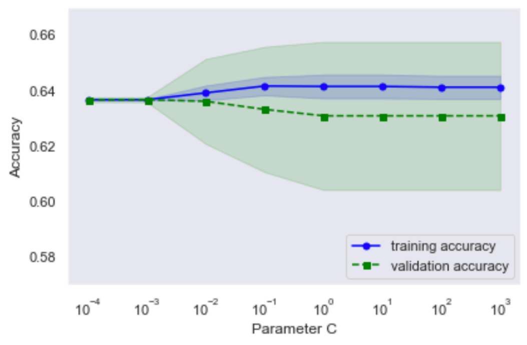


Figure 21. Model 2 Validation Curves

From the above image, it can be seen that the optimal value of the hyperparameter *c* is likely to be between 0.01 and 1. By training the model in a loop for each of these values of *c*, it is concluded that the optimal value of this hyperparameter is 0.11. This model is programmed as follows:

```
Model2 = LogisticRegression(C=0.11)
Model2.fit(X_train_stand.iloc[:,[2,8,32,58,60]], y_train_stand)
```

The variables used (2, 8, 32, 58 and 60) are listed in AppendixA.

The confusion matrix of this new *Model 2* is shown below.

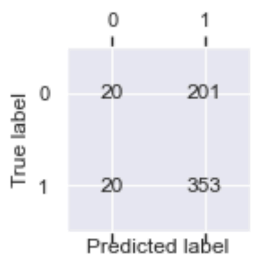


Figure 22. Confusion Matrix of Model 2

By testing this model with the test set, the following evaluation metrics are obtained:

Logistic Regression ***Model 2***	
Accuracy (***ACC***)	0.63
Precision (**PRE**)	0.64
Recall (**REC**)	0.95
$\boldsymbol{F_1}$	0.76
Negative Recall (**N-REC**)	0.10
Negative Precision (**N-PRE**)	0.51

Table 8. Evaluation Metrics of Model 2

Analyzing the table above, it can be concluded that the metrics obtained for *Model 1* are improved. In particular, it matches the accuracy of *Model 0* and exceeds its precision. However, it is still not optimal.

As a positive point, we can highlight that this model is already better than Model 0. It manages to classify 63% of the samples well and does so with an accuracy of 64%. This means that, when the model says that a stock will go up, in 64% of the cases it will do so.

Nonetheless, this improvement is due to the increase in recall (95%). Paying attention to this value, it can be concluded that the model predicts most of the time that the stock will go up. In fact, it predicts only 20 dips compared to 554 rises predicted. As for the negative accuracy, it has increased to 51%. This means that negative predictions have a reliability of 51%.

In conclusion, it can be said that this second model improves the predictive capacity of the base model. Specifically, it improves both the positive and negative precision, i.e., the reliability that can be given to a classification.

It can also be concluded after this that, indeed, logistic regressions work better with standardized data. If we train *Model 2* with unstandardized data and optimize its hyperparameters, we obtain the following metrics:

Logistic Regression *Model 2* Without Standardized Data	
Accuracy (***ACC***)	0.61
Precision (**PRE**)	0.63
Recall (**REC**)	0.95
$\boldsymbol{F_1}$	0.76
Negative Recall (**N-REC**)	0.10
Negative Precision (**N-PRE**)	0.51

Table 9. Evaluation Metrics of Model 2 Without Standardized Data

Indeed, data standardization helps to improve the accuracy and precision of the model. This type of data will then be the only one used from now on.

Model 2 is not very regularized. In fact, as already mentioned, the optimal regularization parameter is $c = 0.11$. To check that there is indeed not much overfitting, we can use this model to predict the training set and compare its metrics with those already obtained. If they are similar, it means that the model, even if it is not extremely accurate, is correctly trained. The metrics obtained with the training set are as follows:

Logistic Regression *Model 2* Using the Training Set	
Accuracy (***ACC***)	0.64
Precision (**PRE**)	0.64
Recall (**REC**)	0.98
$\boldsymbol{F_1}$	0.77
Negative Recall (**N-REC**)	0.04
Negative Precision (**N-PRE**)	0.51

Table 10. Evaluation Metrics of Model 2 Using the Training Set

Indeed, it can be observed that the metrics are very similar to those obtained by testing the model with the test set, therefore, we can say that there is apparently no correlation.

1.13.2.4 Modeling with Data Dimensionality Reduction

In an attempt to further improve the model, we will try to reduce the data set dimension by discarding the least useful information.

There are techniques that make it possible to reduce the dimension of the data. One of them is called *Principal Component Analysis* (*PCA*). The study of this technique is not the subject of this paper, but it is worth mentioning that it is based on generating new columns by preserving the variance of the data [25].

In this way, through a simple Scikit-Learn command, the dimension of a dataset can be reduced from the initial number of figures to the desired number of columns.

Different models will be trained, varying the degree of dimension reduction. For each of them, the hyperparameters will be adjusted.

After training, it was observed that the best model, in this case, not even matched the performance of *Model 2*. This new *Model 3* consists of applying a *PCA* that reduces the dimension to 5 new columns. It is created by applying the following code:

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 5)

model = LogisticRegression(C=0.1, max_iter=1000000)

X_train_pca = pca.fit_transform(X_train_stand)
X_test_pca = pca.transform(X_test_stand)

model.fit(X_train_pca, y_train_stand)

