MÁSTER UNIVERSITARIO EN BIG DATA.

TECNOLOGÍA Y ANALÍTICA AVANZADA

TRABAJO FIN DE MÁSTER

SUBTIPOS COGNITIVOS PARA PREDECIR EL RESULTADO EN EL TRASTORNO AFECTIVO: UN ENFOQUE DE APRENDIZAJE AUTOMÁTICO

Autor: Jairo Silvera Sarmiento

Directora: Ana Laguna Pradas

Madrid

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

SUBTIPOS COGNITIVOS PARA PREDECIR EL RESULTADO EN EL TRASTORNO AFECTIVO: UN ENFOQUE DE APRENDIZAJE AUTOMÁTICO

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

curso académico 2021/22 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.

Fdo.: Jairo Silvera Sarmiento Fecha: 28/septiembre/2022

Autorizada la entrega del proyecto

LA DIRECTORA DEL PROYECTO

Fdo.: Ana Laguna Pradas Fecha: 28/septiembre/2022

MÁSTER UNIVERSITARIO EN BIG DATA.

TECNOLOGÍA Y ANALÍTICA AVANZADA

TRABAJO FIN DE MÁSTER

SUBTIPOS COGNITIVOS PARA PREDECIR EL RESULTADO EN EL TRASTORNO AFECTIVO: UN ENFOQUE DE APRENDIZAJE AUTOMÁTICO

Autor: Jairo Silvera Sarmiento

Directora: Ana Laguna Pradas

Madrid

Agradecimientos

A mi madre, padre y familia, profesores, a mis profesoras del proyecto Ana, Sara y Cristina por toda su ayuda, a mi director y coordinador del máster gracias por su apoyo y motivación, a mis compañeros muchas gracias a todos.

SUBTIPOS COGNITIVOS PARA PREDECIR EL RESULTADO EN EL TRASTORNO AFECTIVO: UN ENFOQUE DE APRENDIZAJE AUTOMÁTICO.

Autor: Silvera Sarmiento, Jairo.
Directora: Laguna Prada, Ana.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

El proyecto consiste en analizar una base de datos con resultados de pacientes con trastornos psiquiátricos que presentan una serie de síntomas, para que aplicando técnicas de Machine Learning se prediga si un paciente debe iniciar un tratamiento o no, esta base de datos está estructurada y contiene datos basados en pacientes. Los métodos de aprendizaje automático para la predicción y detección de patrones son cada vez más frecuentes en la investigación psicológica. Las técnicas de ML son muy útiles y tienen un alto impacto en la psicología, especialmente en la comprensión de patrones ocultos en el comportamiento de las personas. Aunque la psicología, como ciencia que construye teorías, tarda en adoptar el ML como herramienta para analizar resultados experimentales. Algunos resultados de los experimentos son tratados y analizados con técnicas estadísticas inferenciales *(Orrù, Monaro, Conversano, Gemignani, & Sartori, 2020)* pero con los avances obtenidos últimamente se han realizado estudios con grandes resultados.

Palabras clave: Psicología, tratamientos, Machine Learning, Python, datos.

1. Introducción

Los trastornos afectivos o del estado de ánimo son un conjunto de trastornos psiquiátricos, los principales tipos de trastornos afectivos son la depresión y el trastorno bipolar. Los síntomas varían según el individuo y pueden ir de leves a graves También hay otros tipos de trastorno del estado de ánimo como: Trastorno depresivo mayor, Trastorno afectivo estacional (TAE), Trastorno ciclotímico, Trastorno depresivo persistente (distimia) *(Mayoclinic, 2021)*. Entre los síntomas comunes a cada una de las diferentes patologías se encuentran la tristeza prolongada, la falta de interés por las actividades normales, los cambios de humor inusuales y crónicos.

Trastorno afectivo o trastorno del estado de ánimo es un conjunto de trastornos psiquiátricos, Los principales tipos de trastornos afectivos son la depresión y el trastorno bipolar. Los síntomas varían según el individuo y pueden ir de leves a graves. Entre los síntomas comunes a cada una de las diferentes patologías están la tristeza prolongada, la falta de interés por las actividades normales, los cambios de humor inusuales y crónicos.

Hay muchas causas de esta condición, las más comunes son los neurotransmisores o sustancias químicas del cerebro las principales causas cuando hay un desequilibrio de estos, el cerebro no funciona correctamente. algunos eventos o tragedias personales, el

consumo de alcohol o drogas también pueden causar el trastorno del estado de ánimo *(Healthline, 2020)*.

Recientemente, debido a la epidemia del COVID-19 En el primer año de la pandemia, la prevalencia mundial de la ansiedad y la depresión aumentó un enorme 25% una de las principales explicaciones del aumento es el estrés sin precedentes causado por el aislamiento social resultante de la pandemia, entre varias causas están la soledad, el miedo a contraer la enfermedad o la pérdida de seres queridos *(who, 2022)*.

2. Definición del proyecto

Este es un proyecto de Machine Learning, donde se aplicaron funciones y utilidades propias de esta técnica. Este proyecto se basa en conjunto con las técnicas de Aprendizaje Automático y datos donde los datos y su manipulación son muy importantes de acuerdo con la definición de Oxford los datos son hechos o información, especialmente cuando se examinan y utilizan para averiguar cosas o para tomar decisiones. Por lo tanto, los datos se miden, se recogen, se comunican y se analizan, visualizándose mediante gráficos, imágenes u otras herramientas de análisis.

Los datos tienen diferentes fuentes, pueden ser generados por humanos, máquinas o una combinación de ambos. Aquí se tratan los datos generados por humanos, para un mejor procesamiento y análisis de los proyectos de Machine Learning, en este proyecto se aplicaron tecnologías propias del Data Science.

3. Descripción del modelo/sistema/herramienta

El origen es un archivo csv con filas y columnas. Las columnas son diversos datos relacionados con datos clínicos, resultados de pruebas, datos psicológicos, la variable objetivo es `number_treatments` según esto la predicción consistirá en llevar a los pacientes a tratamiento. Los modelos aplicados son la *Regresión Logística*, que es un algoritmo de clasificación supervisado. En un problema de clasificación, la variable objetivo y, sólo puede tomar valores discretos para un conjunto dado de características X los valores de salida son SI y NO. El otro modelo es el **Random Forest**, que es una técnica de conjunto capaz de realizar tanto tareas de regresión como de clasificación con el uso de múltiples árboles de decisión y técnicas como *bagging*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Studiennum	date_t1	date_t2	rehab_days	rehab_week	rehab_week	birthdate	age	sex	diagnoses	group	group_affect	group_unipc	height_cm	height_m	weight
2	1	4/13/2015	5/21/2015	38	5	2	01.02.1959	56,28	2	F32, I44 E66	1	1	1	183,9	1,84	102
3	2	4/13/2015	5/21/2015	38	5	2	06.03.1984	30,86	1	F13, F43	3	0	0	166,8	1,67	56,45
4	3	4/13/2015	5/19/2015	36	5	2	6/14/1965	49,83	2	F33, R73	1	1	1	178,2	1,78	108,4
5	4	4/13/2015	5/21/2015	38	5	2	8/31/1960	54,62	2	F32,	1	1	1	183,6	1,84	89,95
6	5	4/13/2015	5/21/2015	38	5	2	2/18/1943	72,15	1	F32, I10 R55	1	1	1	155,3	1,55	69,05
7	6	4/13/2015	5/21/2015	38	5	2	06.06.1963	51,85	2	F43, Z73 E66	3	0	0	164,2	1,64	74,25
8	7	4/13/2015	5/21/2015	38	5	2	12.11.1963	51,34	1	F33, E66 I10	1	1	1	167,5	1,67	85,15
9	8	4/13/2015	5/21/2015	38	5	2	4/13/1967	48	1	F43, E11 E78	3	0	0	172	1,72	86,35
10	9	4/13/2015	5/21/2015	38	5	2	11/20/1957	57,4	1	F43, F13 K59	3	0	0	167,4	1,67	52,3
11	10	4/13/2015	5/21/2015	38	5	2	10/22/1957	57,48	1	F33, M81	1	1	1	167,3	1,67	64,15
12	11	4/13/2015	5/21/2015	38	5	2	7/18/1958	56,74	2	F33, Z73	1	1	1	158,6	1,59	56,9
13	12	4/13/2015	5/21/2015	38	5	2	01.07.1955	60,27	1	F33, M35 K2	1	1	1	172,2	1,72	63,55
14	13	4/14/2015	5/22/2015	38	5	2	12.08.1951	63,35	2	F31, E66 E78	1	1	1	181,4	1,81	95,6
15	14	4/14/2015	5/22/2015	38	5	2	6/30/1961	53,79	2	F43, E66 I10	3	0	0	183,3	1,83	104,15
16	15	4/14/2015	5/22/2015	38	5	2	04.05.1959	56,02	2	F32, I10 E66	1	1	1	172,7	1,73	98,45
17	16	4/14/2015	5/22/2015	38	5	2	12.12.1947	67,34	2	F31, E66	2	1	0	180,4	1,8	100,35
18	17	4/14/2015	5/22/2015	38	5	2	5/19/1953	61,9	2	F43, I10	3	0	0	170,9	1,71	82,7
19	18	4/14/2015	5/22/2015	38	5	2	3/31/1956	59,04	1	F32, M51 G2	1	1	1	162	1,62	52,2
20	19	4/14/2015	5/22/2015	38	5	2	07.07.1963	51,77	1	F32,	1	1	1	161,7	1,62	53,2
21	20	4/14/2015	5/22/2015	38	5	2	4/28/1969	45,96	1	F32, Z73 E78	1	1	1	166,8	1,67	62,85
22	21	4/14/2015	5/22/2015	38	5	2	05.04.1957	57,94	1	F32, I83	1	1	1	172,9	1,73	55,7

Imagen 1 – Archivo csv con datos clínicos y psiquiátricos

4. Resultados

Inicialmente, la variable objetivo del conjunto de datos tenía valores atípicos de desequilibrio de clase y valores perdidos que daban resultados inexactos. El conjunto de datos también tiene muchas variables que son inútiles para el modelo, por lo que había que hacer una selección de características.

Después de varios intentos desarrollados para obtener la mejor solución para el objetivo la mejor solución para el modelo fue la implementación de un modelo de regresión logística con escalamiento, sin variables correlacionadas, sin valores perdidos y un modelo de Random Forest también hizo un gran rendimiento.

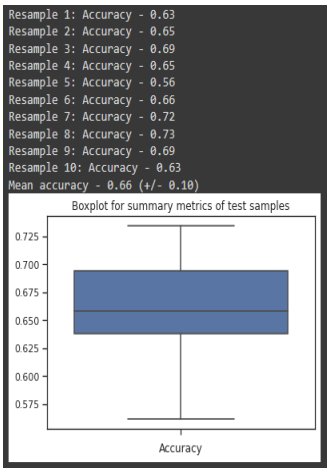


Imagen 2 – Accuracy del modelo de Regresión Logística

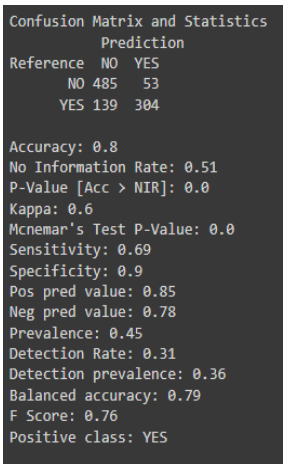


Imagen 3 – Resultados de la matriz de confusión del modelo Random Forest

5. Conclusiones

Se implementaron varios modelos de aprendizaje automático y se analizaron con diferentes métodos de extracción de características para encontrar la mejor solución posible. Los proyectos de Aprendizaje Automático requieren un flujo de trabajo por etapas, que es lo que se hizo aquí, cada etapa puede tener pocas o varias subtareas, todo depende del objetivo y las fuentes de trabajo. La interpretación de las métricas de precisión de la predicción (por ejemplo, como alta, o baja precisión) depende del contexto y requiere experiencia en el campo ya que hay muchas variables y puntos de referencia que los psicólogos utilizan y que en conjunto permiten establecer un diagnóstico del paciente.