pred = model.predict(X_test_pca)
```

The above code generates and tests the new *Model 3*, already with the optimized regularization. The confusion matrix generated by this model is as follows:

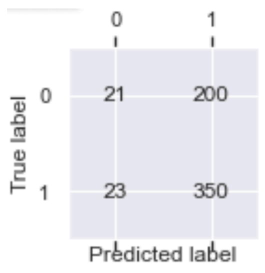


Figure 23. Confusion Matrix of Model 3

Again, the results are shown in the form of evaluation metrics for the model:

Logistic Regression *Model 3*	
Accuracy (***ACC***)	0.62
Precision (**PRE**)	0.64
Recall (**REC**)	0.94
$\boldsymbol{F_1}$	0.76
Negative Recall (**N-REC**)	0.10
Negative Precision (**N-PRE**)	0.48

Table 11. Evaluation Metrics of Model 3

It can be seen how, if compared to the best model so far (the second one), this new model does not improve it. The accuracy drops one point, and the precision remains at 94%. The difference is that the new model detects fewer positive cases (94%, 1% less). Therefore, with this model, a step backwards has been taken.

It should be noted that, as with Model 2, it has been verified that there is no obvious overfitting. By testing it with the training set, very similar figures are obtained.

1.13.2.5 Modeling with Less Data

At this point, it has been observed that, with the data available for this problem and through logistic regressions, it is difficult to obtain good results.

One of the reasons may be that, when working with the stock market, there are patterns that change from year to year. For example, it is well known that, during the 2008 crisis, most of the stocks strongly fell. Therefore, it has been decided to reduce the period of the training data, since, based on the above, if there are recognizable patterns, the model could improve.

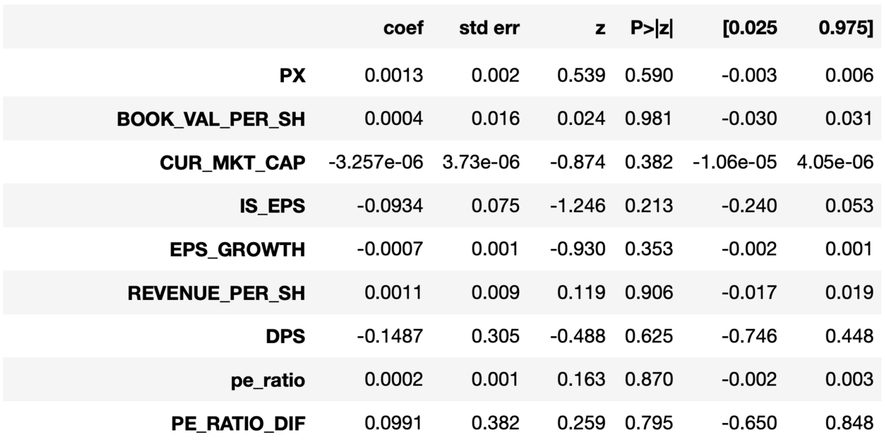
Data from 2017 and 2018 will be used to train the model. The test set, however, will be kept intact (since, if we do not keep them, the weight of each class in the training set could change). The input data for this new model are shown in the following table:

Training Set		**Test Set**	
Variables	All	**Variables**	All
Rows of data	431	**Rows of data**	594
'Class 1' Proportion	65%	**'Class 1' Proportion**	63%

Table 12. Characteristics of the Data Used to Train a Model with a Shorter Time Frame

It must be said that this new model did not obtain the expected results. All the previous techniques were used, starting with the manual selection of features up to the application of *Principal Component Analysis*. However, the best model obtained was identical to *Model 0*, with a disproportionately low *c* value.

It can be therefore concluded that the reduction of the time period used for the model is not a good technique. Surely, 431 rows of data are not enough to train it correctly. In fact, if one looks at the *p-value* obtained from the training of this *Model 4* for each feature, it is generally very high. This is shown in the following figure, where the data of the trained feature are shown:



	coef	std err	z	P>|z|	[0.025	0.975]
PX	0.0013	0.002	0.539	0.590	-0.003	0.006
BOOK_VAL_PER_SH	0.0004	0.016	0.024	0.981	-0.030	0.031
CUR_MKT_CAP	-3.257e-06	3.73e-06	-0.874	0.382	-1.06e-05	4.05e-06
IS_EPS	-0.0934	0.075	-1.246	0.213	-0.240	0.053
EPS_GROWTH	-0.0007	0.001	-0.930	0.353	-0.002	0.001
REVENUE_PER_SH	0.0011	0.009	0.119	0.906	-0.017	0.019
DPS	-0.1487	0.305	-0.488	0.625	-0.746	0.448
pe_ratio	0.0002	0.001	0.163	0.870	-0.002	0.003
PE_RATIO_DIF	0.0991	0.382	0.259	0.795	-0.650	0.848

Figure 24. Model 4 Training Information

1.13.2.6 Modeling by Training Set Tuning

So far, only one model has been obtained that outperforms *Model 0*, albeit narrowly and only in terms of accuracy.

From the above, it can be seen that the models usually fail to predict the *'0 classes'*. Therefore, in this section we will try to apply changes to the training set, so that the algorithm can learn to classify them better.

Elimination of "Gray Classes"

In a further attempt to improve the model, it was decided that it might be desirable to eliminate the *gray classes* from the training set. By *grey classes* we mean all those data rows close to the boundary between classification as '*class 1*' or '*class 0*'. These rows of data could confuse the algorithm.

It is therefore decided to filter the training set data and eliminate those samples that have a price change during the next four-month period that meets the following condition:

$$|PX\_INCREMENT\_1Q| < \mathrm{T}\ \%$$

Being T the threshold value that is set to apply the filter to the database. The following table shows the number of gray data in the training set for different values of T:

Value of **T**	Number of **grey rows**	% of grey rows
0.5%	65	3.8%
1%	123	7.1%
1.5%	194	11.2%
2%	254	14.7%
2.5%	317	18.9%

Table 13. Analysis of the Amount of Gray Data in the Training Set

A $T = 0.5\%$ filter is first applied. By training and optimizing the model, no significant progress was achieved. Seeing that perhaps the filter was too small, an attempt was made to train the model for a $T = 2.5\%$ filter.

For these new data, all the techniques explained above were applied. The variables were again selected based on their correlation with the class, *PCA* techniques were applied, etc. After testing different configurations of the model, we arrived at *Model 5* that, this time, improved the existing ones. Its confusion matrix is as follows:

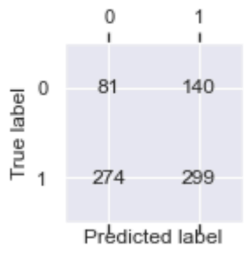


Figure 25. Confusion Matrix of Model 5

This fifth model already provides higher performance than the base model, with slightly more than a 64% accuracy rate.

Following this same strategy and testing different filters, we found that the best model of this type is obtained by applying a $T = 2\%$ filter, i.e., eliminating 14.7% of the training data. This particular model was obtained by applying a three-column *PCA* to the features that maximized the performance of *Model 2*, which, again, proved to be the most correlated ones with the class. The classifying model, after being optimized in terms of hyperparameters ($c = 10$), looked like this:

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 3)

model = LogisticRegression(C=10)

X_train_pca = pca.fit_transform(X_train_stand_T2.iloc[:,[2,8,32,58,60]])
X_test_pca = pca.transform(X_test_stand_T2)

model.fit(X_train_pca, y_train_stand)
pred = model.predict(X_test_pca)
```

Its confusion matrix is shown below.

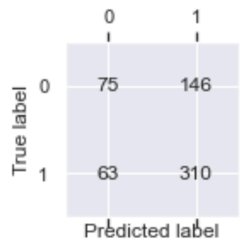


Figure 26. Confusion Matrix of Model 6

As can be appreciated, the results obtained are better than those of any other. Specifically, this *Model 6* improves in terms of the classification of the negative cases, although it gets a little worse in the prediction of positive classes. Its metrics are shown in the following table:

Logistic Regression *Model 6*	
Accuracy (***ACC***)	0.65
Precision (**PRE**)	0.68
Recall (**REC**)	0.83
$\boldsymbol{F_1}$	0.75
Negative Recall (**N-REC**)	0.34
Negative Precision (**N-PRE**)	0.54

Table 14. Evaluation Metrics of Model 6

Indeed, the model certainly improves. The accuracy is now 65%, two points above the minimum required. In addition, the precision of positive cases increases to 68%, which is very useful, knowing that the least desirable error is to make a mistake when buying a stock.

The recall of positive cases decreases a little, but this is what allows the model to improve, as it now classifies considerably more results as '*class 0*'. In terms of its ability to predict this class, the negative Recall also increases, as does its precision.

As a conclusion it can be said that *Model 6* is not a bad model. Of each sample that it classifies as '*class 1*' or '*class 0*', it is correct in 68% and 53% of the cases respectively. This results in an accuracy of 65%.

Balancing the Weight of the Classes on the Training Set

However, there is still room for further improvement in the results. As shown in Figure 26, the main problem with the algorithm is that it does not learn well how to classify negative labels. In fact, when entering a random sample into the classifier, it will classify it as '*class 0*' in only 23% of the cases.

This may be due to class imbalance in the data. As only 37% of the training data belongs to 'class 0' the algorithm surely learns more from rising stocks than from falling stocks. This may be the reason for the low negative recalls obtained until now.

To try to change this, it has been decided to balance the classes in the training set. More gray samples from '*class 1*' (those with revaluations closer to 0) will then be removed from the training set until the weight of the two classes is balanced. This operation is schematized in the last part of the following figure.

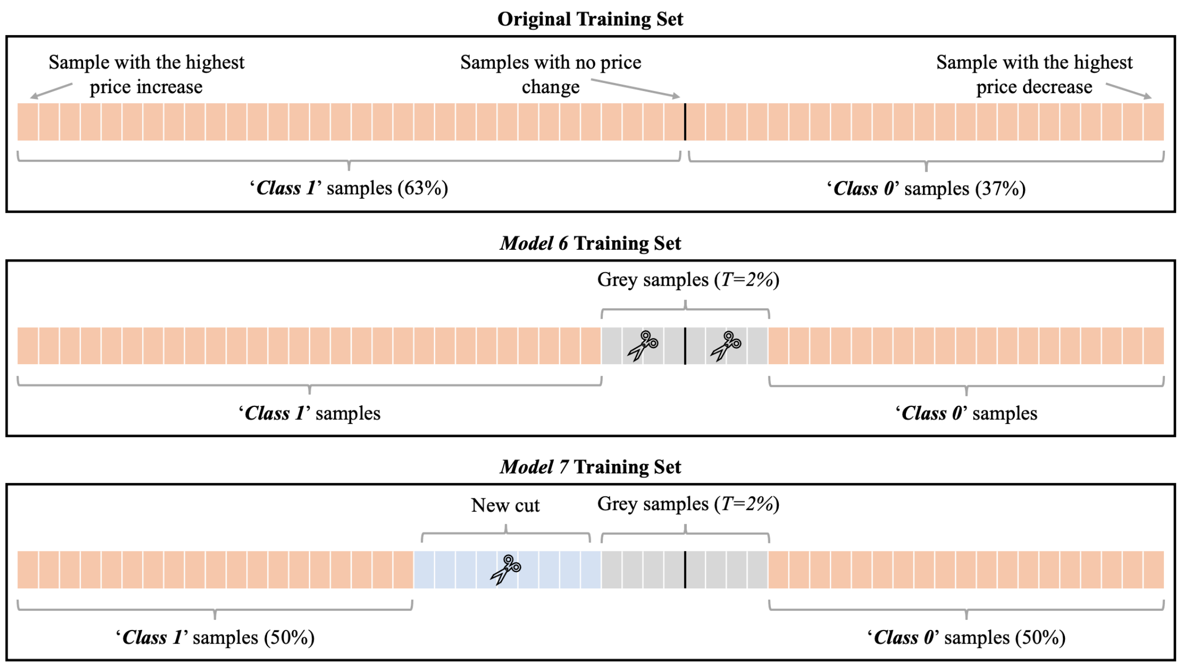


Figure 27. Schematic Diagram of Training Set Modifications

This results in a new training set with slightly more than 1,000 samples, but completely balanced.

Based on the new data set, a new *Model 7* is trained. It will have the same structure as *Model 6* (the best one so far), i.e., the same variables will be used. Once the hyperparameter *c* is optimized with the validation curves, the resulting code to train the new classifier is as follows:

```
from sklearn.linear_model import LogisticRegression
from sklearn.decomposition import PCA

pca = PCA(n_components = 3)
model = LogisticRegression(C=10)

X_train_pca = pca.fit_transform(X_train_stand_T2_bal.iloc[:,[2,8,32,58,60]])
X_test_pca = pca.transform(X_test_stand_T2_bal)

model.fit(X_train_pca, y_train_stand)
```

Its confusion matrix is as follows:

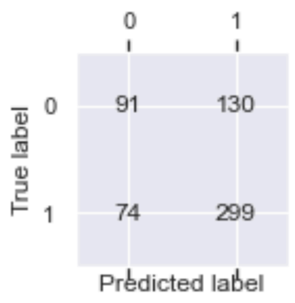


Figure 28. Confusion Matrix of Model 7

In addition, its evaluation metrics are presented in the table below.

Logistic Regression *Model 7*	
Accuracy (***ACC***)	0.66
Precision (**PRE**)	0.70
Recall (**REC**)	0.80
$\boldsymbol{F_1}$	0.74
Negative Recall (**N-REC**)	0.41
Negative Precision (**N-PRE**)	0.55

Table 15. Evaluation Metrics of Model 7

Indeed, with this new model, the results have been further improved. The figures obtained are already considerably better than those of the base model, in fact they exceed the target of 63% accuracy and precision offered by this simple model.

This model, according to the figures in Table 15, is correct 66% of the time. In addition, something interesting is that the probability that it is correct when classifying a sample as class one is 69%.

These improvements have been achieved thanks to the increased ability of the algorithm to classify negative class samples (probably as a consequence of having balanced classes), thus also improving the sorting accuracy for positive samples.

1.14 SUPPORT VECTOR MACHINES

1.14.1 DESCRIPTION OF THE TECHNOLOGIES

The main idea behind these classifiers is based on the *Perceptron* model explained in *Chapter 2*. However, in this case, the aim is not to minimize the classification error, but to maximize the margin, i.e., to maximize the distance between the hyperplane defined by the linear discrimination function and the samples used to train the classifier [12].