Estos modelos dan una visión importante a la clínica y a otras variables importantes para que futuros estudios psicológicos puedan basar sus estudios y dar más énfasis a las variables aquí mencionadas.

6. Referencias

Healthline. (2020, April 1). *healthline.com*. Retrieved from https://www.healthline.com/health/affective-disorders

Mayoclinic. (2021, October 29). *mayoclinic.org*. Retrieved from https://www.mayoclinic.org/diseases-conditions/mood-disorders/symptoms-causes/syc-20365057

Orrù, G., Monaro, M., Conversano, C., Gemignani, A., & Sartori, G. (2020, January 10). Retrieved from https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02970/full

who. (2022, March 2). *World Health Organization*. Retrieved from who.int: https://www.who.int/news/item/02-03-2022-covid-19-pandemic-triggers-25-increase-in-prevalence-of-anxiety-and-depression-worldwide

COGNITIVE SUBTYPES TO PREDICT OUTCOME IN AFFECTIVE DISORDER: A MACHINE LEARNING APPROACH.

Author: Silvera Sarmiento, Jairo.
Supervisor: Laguna Prada, Ana.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

The project consists of analyzing a database with results of patients with psyquiatric disorders which present a series of symptoms, so that by applying Machine Learning techniques it will predict whether a patient should start treatment or not, this database is structured and contains data based on patients. Machine learning methods for prediction and pattern detection are increasingly prevalent in psychological research. ML techniques are very useful and have a high impact in psychology, especially in understanding hidden patterns in people's behavior. Although psychology as a science that builds theories is slow to adopt ML as a tool to analyze experimental results. Some results of experiments are treated and analyzed with inferential statistical techniques, *(Orrù, Monaro, Conversano, Gemignani, & Sartori, 2020)* but with the advances obtained lately studies have been done with great results.

Keywords: Psychology, treatments, Machine Learning, Python, data science.

1. Introduction

Affective disorder or mood disorder are a set of psychiatric disorders, the main types of affective disorders are depression and bipolar disorder. Symptoms vary by individual and can range from mild to severe There are also other types of mood disorder such as: Major depressive disorder, Seasonal affective disorder (SAD), Cyclothymic disorder, Persistent depressive disorder (dysthymia) *(Mayoclinic, 2021)*. Among the common symptoms for each of the different pathologies are prolonged sadness, lack of interest in normal activities, unusual and chronic mood changes.

Affective disorder o mood disorder is a set of psychiatric disorders, The main types of affective disorders are depression and bipolar disorder. Symptoms vary by individual and can range from mild to severe. Among the common symptoms for each of the

different pathologies are prolonged sadness, lack of interest in normal activities, unusual and chronic mood changes.

There are many causes of this condition, the most common being neurotransmitters or brain chemicals the main causes when there is an imbalance of these, the brain does not function correctly. some events or personal tragedies, alcohol or drug use may also cause mood disorder *(Healthline, 2020).*

Recently, due to the epidemic of the COVID-19 In the first year de la pandemic, global prevalence of anxiety and depression increased by a massive 25% one major explanation for the increase is the unprecedented stress caused by the social isolation resulting from the pandemic, among several causes are loneliness, fear of contracting the disease or the loss of loved ones *(who, 2022)*.

2. Project definition

This is a Machine Learning project, where functions and utilities of this technique were applied. This project is based in conjunction with Machine Learning techniques and data where data and its manipulation are very important according to the Oxford definition data are facts or information, especially when they are examined and used to find out things or to make decisions. Therefore, data is measured, collected, communicated and analyzed, visualized by means of graphs, images or other analysis tools.

Data have different sources, they can be generated by humans, machines, or a combination of both. Here we deal with data generated by humans, for a better processing and analysis of Machine Learning projects, Data Science technologies were applied in this project.

3. Description of the system.

The source is a csv file with rows and columns. The columns are diverse data related to clinical data, test results, psychological data, the target variable is `number_treatments` according to this the prediction will consist of getting patients to treatment. The applied models are Logistic Regression which is a supervised classification algorithm. In a classification problem, the target variable *y*, can take only discrete values for a given set of features *X* the output values are YES and NO. The other model is Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and techniques as *bagging*.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Studiennum	date_t1	date_t2	rehab_days	rehab_week	rehab_week	birthdate	age	sex	diagnoses	group	group_affect	group_unipc	height_cm	height_m	weight
2	1	4/13/2015	5/21/2015	38	5	2	01.02.1959	56,28	2	F32, I44 E66	1	1	1	183,9	1,84	102
3	2	4/13/2015	5/21/2015	38	5	2	06.03.1984	30,86	1	F13, F43	3	0	0	166,8	1,67	56,45
4	3	4/13/2015	5/19/2015	36	5	2	6/14/1965	49,83	2	F33, R73	1	1	1	178,2	1,78	108,4
5	4	4/13/2015	5/21/2015	38	5	2	8/31/1960	54,62	2	F32,	1	1	1	183,6	1,84	89,95
6	5	4/13/2015	5/21/2015	38	5	2	2/18/1943	72,15	1	F32, I10 R55	1	1	1	155,3	1,55	69,05
7	6	4/13/2015	5/21/2015	38	5	2	06.06.1963	51,85	2	F43, Z73 E66	3	0	0	164,2	1,64	74,25
8	7	4/13/2015	5/21/2015	38	5	2	12.11.1963	51,34	1	F33, E66 I10	1	1	1	167,5	1,67	85,15
9	8	4/13/2015	5/21/2015	38	5	2	4/13/1967	48	1	F43, E11 E78	3	0	0	172	1,72	86,35
10	9	4/13/2015	5/21/2015	38	5	2	11/20/1957	57,4	1	F43, F13 K59	3	0	0	167,4	1,67	52,3
11	10	4/13/2015	5/21/2015	38	5	2	10/22/1957	57,48	1	F33, M81	1	1	1	167,3	1,67	64,15
12	11	4/13/2015	5/21/2015	38	5	2	7/18/1958	56,74	2	F33, Z73	1	1	1	158,6	1,59	56,9
13	12	4/13/2015	5/21/2015	38	5	2	01.07.1955	60,27	1	F33, M35 K2	1	1	1	172,2	1,72	63,55
14	13	4/14/2015	5/22/2015	38	5	2	12.08.1951	63,35	2	F31, E66 E78	1	1	1	181,4	1,81	95,6
15	14	4/14/2015	5/22/2015	38	5	2	6/30/1961	53,79	2	F43, E66 I10	3	0	0	183,3	1,83	104,15
16	15	4/14/2015	5/22/2015	38	5	2	04.05.1959	56,02	2	F32, I10 E66	1	1	1	172,7	1,73	98,45
17	16	4/14/2015	5/22/2015	38	5	2	12.12.1947	67,34	2	F31, E66	2	1	0	180,4	1,8	100,35
18	17	4/14/2015	5/22/2015	38	5	2	5/19/1953	61,9	2	F43, I10	3	0	0	170,9	1,71	82,7
19	18	4/14/2015	5/22/2015	38	5	2	3/31/1956	59,04	1	F32, M51 G2	1	1	1	162	1,62	52,2
20	19	4/14/2015	5/22/2015	38	5	2	07.07.1963	51,77	1	F32,	1	1	1	161,7	1,62	53,2
21	20	4/14/2015	5/22/2015	38	5	2	4/28/1969	45,96	1	F32, Z73 E78	1	1	1	166,8	1,67	62,85
22	21	4/14/2015	5/22/2015	38	5	2	05.04.1957	57,94	1	F32, I83	1	1	1	172,9	1,73	55,7

Imagen 4 – Csv file with clinic and psychological data

4. Results

Initially the target variable from the dataset had class imbalance outliers and missing values that were giving inaccurate results. The dataset also has lots of variables which are useless for the model that's why the feature selection had to be made.

After several attempts developed to obtain the best solution for the target the best solution for the model was implementing a Logistic Regression model with scaling, without correlated variables, no missing values and a Random Forest model as well did a great perform.

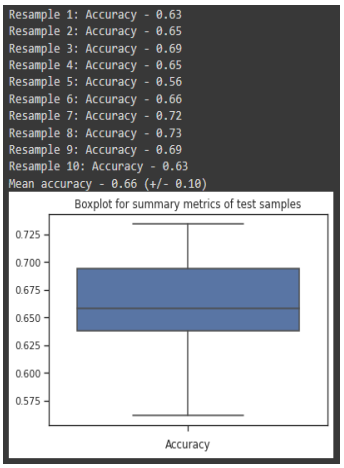


Imagen 5 – Logistic Regression model's accuracy

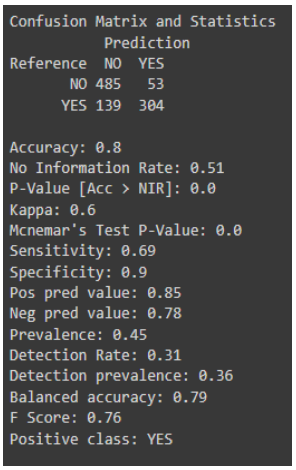


Imagen 6 – Random Forest model's Confusion Matrix and results

5. Conclusions

Several Machine Learning models were implemented and analyzed with different feature extraction methods to find the best possible solution. Machine Learning projects require a staged workflow which was done here, each stage can have few or several subtasks all depending on the target and work sources. The interpretation of the prediction accuracy metrics (e.g., as high, or low accuracy) depends on the context and requires experience in the field since there are many variables and benchmarks that psychologists use and that together allow to establish a diagnosis of the patient.

These models give an important vision to the clinic and several other importants variables so that future psychological studies can base their studies and give more emphasis to the variables mentioned here.

6. Bibliography

Healthline. (2020, April 1). *healthline.com*. Retrieved from https://www.healthline.com/health/affective-disorders

Mayoclinic. (2021, October 29). *mayoclinic.org*. Retrieved from https://www.mayoclinic.org/diseases-conditions/mood-disorders/symptoms-causes/syc-20365057

Orrù, G., Monaro, M., Conversano, C., Gemignani, A., & Sartori, G. (2020, January 10). Retrieved from https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02970/full

who. (2022, March 2). *World Health Organization*. Retrieved from who.int: https://www.who.int/news/item/02-03-2022-covid-19-pandemic-triggers-25-increase-in-prevalence-of-anxiety-and-depression-worldwide

Índice de la memoria

Índice de imágenes

Capítulo 1. INTRODUCCIÓN

Los tratamientos psicológicos son intervenciones profesionales que se basan en muchas variables y aspectos de cada paciente, la depresión y los trastornos afectivos son trastornos graves que generan un alto impacto en las personas y en alto volumen a la sociedad. La psicología trata a cada paciente con técnicas y estrategias especializadas, pero a veces no son efectivas para la variedad y cantidad de pacientes. Es por ello que la correcta identificación de los síntomas junto con las pruebas y test realizados a los pacientes deben ser efectivos en el tratamiento ya sea básico o si requiere de tratamiento clínico.

1.1 MOTIVACIÓN DEL PROYECTO

La motivación del proyecto es que a través del uso combinado de la ciencia de los datos y el Machine Learning podamos determinar la atención que requiere un paciente, el correcto uso e interpretación del Machine Learning en entornos médicos se convierte en una herramienta más para procesar y ofrecer resultados junto con el criterio del profesional de la psicología.

En el marco de este proyecto me parece interesante e importante el uso de Machine Learning que proporciona apoyo o ayuda en la toma de decisiones y tratamiento a los pacientes, a través del uso adecuado de estas herramientas proporcionando además agilidad en algunos procesos y un diagnóstico eficaz en conjunto con la experiencia del profesional.

Mediante estas técnicas, la actividad y el trabajo de los profesionales puede aumentar debido a la precisión del sistema, facilitando así el desarrollo y la innovación de nuevos productos y servicios médicos.

1.2 DESCRIPCIÓN DE LAS TECNOLOGÍAS

Como se describió anteriormente este es un proyecto de Machine Learning, donde se aplicaron las siguientes funciones y librerías. Este es un proyecto de Ciencia de Datos en conjunto con técnicas de Aprendizaje Automático donde los datos y su manipulación son muy importantes de acuerdo con la definición de Oxford los datos son hechos o información, especialmente cuando se examinan y se utilizan para averiguar cosas o para tomar decisiones. Por lo tanto, los datos se miden, se recogen, se comunican y se analizan, se visualizan mediante gráficos, imágenes u otras herramientas de análisis.