The concept of SVM is very well illustrated in the following image, taken from the book *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* [25]:

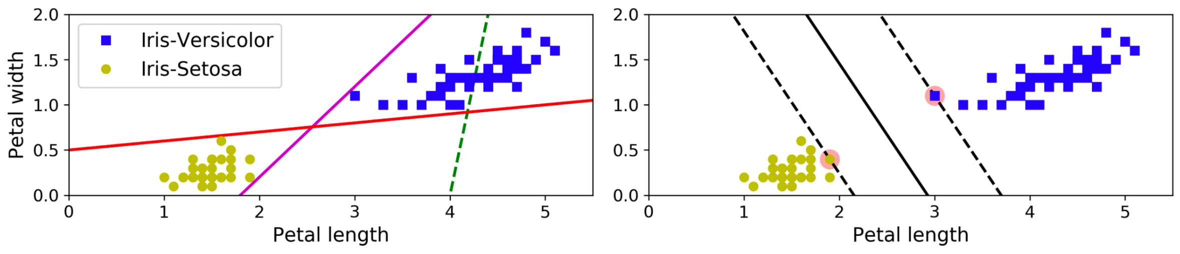


Figure 29. SVM Concept (Source: Book [25])

As can be seen in the image, the aim is to obtain the best of the possibilities for the linear discrimination function. Thus, being farther away from all the training samples, it will probably be also far away from the test set samples.

The main difference compared to the perceptron is that, in this case, the function to be maximized is as follows [12]:

$$Margin = \frac{\overline{w}^T(\bar{x}_+ - \bar{x}_-)}{\|\overline{w}\|}$$

Being the margin, the distance between the two different hyperplanes that are shown in Figure 29.

In this case, the *Scikit-Learn* hyperparameter c is not used to control regularization, but a similar concept, that of soft margin classification. This type of classification is used when the data are not perfectly linearly separable, as in the case of the stock market. Using the hyperparameter *c*, we control the convergence of the optimization for data of this type. The smaller this hyperparameter is, the less we penalize classification errors [12].

The Kernel Method

The kernel method is a little trick that facilitates the use of *SVM* on non-linearly separable data. It consists of "creating non-linear combinations of the original features to project them into a higher dimensional space, where they become linearly separable" [12]. Using this trick allows to create non-linear separations between the data.

Scikit-Learn offers the possibility to implement this type of models. The hyperparameter used to tune the degree of adjustment of the decision boundary is γ, which must be optimized.

1.14.2 MODEL GENERATION

In this section different models will be trained using *SVM* and *Kernelized SVM*. The main objective of this section is to improve *Model 7*, obtained with logistic regressions. For this reason, only the training process of a basic model (to understand the methodology) will be shown, followed by the optimal model results, which will also be explained.

The first basic model is trained, i.e., a model containing the entire data frame. Considering that, as it has been shown so far, linear classifiers are generally very susceptible to unscaled data, it has been decided to use the already standardized database (*df_stand* and *df_out_stand*) from the very beginning.

A first model that uses all data is, then, trained. As the number of hyperparameters to be optimized is now larger than that of the logistic regression model, it will not be possible to optimize them using the validation curves. The *grid sea*rch technique will then be used, which consists of calling an algorithm that, based on a list of possible values that the hyperparameters can take, executes and compares all the possible solutions in order to choose the best one [25].

The following code is then executed, which calls the *GridSearchCV() Scikit-Learn* function. The basis of the code is taken from the *Python Machine Learning* book [12]:

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

p_r = [0.0001, 0.001, 0.01, 0.1, 1.0, 10, 100]

p_g = [{'C': p_r, 'kernel': ['linear']}, {'C': p_r, 'gamma': p_r, 'kernel':
   ['rbf']}]

gs = GridSearchCV(estimator = SVC(random_state = 1), param_grid = p_g, scoring =
   'accuracy', cv = 10)

gs = gs.fit(X_train_out_stand, y_train_out_stand)

print(gs.best_score_.round(4))
print(gs.best_params_)
-Output: 0.5345 {'C': 0.001, 'gamma': 0.1, 'kernel': 'rbf'}
```

It can be observed that *Kernelized SVM* offer a better performance. The main reason is that, as mentioned above, they are prepared to manage non-linearly separable data.

Again, as with the validation curves, it should be noted that this new way of optimizing the hyperparameters also uses *cross validation*. Therefore, knowing that it is not a suitable method to test the models, it is convenient to check the performance of the classifiers with the test set. The following code is executed for this purpose:

```
from sklearn.linear_model import SVC

svm = SVC(kernel = 'rbf', C = 0.001, gamma = 0.1, random_state=1)
svm.fit(X_train_out_stand, y_train_out_stand)

pred = svm.predict(X_test_stand)
```

In this case, it should be noted that, for this new basic model (*Model 8*) with *Kernelized SVM*, it is convenient to use data without outliers to train it, since the performance they offer is better.

Model 8 is the worst so far. Nevertheless, it will be analyzed. Its confusion matrix is presented below.

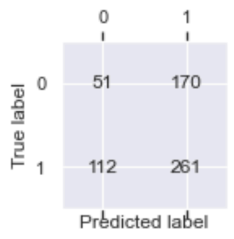


Figure 30. Confusion Matrix of Model 8

Its metrics are also shown in the following table.

K-SVM Model 8	
Accuracy (***ACC***)	0.53
Precision (**PRE**)	0.61
Recall (**REC**)	0.70
$\boldsymbol{F_1}$	0.65
Negative Recall (**N-REC**)	0.23
Negative Precision (**N-PRE**)	0.31

Table 16. Evaluation Metrics of Model 8

Indeed, it is a model that barely gets half of its predictions right. Therefore, it is clearly below Model 0 and Model 7 in terms of performance.

Given the low accuracy of the model, it could be over-fitted. To check this, it is tested again, but this time, using the training data. The following metrics were obtained:

K-SVM Model 8 with the Training Set	
Accuracy (***ACC***)	0.66
Precision (**PRE**)	0.57
Recall (**REC**)	0.21
$\boldsymbol{F_1}$	0.77
Negative Recall (**N-REC**)	0.23
Negative Precision (**N-PRE**)	0.31

Table 17. Evaluation Metrics of Model 8 using the Training Set

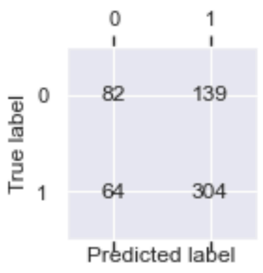
In spite of being a model with already optimized hyperparameters, it can be verified that it is overfitted, as it has an accuracy of 67% with the training set. To try to solve this problem, from now on simpler models will be proposed, reducing the number of features to be used.

1.14.2.1 Other Models

The models were trained according to *Pearson's coefficient*, using *PCA*, etc. Most of these, showed very similar performance to their counterparts trained using logistic regressions.

One model significantly improved the results provided by *Model 0*. This was a *Kernelized SVM* classifier trained with the most significant variables (obtained by training models based on Pearson's coefficient and p-value) of the data frame *df_out_stand*. In addition, a *PCA* was applied to these variables, reducing the dimension from 7 to 3.

The confusion matrix of this new classifier (*Model 9*) is as follows:



True label \ Predicted label	0	1
0	82	139
1	64	304

Table 18. Confusion Matrix of Model 9

Its valuation metrics are:

K-SVM Model 9	
Accuracy (***ACC***)	0.65
Precision (**PRE**)	0.68
Recall (**REC**)	0.83
$\boldsymbol{F_1}$	0.75
Negative Recall (**N-REC**)	0.37
Negative Precision (**N-PRE**)	0.56

Table 19. Evaluation Metrics of Model 9

This model is already acceptable, although it still does not improve the performance of the seventh model.

1.14.2.2 Modeling by Training Set Tuning

With the information shown in the table above, it can be concluded that the *K-SVM* trained classifiers also have a slight tendency to classify the data positively. In fact, Model 9 only classifies 37% of the '*0 classes*' well. With this in mind, it may again be a good practice to try to eliminate the *grey data* and balance the weight the classes in the training set.

Following the scheme shown in Figure 27, a model that can be considered optimal for the purposes of this project is reached.

The latter classifier is modeled as follows:

```
from sklearn.svm import SVC
from sklearn.decomposition import PCA

pca = PCA(n_components = 4)
X_train_pca = pca.fit_transform(X_train_out_stand_T2_bal.iloc[:,[2,8,10,32,34,43,
60]])
X_test_pca = pca.transform(X_test_out_stand_T2_bal)

model = SVC(kernel = 'rbf', C = 0.1, gamma = 0.1, random_state=1)
model.fit(X_train_pca, y_train_stand)
```

The balancing of the classes through the elimination of the gray data paid off. With the above code, the best model of the work is obtained. It is a *K-SVM* (*Model 10*) that has as inputs a series of variables (optimized based on the correlation with the class) to which a *PCA* that reduces the number of features to 4 has been applied. For the sake of completeness, this model was obtained by applying a $T = 2\%$ filter, which is the one that gives the best results also in this case.

The confusion matrix of *Model 10* is the following:

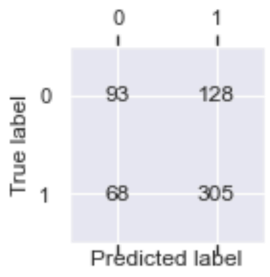


Figure 31. Confusion Matrix of Model 10

Finally, its metrics are shown:

K-SVM Model 10	
Accuracy (***ACC***)	0.67
Precision (**PRE**)	0.70
Recall (**REC**)	0.81
F_1	0.75
Negative Recall (**N-REC**)	0.42
Negative Precision (**N-PRE**)	0.58

Table 20. Evaluation Metrics of Model 10

This is the best model so far, with a 67% accuracy rate, 4 points higher than the basic model. In addition, the precision when classifying positive classes is 70%, 7 points higher than that of *Model 0*.

CHAPTER 7. ANALYSIS OF THE RESULTS

In the previous chapter, following a trial-and-error methodology, two good predictive models have been attained, one based on *Logistic Regressions* and the other using *Kernelized SVM*. In this chapter, the main differences between them will be analyzed in order to choose the best one.

1.15 MODEL WITH LOGISTIC REGRESSION

The finest model obtained with *LR* is *Model 7*. The following figure schematizes it and includes its most important characteristics and metrics:

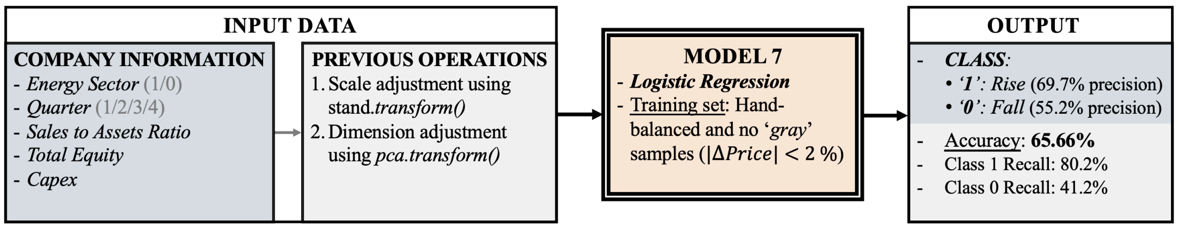


Figure 32. Schematic diagram of Model 7

1.16 MODEL WITH KERNELIZED SUPPORT VECTOR MACHINE

The top model acquired with *K-SVM* is *Model 7*. The next image schematizes it and covers its most relevant features and metrics:

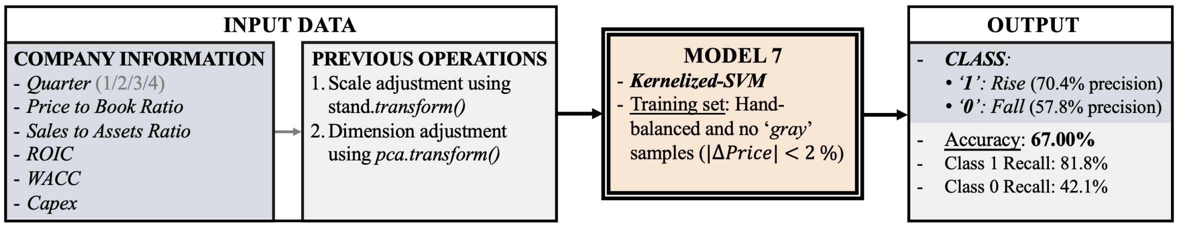


Figure 33. Schematic diagram of Model 10

1.17 COMPARISON OF THE MODELS

The table below lists the main metrics that will serve as a guide to compare the models.

Model 7 *(LR)*		***Model 10*** *(K-SVM)*	
Accuracy	65.7%	Accuracy	67.0%
Precision	69.7%	Precision	70.4%
Recall	80.2%	Recall	81.8%
F_1	74.6%	F_1	75.7%
Precision (class 0)	55.2%	Precision (class 0)	57.8%
Specificity	41.2%	Recall (class 0)	42.1%

Table 21. Metrics of the Best Models

When contrasting the evaluation metrics of both models, it is clear that *Model 10* is better overall. The bar chart below shows both the differences and the improvements (differences presented as a multiplier) that this classifier provides in regard to *Model 7*.

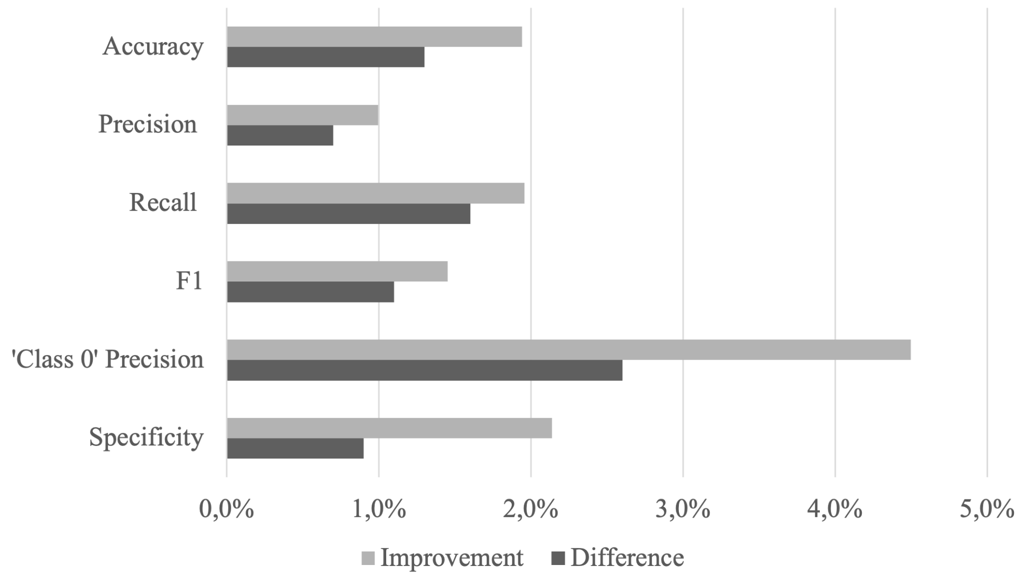


Figure 34. Improvements Provided by Model 10 Compared to Model 7

Looking at the data shown in the graph, it can be considered that *Model 10* is superior on every account. However, an analysis of each of the metrics will be carried out to get a greater understanding of both models.

1.17.1 EVALUATION METRICS

One method to compare the models is on the basis of the accuracy they offer, i.e., looking at percentage of success they have in predicting the training set samples. *Model 7*, in this case, loses out, since it has an accuracy of 65.66%. In contrast, *Model 10* has a hit rate of 67.00%, which means that it gets 2% more correct classifications than the first one. In terms of this evaluation metric, both models outperform *Model 0* (the one that classifies all stocks in the positive class). Therefore, it can be concluded that either of them are recommended over the random classification model.

Another key metric is classification precision; here, again, *Model 10* outperforms *Model 7*. Using the former, 70.4% accuracy is obtained in classifying positive cases (cases of stock rises), which is an improvement of close to 1% over the classifier that uses logistic regression. Hence, it can be stated that with the best model, out of every 100 companies whose shares are predicted to rise, 70 will actually rise.

In terms of negative precision, i.e., the classification accuracy for the '0' classes, *Model 10* is also better, incorporating an improvement of approximately 4.5% over the same metric for *Model 7*. Following the same example, out of every 100 stocks for which *Model 10* predicts a rise, only 58 will rise.

As for recall (or sensitivity), it should be remarked that, as expected, Model 10 improves Model 7, recognizing 81.8% of the positive classes compared to 80.2% of the latter. If this concept is extrapolated to the negative cases, one can talk then of specificity, where again the classifier that uses K-SVM performs better, correctly identifying 42% of the dips that occur in the test set.

1.17.2 ANALYSIS OF POSSIBLE ERRORS

It is important to examine which figures are the most interesting ones for the project. For this purpose, the two types of errors that models can make in this case will be presented:

- *Mistake 1*: recommending to buy the stock when it will later fall.
- *Mistake 2*: recommending not to buy the stock when it will later rise.

From a financial point of view, it is clear that *Mistake 1* is the most serious error, as it is the one that can cause money to be lost. *Mistake 2*, however, only implies a lost opportunity. Therefore, it should be emphasized that it will always be better for the classifier to recommend not to buy a stock even if it finally rises.

The important matter in this case will therefore be that the model has the minimum possible number of false positives, i.e., that it has a high precision (for the positive classes) and a high specificity. Both algorithms meet the first requirement, having precisions close to 70%, but they fail somewhat more in specificity, being close to 42%, i.e., they classify 58% of the 0 cases as 1.

1.17.3 COMPARISON WITH 'MODEL 0'

As mentioned above, *Model 10* is the best of the two models obtained. If we compare the two most important metrics, i.e., the specificity and accuracy of this model with those of Model 0, we obtain the following graph:

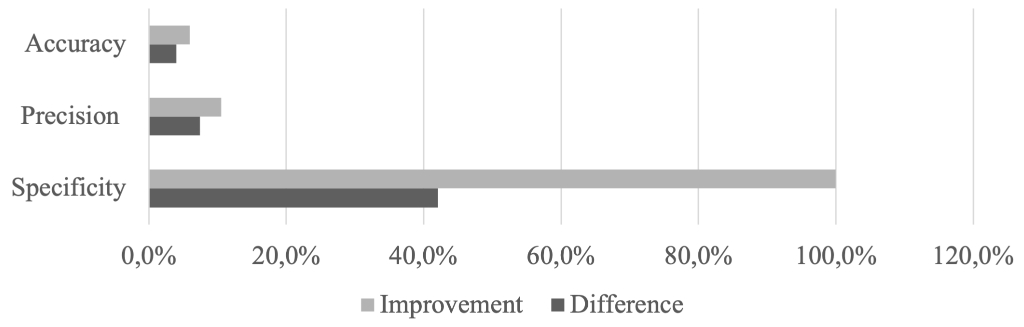


Figure 35. Improvements Provided by Model 10 Compared to Model 0

Although it has been said that *Model 10* does not have a good specificity, if it is compared with that of *Model 0*, things change. This metric goes from having a value of 0 to 42%. In terms of accuracy, it improves by 10.5%.

1.18 THE METHODOLOGY AS A RESULT

To arrive at models 7 and 10, a classifier training methodology has been applied that has been proven to work. A brief outline has been drawn up showing the steps that have been followed:

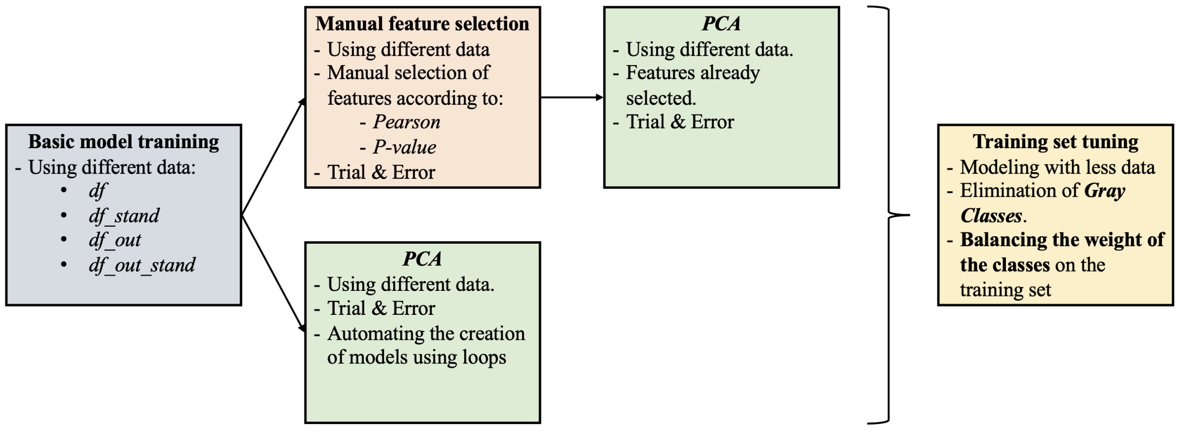


Figure 36. Summary of the Methodology Used

It should also be noted that not all datasets proved to be equally effective in training the models. Standardized data turned out to be much more useful than unscaled data for both *Logistic Regressions* and *SVM*. As for partial outlier removal, it only proved useful for *SVM* training.

Looking at the overall performance of *Model 10*, one might consider that it is not a good model, since it does not even exceed 70% accuracy; however, one should be aware that the research has been carried out with linear classifiers. These classifiers work perfectly well when the data are linearly separable through hyperplanes. Using *logistic regressions*, it was shown that the linear separability of the data was very poor. Therefore, better results were obtained with *K-SVM*, which allow the use of less linearly separable data using the Kernel trick.

CHAPTER 8. **CONCLUSIONS AND FUTURE WORKS**

1.19 CONCLUSIONS

The main objective of this project was to obtain a series of predictive models for the stock market using linear and kernelized classifiers. The problem was presented as a binary classification task that would allow us to distinguish between two types of stocks: those that will rise and those that will fall in the next quarter.

This major goal has been achieved with relative success, having obtained two models that can be considered acceptable and relatively good, one for each type of classifier used.

A first model was thus attained using logistic regression, a technique that seeks to maximize the probabilities of class membership for each sample. This model had an accuracy of 66.5%. The results were later improved by obtaining a new model, this time using *K-SVM*. The classification accuracy thus rose to 67%. It should be noted that both models outperformed another "basic" model that classified all shares in a single class, thus fulfilling one of the main objectives of the project.

It must be considered that during the analysis of the results, it has been possible to conclude that both models offered a good performance. From a financial point of view, what is sought for a classifier of this type is to have high accuracy and specificity, avoiding as much as possible the cases of false rise predictions, which are the ones that would lead to lose money. In this case the best model offered an accuracy of 70% and a specificity of 42%. This last value could be regarded as insufficient; however, it introduces a 100% improvement over the specificity of the "basic" model, thus making it a good evaluation metric.

Concerning the training of these models, it is worth mentioning, first of all, the importance of the data pre-processing phase in the correct execution of the project. As the values to work with were very different in terms of scales, it was decided to standardize the data, in the belief that, in this way, the models would offer better performance, avoiding the prioritization of some variables over others. After obtaining the classifiers, this hypothesis proved to be true, securing better metrics with those models that used standardized data.