Los datos tienen diferentes fuentes, pueden ser generados por humanos, máquinas o una combinación de ambos. Aquí se trata de datos generados por humanos, para un mejor procesamiento y análisis de proyectos de Machine Learning, en este proyecto se utilizan las siguientes tecnologías.

- Data acquisition: database, archivo tabulado csv con información del paciente, entre las variables de interés hay varios grupos: Cognitive raw scores, Cognitive domain z scores, Functional outcome scales, Somatic health variables. Estas variables tienen la notación _T1 admission to therapy y _T2 dismissal (4-6 weeks after T1)
- Se utilizaron tres modelos para el selection feature, Logistic Regression, mrmr y Random Forest.
- Todo el proyecto se realizó en Python 3 junto con los cuadernos de Google Colab por su rapidez y facilidad de trabajo en línea.
- Las librerías utilizadas corresponden al paquete Scikit Learn que ofrece múltiples funciones para Machine Learning.
- Para el análisis exploratorio de datos, gráficos y otras funciones se utilizó paquetes como Pandas, Numpy, Seaborn, Matplotlib.

1.3 ETAPAS DE MACHINE LEARNING

Para la ejecución de este proyecto se siguieron una serie de pasos hasta llegar al modelo final y concluir resultados, el propósito es crear un pipeline y ensamblar varios pasos que puedan ser validados con cross validation mientras se establecen diferentes parámetros. Dentro de nuestro pipeline realizamos imputación, escalado, selección de características y optimización de hiperparámetros, estas etapas hacen parte de la metodología general usada en este proyecto la cual se explica más adelante.

- Collection of Data: para este proyecto fue necesario contar con la información tabulada de los pacientes con todos los resultados clínicos y de los tests realizados en cada uno de ellos, como se indicó previamente es un archivo csv
- Data Wrangling: se divide en varias etapas donde se analiza el archivo suministrado, ver las features y las relaciones entre ellas, identificar outliers y missing values, realizar las conversiones adecuadas de tipos de datos, escalado de features el cual se hizo antes de ejecutar el modelo

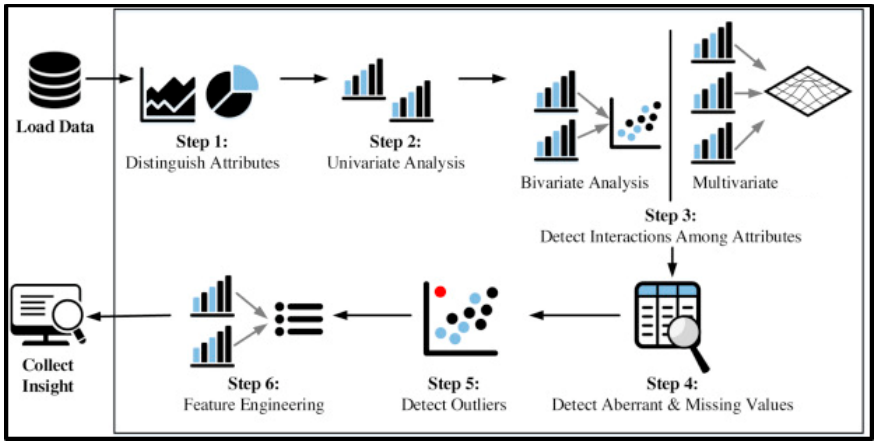


Imagen 7 - Data Wrangling Fuente: www.analyticsvidhya.com

- Model Building: el modelo se divide en training y test, en el conjunto de training se entrena el algoritmo seleccionado, el conjunto de test se utiliza para ver lo bien que la máquina puede predecir nuevas respuestas basándose en su entrenamiento.

Para lo cual se hace el test con los datos que no hacen parte del conjunto de training.

- Model Evaluation: las métricas y accuracy para los modelos de clasificación utilizadas son: Confusion Matrix, Accuracy Score, AUC, F-Score
- Model Deployment: según la finalidad del proyecto se hace despliegue del modelo en un ambiente de producción para obtener los recursos esperados en base a los resultados del modelo.

Imagen 8 - Pipeline Machine Learning. Fuente: www.analyticsvidhya.com

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Las siguientes son las tecnologías usadas en el proyecto:

Python: es un gran lenguaje de programación orientado a objetos, interpretado e interactivo, que permite trabajar con rapidez e integrar sistemas de forma más eficaz. A menudo se le compara con Lisp, Tcl, Perl, Ruby, C#, Visual Basic, Visual Fox Pro, Scheme o Java. Python combina una notable potencia con una sintaxis muy clara. Tiene módulos, clases, excepciones, tipos de datos dinámicos de muy alto nivel y tipado dinámico. Hay interfaces para muchas llamadas al sistema y bibliotecas, así como para varios sistemas de ventanas. Los nuevos módulos incorporados se escriben fácilmente en C o C++ (u otros lenguajes, dependiendo de la implementación elegida). Python también puede utilizarse como lenguaje de extensión para aplicaciones escritas en otros lenguajes que necesiten interfaces de scripting o automatización fáciles de usar. *(The Python Wiki, 2018)*

Google Colab: se viene trabajando con notebooks del proyecto Jupyter los cuales ofrecen un entorno ideal para proyectos de Data Science y Machine Learning. Google Colab es un producto de Google Research. Permite a cualquier usuario escribir y ejecutar código arbitrario de Python en el navegador. Es especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación. Desde un punto de vista más técnico, Colab es un servicio de cuaderno alojado de Jupyter que no requiere actualización e instalación física y que ofrece acceso sin coste adicional a recursos informáticos, como GPUs. *(Google Inc, n.d.)*

Scikit-learn: es una librería de software libre para el uso de Machine Learning es sencillo y dispone de herramientas eficaces para el análisis predictivo de datos, Accesible a todos y reutilizable en varios contextos, de código abierto y utilizable comercialmente. *(scikit-learn, 2022)*

Pandas: es una herramienta de análisis y manipulación de datos de código abierto, rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python. *(Pandas.org, 2022)*

Numpy: NumPy es el paquete fundamental para la computación científica en Python. Es una librería de Python que proporciona un objeto array multidimensional, varios objetos derivados (como arrays enmascarados y matrices), y un surtido de rutinas para realizar operaciones rápidas con arrays, incluyendo operaciones matemáticas, lógicas, manipulación de formas, ordenación, selección, E/S, transformadas discretas de Fourier, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más. *(numpy.org, 2022)*

Seaborn: es una biblioteca de visualización de datos de Python basada en matplotlib. Proporciona una interfaz de alto nivel para dibujar gráficos estadísticos atractivos e informativos. *(seaborn.pydata.org, 2022).*

Capítulo 3. ESTADO DE LA CUESTIÓN

Los proyectos de estudios psicológicos con el uso de Machine Learning son especializados no son tan comunes, previamente se han realizado algunos estudios en conjunto con otras universidades, hospitales y entidades de investigación las cuales han avanzado en algunos temas específicos, estudios interesantes pueden encontrarse en https://www.ncbi.nlm.nih.gov muchos son estudios clínicos que cuentan con el apoyo y trabajo mutuo entre profesionales de la salud y científicos de datos.

En el ámbito psicológico no son tantos los estudios y proyectos realizados o los que hay son más especializados en otros temas y afecciones más específicos. Esta ciencia junto con la Inteligencia Artificial y el uso del Deep Learning están teniendo un alto impacto en las soluciones y tratamiento a pacientes

En los portales https://www.sciencedirect.com y https://www.ncbi.nlm.nih.gov se pueden encontrar trabajos similares con algunos resultados y metodologías similares.

Por tal razón me parece interesante e innovador este tipo de trabajos ya que a futuro la ciencia de datos junto con la medicina tendrá que trabajar de la mano y complementarse para agilizar los diagnósticos, facilitar la toma de decisiones y contribuir a la prevención y tratamiento de enfermedades.

Se espera que la inteligencia artificial (I.A.) influya significativamente en la práctica de la medicina y la prestación de asistencia sanitaria en un futuro próximo Hay una lista creciente de publicaciones sobre el tema en forma de artículos académicos, informes sobre política sanitaria, declaraciones de sociedades profesionales y cobertura de los medios de comunicación populares. *(Meskó, B., Görög, M, 202)*

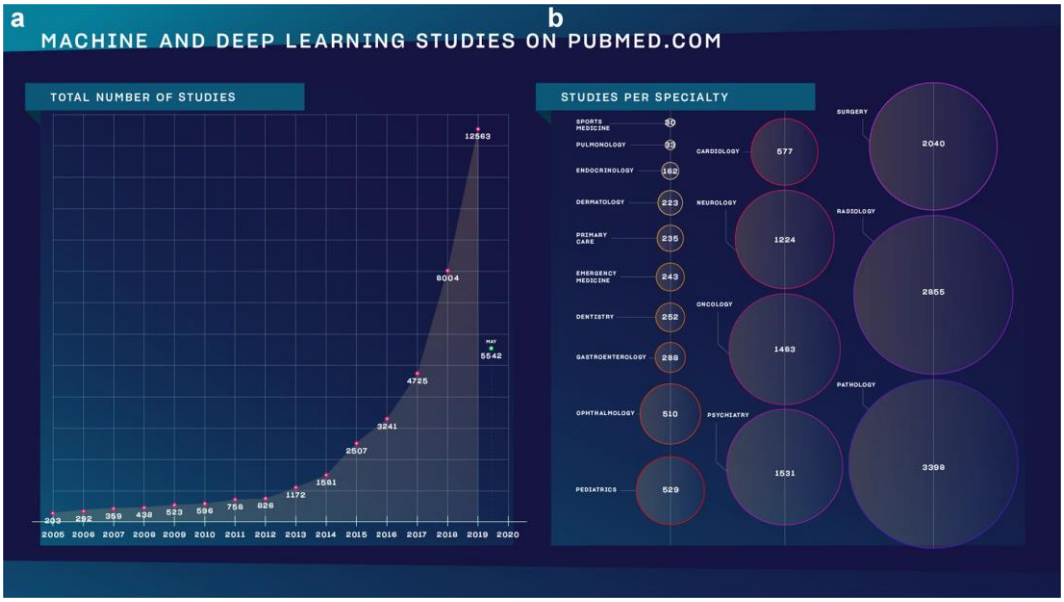


Imagen 9 - Número de estudios de I.A. médica 2010 - 2020 fuente: www.nature.com/

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

El proyecto se va a llevar a cabo debido al creciente uso de las tecnologías junto con la medicina y psicología, la idea es ayudar al profesional médico en el tratamiento y diagnóstico para poner en servicio una mejor ayuda para sus pacientes y procedimientos.

Estos proyectos tipo e-health están teniendo un impacto actualmente, se han hecho estudios al respecto donde incluso se puede dar otra ciencia como la Ingeniería del Comportamiento y es que en estos proyectos un vez programado el algoritmo y entrenado con los datos adecuados, mediante el uso del Machine Learning se puede obtener un juicio objetivo que el profesional en su experiencia debe analizar y otorgar un resultado y tratamiento esto permite una interpretación del paciente con algún trastorno psicológico donde la fuente de datos debe ser de calidad, real y libre de sesgos.

En otros campos como la Inteligencia Artificial se han dado hitos y avances históricos donde se ha podido acertar hasta un 87% de los casos en el diagnóstico de tumores cerebrales (Bejerano, 2018). Por lo tanto, el Machine Learning es clave en esta nueva etapa y colaboración con la medicina

Este proyecto se plantea orientado a prestar ayuda en el tratamiento a trastornos psicológicos con una ventaja clara y es que no tiene mayor costo, puesto todo se está haciendo con software libre, es una herramienta más para los profesionales de la psicología y presta un apoyo para el tratamiento a pacientes.

4.2 OBJETIVOS

El objetivo principal de este trabajo es predecir si un paciente es ingresado a tratamiento psicológico dado una serie de datos y características que vienen en el archivo fuente.