On the other hand, it was observed that there were a large number of outliers in the database. Almost 40% of the information had a *Z-Score* greater than 3. Therefore, it was deemed convenient to create a second data frame in which the most extreme outliers, those with a *Z-Score* greater than 5, were eliminated. This technique paid off in the *Kernelized-SVM* modeling, improving the models. However, it did not offer any significant enhancement over the database with all outliers in the logistic regression modeling. It can consequently be concluded that, since there are so many outliers, the information provided by the outliers can be considered essential in the development of predictive stock market models using logistic regression.

Finally, regarding the methodology used in the training of these models, it should be said that using all 72 variables of the database greatly reduces the accuracy of the models. For this reason, a manual selection of the variables to be used in them was applied according to their correlation with the class, which, together with data dimensionality reduction techniques such as *Principal Component Analysis*, yielded results.

Using only these two data selection techniques did not lead to good models. Analyzing the results, it was deduced that the algorithm offered very low specificity values. Trying to solve this, it was decided to eliminate from the training set those samples considered as "gray", i.e. to remove the information relative to those samples that had very small price variations. In addition, in order for the learning algorithms to better classify the negative samples, the weights of the classes in the training set were balanced. With these two modifications, the best results were acquired.

It can accordingly be concluded that the removal of gray cases from the training set, together with obtaining a class-balanced database, have proved to be two useful techniques for training predictive stock market models with linear and kernelized classifiers.

1.20 FUTURE WORKS

After carrying out this project, the following improvements, or future work, may be proposed:

- Obviously, after observing that stock market data are usually not linearly separable, the use of other types of classifiers could certainly produce better results. Therefore, the possibility of carrying out the project in the future using decision trees or random forests is proposed.
- On the contrary, using a larger amount of data on more companies, not only belonging to the S&P500, to obtain predictive models could be another exercise to be conducted in the future.

CHAPTER 9. **BIBLIOGRAPHY**

[1] Ertel, W., & Black, N. T. (2018). *Introduction to Artificial Intelligence (Undergraduate Topics in Computer Science)* (2nd ed. 2017 ed.). Springer.

[2] PwC, Rao, A., & Verweij, G. (2017). Sizing the prize. *What's the real value of AI for your business and how can you capitalise?* PwC. https://www.pwc.com/gx/en/issues/analytics/assets/pwc-ai-analysis-sizing-the-prize-report.pdf.

[3] *Andrew Ng: Why AI Is the New Electricity*. (2021, May 4). Stanford Graduate School of Business. Retrieved April 2021, from https://www.gsb.stanford.edu/insights/andrew-ng-why-ai-new-electricity.

[4] Dixon, M. F., Halperin, I., & Bilokon, P. (2021). *Machine Learning in Finance: From Theory to Practice* (2020 ed.). Springer.

[5] European Central Bank. (2021a, July 14). *Eurosystem launches digital euro project* [Press release]. Retrieved April 2021, from https://www.ecb.europa.eu/press/pr/date/2021/html/ecb.pr210714%7Ed99198ea23.en.html.

[6] Vallejo, C., Torres, Ó., & Pereira, Ó. T. (2012). *Manual de la inversión en bolsa* (Vol. 1). Inversor Ediciones.

[7] Smart, S. B., & Zutter, C. J. (2019). Analyzing Common Stocks – Fundamental Analysis. *Fundamentals of Investing, Global Edition* (pp. 296-313). (14th ed.). Pearson.

[8] McKinsey & Company Inc., Koller, T., Goedhart, M., & Wessels, D. (2020). *Valuation: Measuring and Managing the Value of Companies* (Wiley Finance) (7th ed.). Wiley.

[9] Ross, S. A., Westerfield, R. W., Jaffe, J., & Jordan, B. D. (2017). Risk, Cost of Capital and Valuation. In *Corporate Finance: Core Principles and Applications* (5th ed., pp. 357–389). McGraw-Hill Education.

[10] Rich, E. (1983). *Artificial Intelligence (McGraw-Hill series in artificial intelligence) 1st edition by Rich, Elaine* (1983) Hardcover (1st ed.). McGraw-Hill.

[11] The Special Interest Group of AI (SIGAI), CWI, LU, RU, RUG, TUD, TUe, UM, UT, UU, UvA, UvT, & VU. (2018, September). *Dutch Artificial Intelligence Manifesto*. http://ii.tudelft.nl/bnvki/wp-content/uploads/2018/09/Dutch-AI-Manifesto.pdf.

[12] Mirjalili, V., & Raschka, S. (2020). *Python Machine Learning* (Spanish Edition) (1st ed.). Marcombo.

[13] Murty, M. N., & Raghava, R. (2016). *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks* (SpringerBriefs in Computer Science) (1st ed. 2016 ed.). Springer.

[14] Kearns, M., & Nevmyvaka, Y. (2013). *Machine Learning for Market Microstructure and High Frequency Trading*. In M. O'Hara, D. Easley, & M. Lopez de Parado, High Frequency Trading - New Realities for Traders, Markets and Regulators. Risk Books.

[15] Rühr, A., (2020). *Robo-Advisor Configuration: An Investigation of User Preferences and the Performance-Control Dilemma*. Association for Information Systems, Research Papers. 94. https://aisel.aisnet.org/ecis2020_rp/94

[16] Milosevic, N. (2016, March). *Equity forecast: Predicting long term stock price movement using machine learning*. https://arxiv.org/abs/1603.00751

[17] Kumar, M., & Thenmozhi, M. (2006, January). *Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest*. Indian Institute of Capital Markets 9th Capital Markets Conference Paper. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=876544.

[18] Timor, M., Dinçer, H., & Emir, S. (2012, January). *Performance comparison of artificial neural network (ANN) and support vector machines (SVM) models for the stock selection problem: An application on the Istanbul Stock Exchange (ISE) - 30 index in Turkey*. African Journal of Business Management Vol. 6(3), (pp. 1191–1198). http://www.academicjournals.org/AJBM.

[19] Ou, P. (2019, December). *Prediction of Stock Market Index Movement by Ten Data Mining Techniques*. Modern Applied Science, Vol.3(12). https://doi.org/10.5539/mas.v3n12p28.

[20] THE 17 GOALS | *Sustainable Development*. (n.d.). United Nations (UN). Retrieved August 20, 2021, from https://sdgs.un.org/goals.

[21] Mukhiya, S. K., & Ahmed, U. (2020b). *Hands-On Exploratory Data Analysis with Python: Perform EDA techniques to understand, summarize, and investigate your data*. Packt Publishing.

[22] Tallon-Ballesteros, A. J., & Riquelme, J. (2014, July). *Deleting or Keeping Outliers for Classifier Training?* https://core.ac.uk/download/pdf/51405318.pdf

[23] Müller, A. C., & Guido, S. (2016). *Introduction to Machine Learning with Python: A Guide for Data Scientists* (1st ed.). O'Reilly Media.

[24] Carretero, C. R., & Guillén, R. J. (2019). *Estadística (Papel + e-book): Yo no soy mala, me han dibujado así* (Tratados y Manuales de Derecho) (Spanish Edition) (2nd ed.). Civitas.

[25] Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). O'Reilly Media.

APPENDIX A. FINANCIAL VARIABLES USED IN THE PROJECT

The financial variables that make up the original database used to train the models are listed below.

Stock information data

In order to analyze the fundamentals of a company and decide whether to buy its shares, it is essential to know the characteristics of its shares. The variables that can be included in this category are (the number of the variable is shown in square brackets, this value will be used in the codes):

- *Book value per share [10].*
- *Market capital [11].*
- *Earnings per share* (*eps*) [12].
- *Earnings per share growth* (*eps growth*) [13].
- *Revenue per share [14].*
- *Dividends per share [15].*

Financial statement data

- *Assets [54].*
- *Current assets [55].*
- *Liabilities [56].*
- *Current liabilities [57].*
- *Total equity [58].*
- *EBITDA [59].*
- *CAPEX [60].*
- *Working capital [61].*
- *Sales turnover [62].*
- *Free cash flow [63].*
- *Operating cash flow [64].*

- *Tax rate [66].*

Financial ratios

Efficiency ratios

- *ROIC [34].*
- *ROCE [35].*
- *ROE [36].*
- *ROA [37].*
- *EBITDA to revenue [48].*
- *Operating margin [49].*
- *Gross margin [50].*
- *Profit margin [51].*

Leverage ratios

- *Long term debt to total assets [39].*
- *Net debt to EBITDA [40].*
- *Total debt to total equity [41].*
- *Financial leverage [52].*
- *FCF yield [53].*

Liquidity ratios

- *Current ratio [67].*
- *Quick ratio [68].*

Comparable multiples

- *P/E [16].*
- *P/CF [18].*
- *P/B [20].*
- *P/S [22].*

- *EV/FCF [24].*
- *EV/EBITDA [26].*
- *EV/Sales [28].*
- *EV/Invested capital [30].*
- *Sales/Assets [32].*

WACC ratios

- *WACC [43].*
- *K_e [44].*
- *Equity weight (%) [45].*
- *K_d [46].*
- *Debt weight [47].*

Other ratios

- *ROIC/WACC [38].*
- *Working capital/Sales [42].*
- *Analyst recommendation [69].*
- *Insiders(%) [70].*

APPENDIX B. COST OF EQUITY AND COST OF DEBT ACCORDING TO THE *CAPM*

The CAPM is an asset valuation model through which both the cost of capital and the cost of debt of a company can be estimated as follows [9]:

Cost of Equity

$$k_e = R_{free} + \beta(R_m - R_{free})$$

Where:

R_{free} is the *risk free rate*.

R_m is the average *market rate of return*.

β is the volatility of the market.

Cost of Debt

$$k_d = R_{free} + C_{spread}$$

Where:

R_{free} is the *risk free rate*.

C_{spread} is the *credit spread*, which is the difference between the interest paid as debt by the company and that paid for a risk-free reference security.

It should be noted that this is the pre-tax cost of debt. The effect of taxes on the *WACC* is already included in the equation *E. 1*.