4.3 METODOLOGÍA

La metodología aplicada en este proyecto es el modelo CRISP-DM Cross Industry Standard Process for data Mining este modelo fue definido como un proyecto en colaboración con las empresas ISL, NCR y Daimler Chrysler. (ibm, 2021)

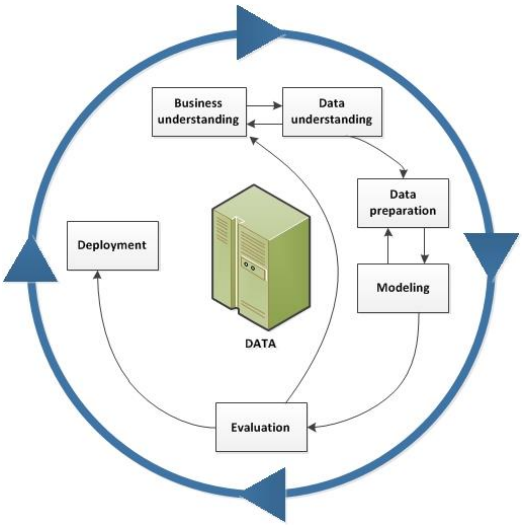


Imagen 10 - CRISP-DM methodology. Fuente: www.ibm.com

Las siguientes son las etapas de la metodología:

1. Business Understanding: entender los objetivos del proyecto y los requisitos lo cual indica que en base a una base de datos finalmente el target es predecir si un paciente ingresa a tratamiento.
2. Data Understanding: recolección inicial de datos dado el Business Understanding aquí se determina de donde y con que métodos los datos son extraídos con la intención de familiarizarse con ellos, en este punto ya sabemos que los datos ya fueron recolectados previamente son datos de pacientes con algunos trastornos psicológicos y todo fue ingresado a una base de datos posteriormente exportados a un archivo csv con el cual se está trabajando.
3. Data Preparation: una vez recogidos los datos, hay que transformarlos en un subconjunto utilizable, a menos que se determine que se necesitan más datos. Una vez elegido el conjunto de datos hay que comprobar que no haya casos dudosos, que falten o que sean ambiguos.
4. Modeling: cuando los datos ya están para su uso, deben aplicarse los modelos que sean apropiados, consiste en la selección y aplicación de diferentes modelos de machine learning.
5. Evaluation: el modelo seleccionado debe ser evaluado con las distintas técnicas utilizadas en ML. Los resultados de esta prueba se utilizan para determinar la eficacia del modelo y comprobar si los resultados obtenidos son acordes a los objetivos del proyecto.
6. Deployment: desplegar el modelo obtenido a producción.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

5.1 BUSINESS UNDERSTANDING

El proyecto nace de la necesidad de predecir si un paciente es ingresado a tratamiento con base a diferentes datos propios de cada paciente.

5.2 DATA UNDERSTANDING

Los datos de pacientes son diversos y variados, constan de datos personales y físicos, los datos vienen clasificados así donde _T1 indica admisión a terapia y _T2 salida (4-6 semanas después de la T1). El hospital de Gratz en colaboración con la universidad de Gratz ha venido recopilando información desde 2015 a 2017 de distintos pacientes con patologías relacionadas a trastornos afectivos, la edad de los pacientes oscila entre los 19 y 80 años, es un dataset clínico donde la información de las variables está catalogada de esta forma

- Cognitive raw scores at t1: son las puntuaciones brutas cognitivas de los pacientes al ingresar, la cognición aplica a todas las formas de conocimiento y conciencia, como percibir, concebir, recordar, razonar, juzgar, imaginar y resolución de problemas.
- Cognitive domain z scores at t1: estas variables están relacionadas con la atención y memoria ya que se incluye el California Verbal Learning Test o el Trail Making Test
- Functional outcome scales at t1 and t2: son las escalas de resultados funcionales, básicamente son los resultados de los cambios reales producidos en los pacientes al ingresar _T1 y al salir _T2.
- Somatic health variables: las variables somáticas son aquellas que surgen del cuerpo o están relacionadas al cuerpo físico por ejemplo BMI, presión sanguínea.

El dataset es un archivo csv donde viene tabulada toda la información.

- Cognitive raw scores at t1: en esta sección se encuentran variables como

Variable	Definición
D2_KL_RW_t1	concentration (d2R) at t1
CVLT_LS_DG1_5_RW1 CVLT_VFWI_RW1 CVLT_WAI_RW1 CVLT_VFWII_RW1 CVLT_WAII_RW1	California verbal learning test (sum1-5) t1 CVLT short-delayed free recall t1 CVLT short-delayed cued recall t1 long-delayed free recall t1 long-delayed cued recall t1
FWIT_FWL1 FWIT_FSB1 FWIT_INT1	Stroop color word reading t1 Stroop color naming t1 Stroop interference t1
TMTA1 TMTB1	Trail Making Test part A t1 Trail Making Test part B t1
MWTB	premorbid IQ (raw score from MWTB)

- Cognitive domain z scores at t1:

Definición	Variable
Attention_processing_speed_1	TMT A1, D2_KL_RW_t1, FWIT_FWL1 FWIT_FSB1

Verbal_memory_1	CVLT_LS_DG1_5_RW1 CVLT_VFWI_RW1 CVLT_WAI_RW1 CVLT_VFWII_RW1 CVLT_WAII_RW1
Excecutive_function_1	FWIT_INT1, TMTB1
ZAttention_processing_speed_1	Domain z score of attention
ZVerbal_memory_1	Domain z score of memory
ZExcecutive_function_1	Domain z score of executive function

- Functional outcome scales at t1 and t2

Variable	Definición
GAF_t1 GAF_t2	Global Assessment of Functioning t1 Global Assessment of Functioning t2
SF12_physical_t1 SF12_psychological_t1 SF12_physical_t2 SF12_psychological_t2	SF12 physical score t1 SF-12 mental score t1 SF12 physical score t2 SF-12 mental score t2
SCLR_GSI_t1 SCLR_PSDI_t1 SCLR_PST_t1 SCLR_GSI_t2	GSI (Global severity index) t1 (SCLR) PSDI (Positive Symptom Distress Index) t1 (SCLR) Positive Symptom Total t1 (SCLR) GSI (Global severity index) t2 (SCLR)

SCLR_PSDI_t2 SCLR_PST_t2	PSDI (Positive Symptom Distress Index) t2 (SCLR) Positive Symptom Total t2 (SCLR)
BDI_sum_t1R BDI_sum_t2R	Beck Depression Inventory t1 (self- rating of depressive symptoms) // Beck Depression Inventory t2
HAMD_sum_t1 HAMD_sum_t2	Hamilton Depression Scale (external rating of depressive symptoms)
WHOQOL_psych_t1 WHOQOL_phys_t1 WHOQOL_soc_t1 WHOQOL_environ_t1 WHOQOL_global_t1 ********** WHOQOL_psych_t2 WHOQOL_phys_t2 WHOQOL_soc_t2 WHOQOL_environ_t2 WHOQOL_global_t2	WHOQOL (Quality of Life) psychological health t1 WHOQOL (Quality of Life) physical health t1 WHOQOL (Quality of Life) social relationship t1 WHOQOL (Quality of Life) environment t1 WHOQOL (Quality of Life) global score t1 ********** WHOQOL (Quality of Life) psychological health t2 WHOQOL (Quality of Life) physical health t2 WHOQOL (Quality of Life) social relationship t2 WHOQOL (Quality of Life) environment t2 WHOQOL (Quality of Life) global score t2

- Somatic health variables

Variable	Definición
BMI_T1 BMI_T2	Body Mass Index
WHR_t1 WHR_t2	Waist to hip ratio
WHTR_t1 WHTR_t2	Waist to height ratio
RR_sys RR_dia	Blood pressure (t1)
HDL365_t1 LDL366_t1 TG367_t1	High density lipoprotein low density lipoprotein Triglycerides (lipid metabolism)
CRh393_t1	hsCRP (marker of inflammation)
IL6457_t1	Il6 (marker of inflammation)
Tryptophan_t1 Kynurenine_t1 Kyn_Trp_t1	Tryptophan Kynurenine Kyn-Tryp-Ratio (marker of chronic low grade inflammation)

Todas estas variables son de las más importantes del dataset pero hay más, finalmente se trabajó con todas las features para obtener mejores resultados.

En este punto del Data Understanding es importante conocer el significado de algunas de estas variables puesto son temas y conceptos especializados los cuales no son conocidos por el público en general.

1. Concentration: el acto de reunir o concentrar, como, por ejemplo, reunir los procesos de pensamiento de una persona en torno a un problema o tema central. *(American Psychological Association, 2022)*
2. California Verbal Learning Test (CVLT): es una prueba de aprendizaje de listas de palabras que consta de 16 ítems pertenecientes a una de las cuatro categorías. Actualmente en su segunda edición (CVLT-II), la prueba evalúa el recuerdo libre inmediato tras cada uno de los cinco ensayos de aprendizaje, así como un ensayo de interferencia. *(American Psychological Association, 2022)*

3. Stroop color (FWIT): una prueba de tres partes en la que (a) se leen los nombres de los colores lo más rápido posible; (b) se nombran rápidamente los colores de las barras u otras formas y, lo que es más importante, (c) se nombran rápidamente los tonos de los colores cuando se utilizan para imprimir los nombres de otros colores (como la palabra verde impresa en el color rojo). El grado de interferencia de los participantes con las palabras impresas es una medida de su flexibilidad cognitiva, atención selectiva e inhibición (o desinhibición) de la respuesta. *(American Psychological Association, 2022)*

4. Trail Making Test (TMTA): una tarea de conectar los puntos que forma parte de la Batería Neuropsicológica Halstead-Reitan. Las pistas A requieren la conexión en secuencia de 25 puntos etiquetados con números. La prueba B requiere la conexión en secuencia de 25 puntos etiquetados con números y letras alternados (1-A-2-B-3-C). La prueba, una de las más utilizadas para el deterioro cognitivo, se supone que mide varias funciones, en particular la flexibilidad cognitiva, la atención alternante,

la secuenciación, la búsqueda visual y la velocidad motora. *(American Psychological Association, 2022)*

5. MWTB (premorbid IQ): caracterizar el estado de un individuo antes de la aparición de alguna enfermedad o trastorno. *(American Psychological Association, 2022)*

6. Global Assessment of Functioning Scale (GAF): una escala utilizada para la planificación del tratamiento y la evaluación de los resultados en el Eje V del sistema de evaluación multiaxial del DSM-IV-TR. Las puntuaciones (1-100) reflejan el juicio del clínico sobre el nivel general de funcionamiento psicológico, social y ocupacional del paciente en el momento de la evaluación. *(American Psychological Association, 2022)*

7. Social Functioning (SF12): es una medida de la calidad de vida relacionada con la salud en diversos grupos de población. (Huo,Tianyao; Guo,Yi ; Shenkman, Elizabeth; Muller,Keith, 2018)

8. SCLR_GSI: SCLR es un cuestionario ampliamente utilizado para determinar una serie de síntomas psicológicos, el Global Severity Index (GSI) está diseñado para ayudar a cuantificar la gravedad de la enfermedad de un paciente y proporciona una única puntuación compuesta para medir el resultado de un programa de tratamiento basado en la reducción de la gravedad de los síntomas. GSI - Global Severity Index, Ayuda a medir el nivel de malestar psicológico general. (Derogatis, Leonard R, 1993)

9. SCLR_PSDI: SCLR es un cuestionario ampliamente utilizado para determinar una serie de síntomas psicológicos, el PSDI - Positive Symptom Distress Index, Ayuda a medir la intensidad de los síntomas. (Derogatis, Leonard R, 1993)

10. SCLR_PST: es un cuestionario ampliamente utilizado para determinar una serie de síntomas psicológicos, el PST - Positive Symptom Total, Informa del número de síntomas autodeclarados. (Derogatis, Leonard R, 1993)

11. Beck Depression Inventory (BDI_sum): es un cuestionario de autoinforme diseñado para evaluar la gravedad de los síntomas depresivos en adolescentes y adultos. Utilizado ampliamente tanto en el ámbito clínico como en el de la investigación, consta de 21 grupos de ítems, cada uno de los cuales incluye cuatro afirmaciones de gravedad creciente. *(American Psychological Association, 2020)*

12. Hamilton Depression Scale (HAMD_sum): es un cuestionario de múltiples ítems que se utiliza para proporcionar una indicación de la depresión, y como guía para evaluar la recuperación. *(Wikipedia.org, 2022)*

13. WHOQOL (Quality of Life): es una evaluación de la calidad de vida desarrollada por el grupo WHOQOL con quince centros internacionales de campo, simultáneamente, en un intento de desarrollar una evaluación de la calidad de vida que sea aplicable transculturalmente. *(World Health Organization, 2012)*

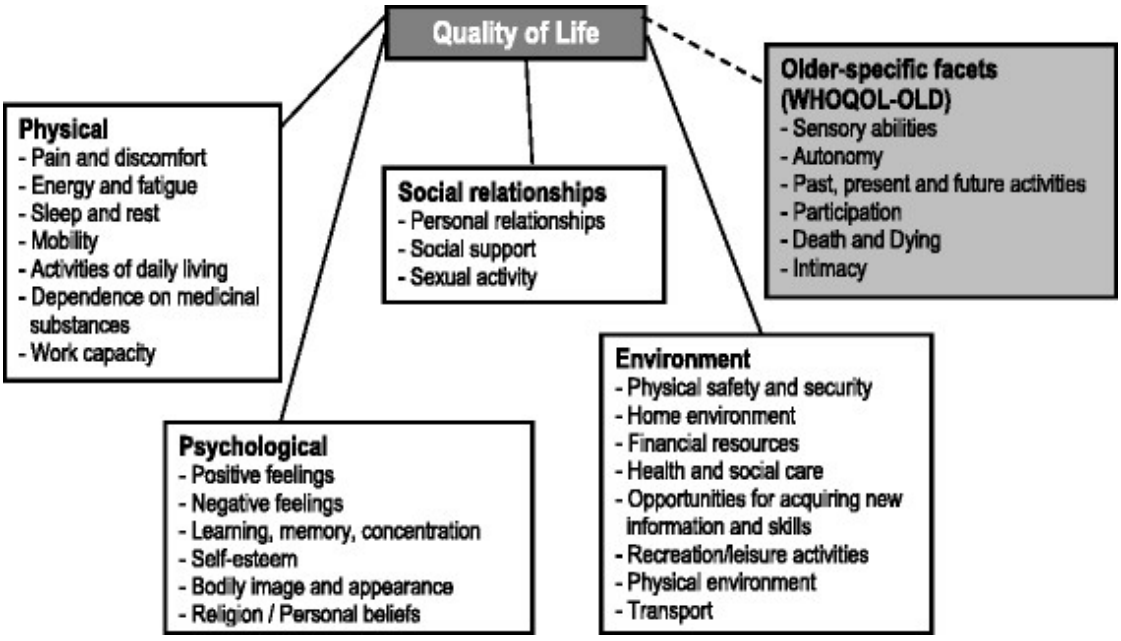


Imagen 11 - Quality of life dimensions. Fuente: www.researchgate.net

14. Body Mass Index (BMI): una medida de adiposidad u obesidad ampliamente utilizada, basada en la siguiente fórmula: peso (kg) dividido por la altura al cuadrado (m2). *(American Psychological Association, 2022)*
15. Waist to hip ratio (WHR): es la relación adimensional entre la circunferencia de la cintura y la de las caderas. Se calcula como la medida de la cintura dividida por medición de la cadera, hip measurement (W/H). *(Wikipedia.org, 2022)*

16. Waist to height (WHTR): es el perímetro de la cintura dividido por la altura, ambos medidos en las mismas unidades, ya sean métricas o inglesas. WHtR es una medida de la distribución de la grasa. Waist-to-Height Ratio = Waist Circumference / Height. *(Denea, Denise, 2020)*

17. Blood pressure (RR): La presión ejercida por la sangre contra las paredes de los vasos sanguíneos, especialmente de las arterias. Varía con la fuerza de los latidos del corazón, la elasticidad de las paredes arteriales y la resistencia de las arteriolas, así como con la salud, la edad y el estado de actividad de la persona. Systolic blood pressure (sys) es mayor que el medido cuando el corazón está relajado (diastolic blood pressure, dia). Las lecturas se registran (en milímetros de mercurio) como sistólica/diastólica; la presión arterial normal en reposo para un adulto es inferior a 120/80. *(American Psychological Association, 2022)*
18. High density lipoprotein (HDL365): también llamado colesterol "bueno", absorbe el colesterol y lo lleva de vuelta al hígado. A continuación, el hígado lo elimina del organismo. Los niveles elevados de colesterol HDL pueden reducir el riesgo de padecer enfermedades cardíacas y accidentes cerebrovasculares. (Centers for Disease Control and Prevention, 2020)
19. low density lipoprotein (LDL366): constituye la mayor parte del colesterol de su cuerpo. Los niveles elevados de colesterol LDL aumentan el riesgo de sufrir enfermedades cardíacas y accidentes cerebrovasculares. (Centers for Disease Control and Prevention, 2020)

20. Triglycerides (TG367): Los triglicéridos son un tipo de grasa. Son el tipo de grasa más común en el cuerpo. Proceden de los alimentos, especialmente de la mantequilla, los aceites y otras grasas. Los triglicéridos también provienen de las calorías adicionales.
21. hsCRP - marker of inflammation (CRh393): high-sensitivity C-reactive protein (hsCRP) es un marcador de inflamación que predice el infarto de miocardio, el ictus, la enfermedad arterial periférica y la muerte súbita cardíaca en individuos sanos sin antecedentes de enfermedad cardiovascular, así como los eventos recurrentes y la muerte en pacientes con síndromes coronarios agudos o estables. (Bassuk, Shari S; Rifai, Nader; Ridker, Paul M, 2004)
22. Il6 - marker of inflammation (IL6457): Interleukin 6 (IL-6), se produce rápida y transitoriamente en respuesta a infecciones y lesiones tisulares, contribuye a la defensa del huésped mediante la estimulación de las respuestas de fase aguda, la hematopoyesis y las reacciones inmunitarias. (Tanaka, Toshio ; Narazaki, Masashi; Kishimoto, Tadamitsu, 2014)
23. Tryptophan: L-tryptophan es un aminoácido esencial necesario para la producción de proteínas. (MedlinePlus, 2021)
24. Kynurenine: es un metabolito del aminoácido l-triptófano utilizado en la producción de niacina. *(Wikipedia.org, 2022)*
25. Kyn-Tryp-Ratio (Kyn_Trp): la kinurenina plasmática a triptófano ([Kyn]/[Trp]) se utiliza con frecuencia para expresar o reflejar la actividad de la enzima degradadora de Trp extrahepática, la indoleamina 2,3-dioxigenasa (IDO). Esta proporción se utiliza cada vez más en lugar de la medición de la actividad de la IDO, que suele ser baja o indetectable en las células inmunitarias y de otro tipo en condiciones basales, pero que aumenta enormemente tras la activación inmunitaria. *(Badawy, Abdulla A-B; Guillemin, Gilles , 2019)*

Adicional a las variables mencionadas la variable target es **number_treatments** la cual inicialmente en el dataset viene por número de tratamientos, donde viene cada

número de tratamientos y su frecuencia en el dataset en la siguiente sección se explicará en detalle la transformación de esta variable

```
count = df_bh.groupby(['number_treatments']).size()
count.head(30)
```

```
number_treatments
1018
0          31
1         227
10          8
11          1
12          1
13          1
15          1
2          76
20          1
3          36
4          20
5          13
6           4
7           3
8           4
9           2
9999       74
dtype: int64
```

Imagen 12 - Método size() del dataset df_bh

Este proyecto se trabajó en Google Colaba con los Jupyter Notebooks, el info() inicial es este:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1521 entries, 0 to 1520
Columns: 651 entries, Studiennummer to RZDiff_e
dtypes: float64(3), int64(3), object(645)
memory usage: 7.6+ MB
```

Son en total 1521 filas y 651 columnas la mayoría vienen tipo object no todas las columnas serán utilizadas en el modelo puesto algunas traen datos irrelevantes.

Head() del dataset:

index	Studiennummer	date_t1	date_t2	rehab_days	rehab_weeks	rehab_weeks_4_6	birthdate	age	sex	diagnoses	group	group_affective	group_unipolar
0	1	4/13/2015	5/21/2015	38	5	2	01.02.1959	56,28	2	F32, I44 E66 E79	1.0	1.0	1.0
1	2	4/13/2015	5/21/2015	38	5	2	06.03.1984	30,86	1	F13, F43	3.0	0.0	0.0
2	3	4/13/2015	5/19/2015	36	5	2	6/14/1965	49,83	2	F33, R73	1.0	1.0	1.0
3	4	4/13/2015	5/21/2015	38	5	2	8/31/1960	54,62	2	F32,	1.0	1.0	1.0
4	5	4/13/2015	5/21/2015	38	5	2	2/18/1943	72,15	1	F32, I10 R55 I48 G43 M15	1.0	1.0	1.0

Imagen 13 - head() en el dataset df_bh

5.3 *DATA PREPARATION*

Con los datos cargados del dataset y de acuerdo al pipeline inicialmente se procede con el análisis exploratorio de datos y en general ver la información del dataset para preparar la información necesaria para el modelo de Machine Learning.

El archivo es leído con la librería Pandas: `df_bh = pd.read_csv("/content/drive/MyDrive/TFM/Data/Bad Hell_nina.12.02.22sav.csv", sep=";")` y asignado a la variable df_bh previamente hacer el mount en Google Colab `drive.mount('/content/drive')`

El dataset tiene 1521 filas y 651 columnas la mayoría de estas variables vienen en tipo object por lo cual es necesario convertirlas a int y float para ver datos estadísticos.

```
Number of rows and columns: (1521, 651)

Columns names: Index(['Studiennummer', 'date_t1', 'date_t2',
'rehab_days', 'rehab_weeks','rehab_weeks_4_6', 'birthdate',
'age', 'sex', 'diagnoses',

...
```

```
'Diff_executive','ZDiff_attention','ZDiff_verbal','ZDiff_exec
utive',

'NZDiff_a', 'NZDiff_v', 'NZDiff_e', 'RZDiff_a', 'RZDiff_v',
'RZDiff_e'],

dtype='object', length=651)
```

Una primera muestra con `describe()` muestra lo siguiente:

	Studiennummer	sex	group	group_affective	group_unipolar	BZ394_t1
count	1521.000000	1521.000000	1519.000000	1519.000000	1519.000000	1521.000000
mean	761.000000	487.886917	1469.291639	1468.566162	1468.520737	266.683103
std	439.219194	2151.571405	3539.379484	3539.680421	3539.699272	1284.021359
min	1.000000	1.000000	1.000000	0.000000	0.000000	62.000000
25%	381.000000	1.000000	1.000000	1.000000	0.000000	88.000000
50%	761.000000	2.000000	1.000000	1.000000	1.000000	94.000000
75%	1141.000000	2.000000	3.000000	1.000000	1.000000	103.000000
max	1521.000000	9999.000000	9999.000000	9999.000000	9999.000000	9999.000000

Imagen 14 - Método describe() del dataset df_bh

Solo se muestran algunas variables numéricas que ya venían con ese formato.

En general el dataset contiene valores variados, como atípicos, espacios vacíos, decimales, y enteros, por ejemplo, valores NAN como tal marcados así son muy pocos la mayoría de variables tienen muchos espacios vacíos que propiamente no están marcados como NAN

Ejemplo en la columna target `number_treatments`:

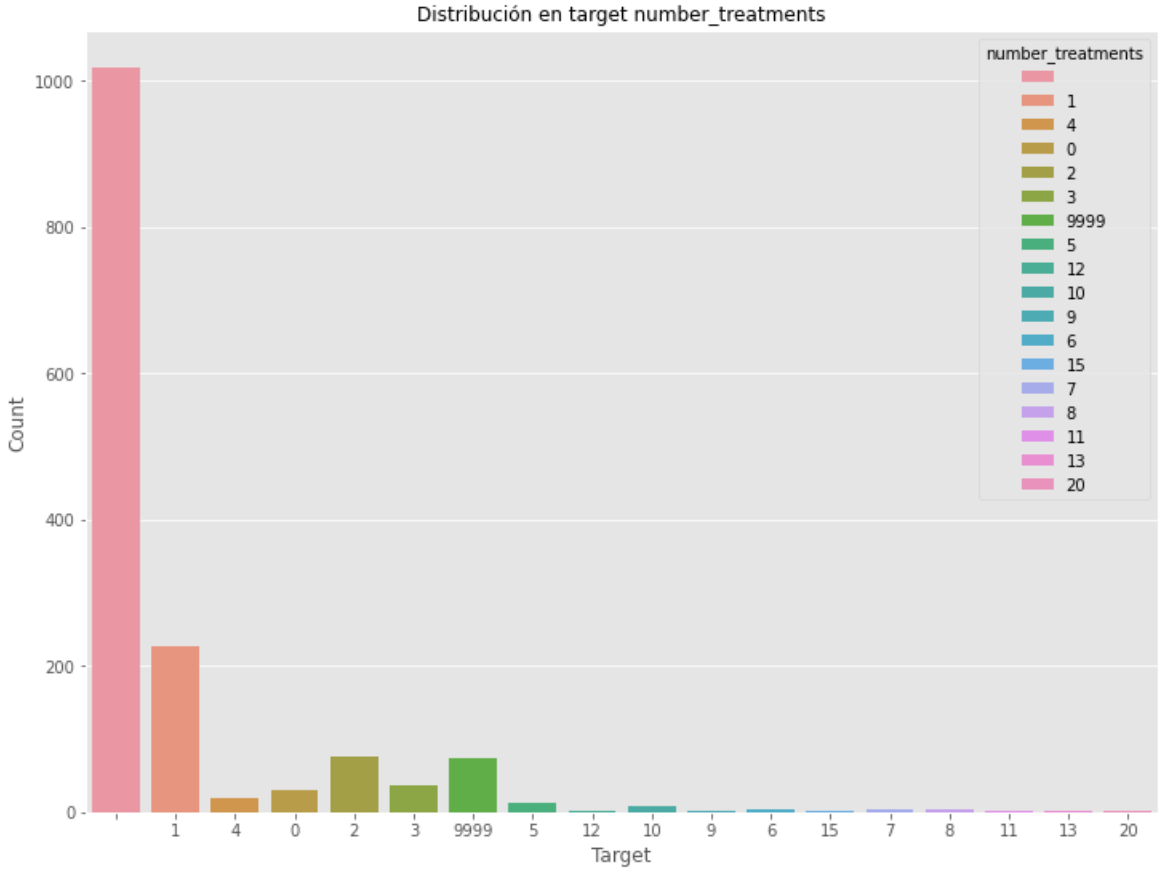


Imagen 15 - Ccountplot target number_treatments del dataset df_bh

Los espacios vacíos son los datos con más frecuencia con más de 1000 muestras, lo cual indica un desbalance.

El total de missing values reales marcados así como vienen del dataset son pocos, estos registro son eliminados con el método `dropna()` ya que son pocos y no afectan el modelo.

```
total = df_bh.isnull().sum().sort_values(ascending = False)

percent = (df_bh.isnull().sum() /
df_bh.isnull().count()).sort_values(ascending = False)

missing_data = pd.concat([total, percent], axis = 1, keys =
['Total', 'Percent'])
```

```
missing_data.head(10)
```

	Total	Percent
group_affective	2	0.001315
group_unipolar	2	0.001315
group	2	0.001315
HDL365_t2	0	0.000000
LDL366_t2	0	0.000000
TG367_t2	0	0.000000
CHR374_t2	0	0.000000
NHD375_t2	0	0.000000
NA376_t2	0	0.000000
K377_t2	0	0.000000

Imagen 16 - Número y porcentaje de valores nulos del dataframe df_bh

Como se muestra en la Imagen 9 el target number_treatments está muy desbalanceado con la frecuencia de espacios vacíos más alta con 66.92% del dataset

```
Percentage of observations per class:
        66.929652
1       14.924392
2        4.996713
9999     4.865220
3        2.366864
0        2.038133
4        1.314924
5        0.854701
```

```
10        0.525970
6         0.262985
8         0.262985
7         0.197239
9         0.131492
12        0.065746
15        0.065746
11        0.065746
13        0.065746
20        0.065746
Name: number_treatments, dtype: float64
```

Debido a esto y para efecto del proyecto donde vamos a predecir si un paciente es ingresado a tratamiento los valores de `number_treatments` son asignados así

```
number_treatments > 0 = 1

number_treatments = 0 = 0
```

Si es igual a 0 se cambia el valor a 0 si es mayor 0 se cambia por 1.

Adicional a esto el dataset contiene dos columnas `date_t1` y `date_t2` donde se indica admisión a terapia y salida de terapia respectivamente.

```
# size date_t1
count = df_bh.groupby(['date_t1']).size()
count.head(10)
```

```
date_t1
              48
01.03.2017     7
01.04.2016     9
01.04.2017     6
01.05.2016     9
01.05.2017     6
01.09.2016     5
01.10.2017     7
01.11.2016    11
01.11.2017     6
dtype: int64
```

Imagen 17 - Método size() de la columna date_t1 en el dataset df_bh

```
# size date_t2
count = df_bh.groupby(['date_t2']).size()
count.head(10)
```

```
date_t2
              1119
02.03.2016       1
02.04.2016       1
02.11.2016       8
06.05.2015       6
06.08.2015       9
06.10.2015      10
08.05.2015       1
08.06.2015       8
08.07.2015       9
dtype: int64
```

Imagen 18 - Método size() de la columna date_t2 en el dataset df_bh

Las dos Imagens muestran las distintas fechas de ingreso y salida de tratamiento de los pacientes.

Dado que la columna `number_treatments` tiene un número considerado de valor atípico (9999), los cambios de 0 y 1 se hacen con base a las columnas `date_t1` y `date_t2`.

Primero asignamos en otro dataframe los valores correspondientes a '9999' de `number_treatments` junto con los no vacíos de `date_t1`

```
df=df_bh.loc[(df_bh['number_treatments']=='9999')&(df_bh['dat
e_t1'] != ' ')]
```

En otro dataframe se hacen las agregaciones para ver la frecuencia del dataframe anterior `df` que ya viene filtrado por number_treatments igual a '9999'

```
datos=df.groupby(['number_treatments','date_t1','date_t2'])['
number_treatments'].agg(['count'])

datos.head(5)
```

			count
number_treatments	date_t1	date_t2	
9999	01.11.2016	02.04.2016	1
	01.11.2017		2
	02.02.2017		1
	06.01.2015		1
	07.07.2015	8/13/2015	1

Imagen - head() en el nuevo dataframe datos

Con el fin de graficar los datos de `number_treaments` y ver la cantidad de valores '9999' se agrupan en otro dataframe

```
df_new2=df.groupby(['number_treatments','date_t1','date_t2'])
["number_treatments"].count().reset_index(name="count")
```

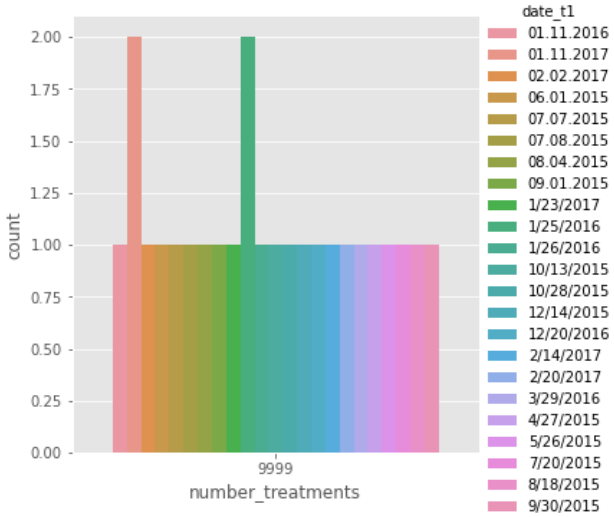


Imagen - Countplot de number_treatments en el nuevo dataframe creado

La anterior Imagen indica que los valores no vacíos de la columna `date_t1` tienen el valor de '9999' lo siguiente ahora es filtrar por '9999' y ver los valores de `date_t1` y `date_t2`.

	number_treatments	date_t1	date_t2	count
598	9999			48
599	9999	01.11.2016	02.04.2016	1
600	9999	01.11.2017		2
601	9999	02.02.2017		1
602	9999	06.01.2015		1
603	9999	07.07.2015	8/13/2015	1
604	9999	07.08.2015	8/14/2015	1
605	9999	08.04.2015	09.10.2015	1
606	9999	09.01.2015	10.09.2015	1
607	9999	1/23/2017		1

Imagen - Los datos filtrados de '9999', de las columnas date_t1 y date_t2

La anterior imagen indica que algunos pacientes tuvieron fecha de entrada y salida y algunos solo fecha de entrada.

En el siguiente gráfico entre `number_treatments` y `date_t1` se observa que number_treatments tiene valores vacíos también, por lo tanto los 74 registros atípicos con '9999' no serán eliminados sino transformados en 1 y 0 al igual que el resto de datos

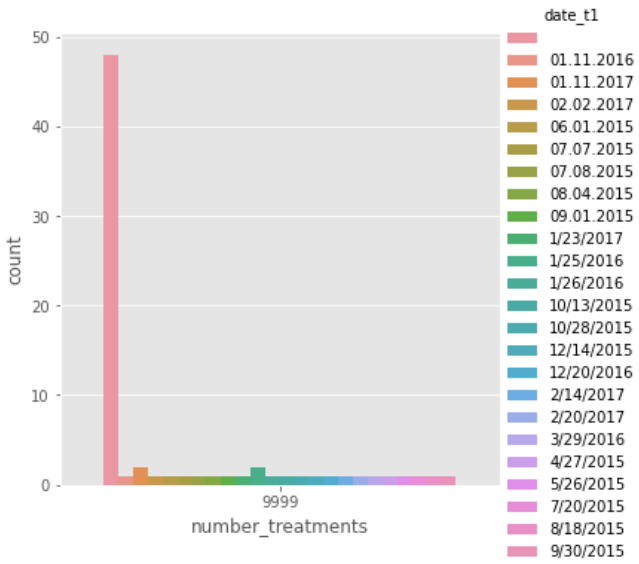


Imagen 19 - filtro de '9999'de la columna date_t1

Los valores iniciales de la columna `number_treaments` serán transformador de acuerdo al valor que contenga a 1 y 0 como se indicó previamente, en la Imagen 17 se muestran los datos iniciales con el método size()

```
# count number_treatments
count = df_bh.groupby(['number_treatments']).size()
count.head(30)
```

```
number_treatments
        1018
0         31
1        227
10         8
11         1
12         1
13         1
15         1
2         76
20         1
3         36
4         20
5         13
6          4
7          3
8          4
9          2
9999      74
dtype: int64
```

Imagen 20 - Frecuencia inicial de datos de number_treaments

- number_treatments cuyo valor es '9999' y date_t1 es vacío se coloca '0'

  Con la siguiente instrucción aseguramos los datos en '0' entre date_t1 y `number_treatments` igual a '9999'

```
df_bh['number_treatments']=np.where((df_bh['number_treatments'] == '9999') & (df_bh['date_t1'] == ' '), '0', df_bh['number_treatments'])
```

```
count = df_bh.groupby(['number_treatments']).size()
count.head(30)
```

```
number_treatments
        1017
0         79
1        226
10         8
11         1
12         1
13         1
15         1
2         76
20         1
3         36
4         20
5         13
6          4
7          3
8          4
9          2
9999      26
dtype: int64
```

Imagen - size() de number_treatments después de cambiar valores

- number_treatments cuyo valor es '9999' y date_t1 y date_t2 son distintos de vacío se coloca '1'

  Con esta instrucción se indica que el valor '9999' con los valores date_t1 y date_t2 distintos de vacío los pacientes tuvieron ingreso a tratamiento, los valores se cambian a '1'

```
df_bh['number_treatments']=np.where((df_bh['number_treatments'] == '9999') & (df_bh['date_t1'] != ' ') & (df_bh['date_t2']!=''), '1', df_bh['number_treatments'])
```

Para comprobar cada resultado se ejecuta el método size()

```
count = df_bh.groupby(['number_treatments']).size()
count.head(30)
```

```
number_treatments
        1017
0         79
1        240
10         8
11         1
12         1
13         1
15         1
2         76
20         1
3         36
4         20
5         13
6          4
7          3
8          4
9          2
9999      12
dtype: int64
```

Imagen - size() de number_treatments después de cambiar valores

- number_treatments cuyo valor es '9999' lo que resta son date_t1 no vacíos con date_t2 vacíos se coloca '0'

  Ya en este punto los valores con '9999' en `number_treatments` tienen a su vez valores vacíos en `date_t2` pero no vacíos en `date_t1` se coloca '0' debido a que no tiene valor en date_t2 y además es un valor atípico el cual no se quiere eliminar.

```
df_bh['number_treatments']=np.where((df_bh['number_treatments'] == '9999') & (df_bh['date_t1'] != ' ') & (df_bh['date_t2']==' '),'0', df_bh['number_treatments'])
```

```
count = df_bh.groupby(['number_treatments']).size()
count.head(20)
```

```
number_treatments
      1017
0       91
1      240
10       8
11       1
12       1
13       1
15       1
2       76
20       1
3       36
4       20
5       13
6        4
7        3
8        4
9        2
dtype: int64
```

Imagen 21 - El valor 9999 ya fue reemplazo dentro del dataframe por 0 y 1

Llegados a este punto los valores de '9999' de la columna `number_treatments` ya fueron cambiados a 0 y 1, resta por cambiar los valores restantes que se muestran en la Imagen 20.

- `number_treatments` cuyo valor es mayor a 0 y `date_t1` o `date_t2` es distinto de vacío se coloca '1'. Con esta instrucción aseguramos que los valores mayores a 0 y con el uso de `OR` en date_t1 y date_t2 distintos de vacíos ya tenemos los 1

```
df_bh['number_treatments']=np.where((df_bh['number_treatments'] > '0') & ((df_bh['date_t1'] != ' ') | (df_bh['date_t2']!='')),'1', df_bh['number_treatments'])
```

```
count = df_bh.groupby(['number_treatments']).size()
count.head(30)
```

```
number_treatments
     1017
0      91
1     411
dtype: int64
```

Imagen 22 - number_treaments size después de cambiar los valores

- `number_treatments` cuyo valor es vacío ' ' y `date_t2` y `date_1` es distinto de vacío se coloca '1'.

  Lo siguiente son los espacios vacíos los cuales deben tratarse y cambiar de acuerdo a los datos de las columnas `date_t1` y `date_t2` y que además son el valor más numeroso de la columna `number_treatments` en este caso aseguramos que tengan fechas ingresadas en las columnas `date,` se coloca 1 para indicar que el paciente ingresó a terapia.

```
df_bh['number_treatments']=np.where((df_bh['number_treatments'] == ' ') & ((df_bh['date_t1'] != ' ') & (df_bh['date_t2'] != ' ')), '1', df_bh['number_treatments'])
```

```
count = df_bh.groupby(['number_treatments']).size()
count.head(30)
```

```
number_treatments
     759
0     91
1    669
dtype: int64
```

Imagen - number_treaments después de cambiar los valores, los espacios en blanco son cada vez más bajos

- Reemplazar valores 0

  El siguiente paso es verificar los 0, para lo cual se asigna otro dataframe con las columnas number_treatments, date_t1, date_t2 con el fin de verificar los datos.

```
df_pru = df_bh[['date_t1', 'date_t2', 'number_treatments']]

df_pru.head()
```

	date_t1	date_t2	number_treatments
0	4/13/2015	5/21/2015	1
1	4/13/2015	5/21/2015	1
2	4/13/2015	5/19/2015	1
3	4/13/2015	5/21/2015	1
4	4/13/2015	5/21/2015	1

Imagen 23 - Dataframe df_pru con las tres columnas

Seguidamente filtro por los vacíos de `number_treatments`

```
df_pru = df_pru[df_pru['number_treatments'] == ' '] #

df_pru.head()
```

date_t1	date_t2	number_treatments
4/27/2015		
4/28/2015		
4/28/2015		
4/28/2015		
05.04.2015		

Imagen 24 - Dataframe df_pru filtrado con espacios en blanco de number_treaments

la imagen 24 muestra que solo la columna `date_t1` tiene fechas ingresadas pero la `date_t2` son vacíos.

para comprobar que la columna date_t1 no tiene vacíos se utiliza la siguiente instrucción con el método `isspace() == False`

```
df_pru[df_pru.date_t1.apply(lambdax:x.isspace()==False)]
```

	date_t1	date_t2	number_treatments
34	4/27/2015		
39	4/28/2015		
40	4/28/2015		
48	4/28/2015		
49	05.04.2015		
...	...	...	...
1514	2/22/2017		
1517	2/23/2017		
1518	2/23/2017		
1519	2/23/2017		
1520	2/23/2017		

759 rows × 3 columns

Imagen - solo la columna date_t1 tiene datos

De acuerdo a la imagen anterior solo la columna `date_t1` del dataframe `df_pru` contiene datos.

Lo siguiente es verificar si las columnas `number_treatments` y `date_t2` están vacías.

```
df_pru[df_pru.number_treatments.apply(lambda
x:x.isspace() == False)]
```

```
df_pru[df_pru.number_treatments.apply(lambda x:x.isspace() == False)]
```

date_t1	date_t2	number_treatments

Imagen - la columna number_treatments vacía

```
df_pru[df_pru.date_t2.apply(lambda x:x.isspace() ==
False)]
```

```
df_pru[df_pru.date_t2.apply(lambda x:x.isspace() == False)]
```

date_t1	date_t2	number_treatments

Imagen - la columna date_t2 vacía

En las anteriores imágenes se muestra que las 2 columnas están vacías, solo date_t1 contiene datos por lo tanto mientras number_treatments está vacío se asigna '0' a los valores restantes de number_treaments

```
df_bh['number_treatments']=df_bh['number_treatments'].str.replace(" ", "0")
```

Ahora la variable target `number_treatments` está balanceada

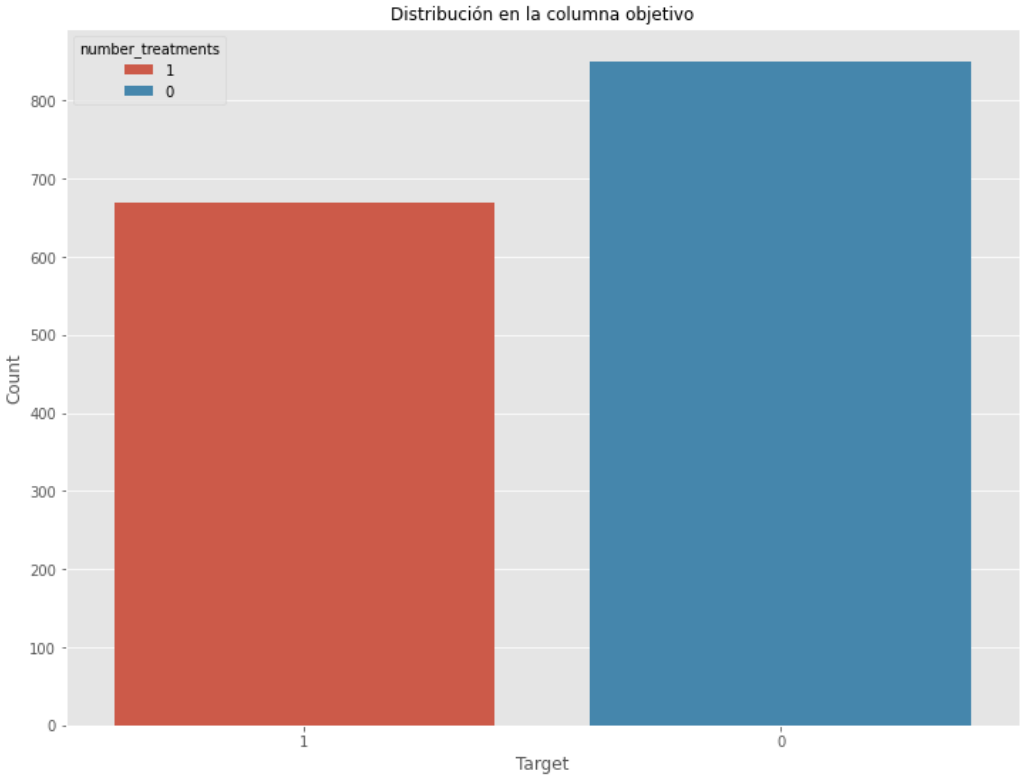


Imagen 25 - number_treatments size() es 0 = 850 registros, 1 = 669 registros

- Feature selection

  Debido a la cantidad de variables presentes en el dataset es conveniente realizar un feature selection con el fin de seleccionar las más importantes y luego lanzar el modelo en la siguiente fase.

  La selección de características es el proceso de reducir el número de variables de entrada al desarrollar un modelo predictivo. Es deseable reducir el número de variables de entrada tanto para reducir el coste computacional de la modelización como, en algunos casos, para mejorar el rendimiento del modelo.

  Los métodos de selección de características basados en la estadística implican la evaluación de la relación entre cada variable de entrada y la variable objetivo mediante la estadística y la selección de las variables de entrada que tienen la relación más fuerte con la variable objetivo. Estos métodos pueden ser rápidos y eficaces, aunque la elección de las medidas estadísticas depende del tipo de datos de las variables de entrada y de salida. *(Brownlee, 2020)*

  Previo al proceso de feature selection es importante preparar los datos para tal efecto, dado que el dataset tiene tantos espacios vacíos estos serán tratados como NAN, así mismo los valores atípicos que no representan datos o información coherente para las columnas.

  Por ejemplo, las columnas `psych_onset` la cual indica la edad del paciente cuando ingresa a tratamiento tiene valores atípicos como ‘2007’ y ‘2011’, los cuales evidentemente no representan una edad.

```
# size psych_onset
count = df_bh.groupby(['psych_onset']).size()
count.sort_values(ascending=False)
```

```
psych_onset
          337
9999       74
40         63
50         61
45         47
          ...
3           1
2011        1
6           1
2007        1
11          1
Length: 69, dtype: int64
```

Imagen 26 - La columna psych_onset tiene algunos outliers

Lo mismo sucede con las variables `age_first_o_treatment` y `age_inpatient` las cuales son variables de edad del paciente y tienen valores como 4444, 458, 411.

```
df_bh.drop(df_bh[df_bh.psych_onset == '2007'].index,
inplace=True)

df_bh.drop(df_bh[df_bh.psych_onset == '2011'].index,
inplace=True)

df_bh.drop(df_bh[df_bh.age_first_o_treatment ==
'4444'].index, inplace=True)

df_bh.drop(df_bh[df_bh.age_first_o_treatment ==
'458'].index, inplace=True)

df_bh.drop(df_bh[df_bh.age_first_o_treatment ==
'421'].index, inplace=True)

df_bh.drop(df_bh[df_bh.age_first_o_treatment ==
'411'].index, inplace=True)

df_bh.drop(df_bh[df_bh.age_first_o_treatment ==
'404'].index, inplace=True)

df_bh.drop(df_bh[df_bh.age_inpatient == '404'].index,
inplace=True)

df_bh.drop(df_bh[df_bh.age_inpatient == '458'].index,
inplace=True)
```

Adicionalmente el dataset contiene muchos valores atípicos con ‘9999’ o ‘999’ y espacios vacíos por ejemplo en la imagen 29 se ve la cantidad de espacios vacíos y 9999 que tiene una variable, más que outliers estos son valores erróneos del dataset que generan ruido por lo tanto la forma de tratar estos valores es reemplazarlos por NAs para luego comprobar cuales son las columnas con más missing values y eliminarlas del dataframe.

```
df_bh = df_bh.replace([' '],np.nan)

df_bh = df_bh.replace(['9999'],np.nan)

df_bh = df_bh.replace(['999'],np.nan)
```

```
df_bh = df_bh.replace(['9999.00'],np.nan)

df_bh = df_bh.replace(['9999.0'],np.nan)

df_bh = df_bh.replace([9999],np.nan)
```

Finalmente se comprueban los cambios:

```
total = df_bh.isnull().sum().sort_values(ascending =
False)

percent = (df_bh.isnull().sum() /
df_bh.isnull().count()).sort_values(ascending = False)

missing_data = pd.concat([total, percent], axis = 1,
keys = ['Total', 'Percent'])

missing_data.head(10)
```

	Total	Percent
diagnosis	1519	1.000000
PSA404_t2	1501	0.988150
blood_samples	1498	0.986175
leu410_t2	1491	0.981567
pro413_t2	1491	0.981567
glu414_t2	1491	0.981567
ket415_t2	1491	0.981567
pH412_t2	1491	0.981567
ubg416_t2	1491	0.981567
nit411_t2	1491	0.981567

Imagen 27 - Valores nulos en el dataframe después de los reemplazos previos

Los missing values son los valores más frecuentes y comunes en el dataframe con una frecuencia de hasta 1491 registros equivalente a un 98% de 1519 filas.

Para efectos del modelo y de la cantidad de columnas las variables son divididas en enteras y float, las variables enteras son 330 variables las cuales se agrupan en otro dataframe para luego unirlas con el grupo de float

```
df_int=df_bh[['psych_onset','rehab_days','sex','BZ394_t1','re
hab_weeks','current_psy_treatment','outpatient_psy_treatment'
...]]
```

En el dataframe `df_int` se guardan las variables `int` y lo siguiente es verificar las columnas con más missing values y convertirlas a `int` puesto la mayoría son tipo object hay unas marcadas como float pero realmente son enteras no tienen valores decimales.

```
total = df_int.isnull().sum().sort_values(ascending =
False)percent=(df_int.isnull().sum()/df_int.isnull().count())
.sort_values(ascending = False)

missing_data = pd.concat([total, percent], axis = 1, keys =
['Total', 'Percent'])

missing_data.head(15)
```

El resultado anterior muestra la cantidad de missing values dentro del dataframe.

	Total	Percent
bil417_t2	1483	0.981469
pro413_t2	1483	0.981469
ubg416_t2	1483	0.981469
ery418_t2	1483	0.981469
glu414_t2	1483	0.981469
leu410_t2	1483	0.981469
nit411_t2	1483	0.981469
QuinoA_t1	1414	0.935804
Kyn_t1	1414	0.935804
QuinoA_t2	1414	0.935804
Trp_t1	1414	0.935804
Kyn_t2	1414	0.935804
Trp_t2	1414	0.935804
other	1301	0.861019
rehab_weeks	1111	0.735275

Imagen 28 - Valores nulos del dataframe df_int, total y porcentaje

Las columnas con más missing values son eliminadas:

```
df_int=df_int.drop(['bil417_t2','pro413_t2','ubg416_t2',
'ery418_t2', 'glu414_t2', 'leu410_t2', 'nit411_t2',
'QuinoA_t1', 'Kyn_t1', 'QuinoA_t2'...], axis=1)
```

Del dataframe anterior quedan 22 columnas las cuales aún tienen missing values pero no se eliminan puesto se necesitan para el modelo más adelante se eliminarán algunas filas con registros missing values.

Con el fin de convertir las variables a `int` primero se pasan a `float` y luego a `int` puesto de lo contrario no se puede.

La lista de columnas se saca con el método tolist()

```
print(df_int.columns.tolist())

type(df_int.columns.tolist())
```

Primero se convierten a float con el método `astype('float')`

```
df_int[['sex',   'BZ394_t1',   'THR323_t1',   'CK355_t1',
'LDH357_t1',   'GOT358_t1',   'GPT359_t1',   'GGT360_t1',
'CHO364_t1',   'NHD375_t1',   'NA376_t1',   'CL378_t1',
'hba396_t1',   'glu414_t1',   'bil417_t1',   'RZAttent',
'RZExecut_2',   'HDL365_t1',   'LDL366_t1',   'TG367_t1',
'number_treatments',        'rehab_weeks_4_6']]        =
df_int[['sex',   'BZ394_t1',   'THR323_t1',   'CK355_t1',
'LDH357_t1',   'GOT358_t1',   'GPT359_t1',   'GGT360_t1',
'CHO364_t1',   'NHD375_t1',   'NA376_t1',   'CL378_t1',
'hba396_t1',   'glu414_t1',   'bil417_t1',   'RZAttent',
'RZExecut_2',   'HDL365_t1',   'LDL366_t1',   'TG367_t1',
'number_treatments', 'rehab_weeks_4_6']].astype('float')
```

Luego a entero con el método `astype('Int64')`

```
df_int[['sex',   'BZ394_t1',   'THR323_t1',   'CK355_t1',
'LDH357_t1',   'GOT358_t1',   'GPT359_t1',   'GGT360_t1',
'CHO364_t1',   'NHD375_t1',   'NA376_t1',   'CL378_t1',
'hba396_t1',   'glu414_t1',   'bil417_t1',   'RZAttent',
'RZExecut_2',   'HDL365_t1',   'LDL366_t1',   'TG367_t1',
'number_treatments',        'rehab_weeks_4_6']]        =
df_int[['sex',   'BZ394_t1',   'THR323_t1',   'CK355_t1',
'LDH357_t1',   'GOT358_t1',   'GPT359_t1',   'GGT360_t1',
'CHO364_t1',   'NHD375_t1',   'NA376_t1',   'CL378_t1',
'hba396_t1',   'glu414_t1',   'bil417_t1',   'RZAttent',
'RZExecut_2',   'HDL365_t1',   'LDL366_t1',   'TG367_t1',
'number_treatments', 'rehab_weeks_4_6']].astype('Int64')
```

El mismo procedimiento para las variables float, las variables float se guardan en el dataframe `df_flo,` y las comas se reemplazan por puntos.

```
df_flo=df_bh[['group','group_affective','age','height_m'
,'weight','Hip_t1','Hip_t2','Weight_T1'...]]
```

De esta manera se reemplazan comas por puntos:

```
df_flo=df_flo.apply(lambdax:x.str.replace(',','.'))
```

Se verifica la cantidad de missing values por columna en el nuevo dataframe:

```
total = df_flo.isnull().sum().sort_values(ascending =
False)

percent=(df_flo.isnull().sum()/df_flo.isnull().count()).
sort_values(ascending = False)

missing_data = pd.concat([total, percent], axis = 1, keys
= ['Total', 'Percent'])

missing_data.head(10)
```

	Total	Percent
PSA404_t2	1493	0.988087
pH412_t2	1483	0.981469
ket415_t2	1483	0.981469
spg419_t2	1483	0.981469
A2G438_t2	1465	0.969557
A1G437_t2	1465	0.969557
a2g446_t2	1465	0.969557
GG441_t2	1465	0.969557
B2G440_t2	1465	0.969557
B1G439_t2	1465	0.969557

Imagen 29 - Valores nulos del dataframe df_flo, total y porcentaje

Al igual que en el dataframe de variables enteras son muchas las columnas que quedaron con missing values por lo tanto se eliminan las que tengan más missings.

```
df_flo=df_flo.drop(['PSA404_t2','pH412_t2','group_affect
ive', 'group', 'ket415_t2', 'spg419_t2','A2G438_t2',
'A1G437_t2'...], axis=1)
```

Después de eliminar las columnas con el número de missings más alto quedan 110 columnas float

Igual que el procedimiento anterior con el método tolist() se listan las columnas restantes

```
print(df_flo.columns.tolist())
```

```
type(df_flo.columns.tolist())
```

Las variables restantes quedan en el dataframe `df_flo` y se cambia el tipo de dato a `float` con la siguiente instrucción:

```
df_flo[df_flo.columns]=df_flo[df_flo.columns].apply(pd.to_numeric,errors='coerce')
```

```
df_flo.dtypes

age                    float64
height_m               float64
weight                 float64
Weight_T1              float64
Weight_T2              float64
                         ...
ZExecutive_function_2  float64
BMI_T1                 float64
CRh393_t1              float64
IL6457_t1              float64
height_cm              float64
Length: 110, dtype: object
```

Imagen 30 - Variables float del dataframe df_flo

Finalmente se concatenan los dos dataframes los cuales tienen el mismo número de filas en nuevo dataframe llamado `df_concat` con el cual se trabajará desde este punto.

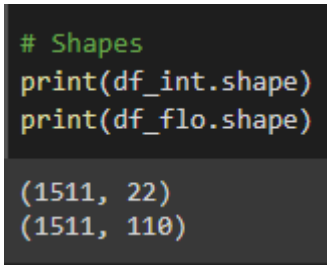


Imagen 31 - Shapes de los dos dataframes para concatenar

```
df_concat = pd.concat([df_int, df_flo], axis="columns")
```

Aún quedan algunas columnas con missing values que no se eliminaron en este punto se eliminan sólo las filas que contengan missing values

```
total = df_concat.isnull().sum().sort_values(ascending =
False)

percent = (df_concat.isnull().sum() /
df_concat.isnull().count()).sort_values(ascending =
False)

missing_data = pd.concat([total, percent], axis = 1,
keys = ['Total', 'Percent'])

missing_data.head(10)
```

	Total	Percent
Weight_T2	97	0.064196
height_cm	96	0.063534
height_m	96	0.063534
BMI_T1	96	0.063534
Weight_T1	96	0.063534
weight	96	0.063534
LDH357_t1	89	0.058901
ket415_t1	88	0.058240
K377_t1	82	0.054269
CRh393_t1	82	0.054269

Imagen 32 - Los valores restantes que faltan en dataframe df_concat

Con las siguientes instrucciones los missing values son eliminados:

```
df_concat = df_concat.dropna(subset=['Weight_T2'])

df_concat = df_concat.dropna()
```

Finalmente, ya no hay missing values, ni valores erróneos en el dataset para lanzar el algoritmo del feature selection.

```
total = df_concat.isnull().sum().sort_values(ascending =
False)

percent=(df_concat.isnull().sum()/df_concat.isnull()

.count()).sort_values(ascending = False)

missing_data = pd.concat([total, percent], axis = 1,
keys = ['Total', 'Percent'])

missing_data.head(10)
```

	Total	Percent
sex	0	0.0
ZCVLT_WAI_RW2	0	0.0
Pos_zFWIT_INT2	0	0.0
Pos_zFWIT_FSB2	0	0.0
Pos_zFWIT_FWL2	0	0.0
Pos_zCVLT_VFWII_RW2	0	0.0
Pos_zCVLT_VFWI_RW2	0	0.0
Pos_zTMTB2	0	0.0
Pos_zTMTA2	0	0.0
ZTMTB2	0	0.0

Imagen 33 - valores nulos fueron eliminados, dataframe df_concat libre de nulos

Se hicieron tres modelos de feature selection para el proyecto con el cual se estableció un número de variables para trabajar el modelo machine learning, ya que se tienen muchas variables que aún después de eliminar missing values, muchas de estas representan ruido y lo mejor es seleccionar las más representativas para predecir el target variable, algunas variables pueden ser útiles en combinación con otras por eso es que principalmente este tipo de procedimiento lo que hace es centrarse en eliminar variables no informativas o redundantes del modelo. Tener en el modelo variables irrelevantes puede reducir la precisión de este, especialmente en este caso se usará la Regresión Logística.

Algunas ventajas del feature selection son:

- Reduce el sobreajuste: Menos datos redundantes significa menos oportunidades de tomar decisiones basadas en el ruido.
- Mejora la precisión: Menos datos erróneos significan que la precisión de los modelos mejora.
- Reduce el tiempo de entrenamiento: Menos datos significa que los algoritmos se entrenan más rápido.

El proyecto se basa en un modelo de clasificación con el cual vamos a predecir si un paciente es internado es decir valores SI y NO. En modelos de Regresión Logística

la función sigmoide es utilizada para devolver la probabilidad de una etiqueta en 1 o 0, si o no, etc.

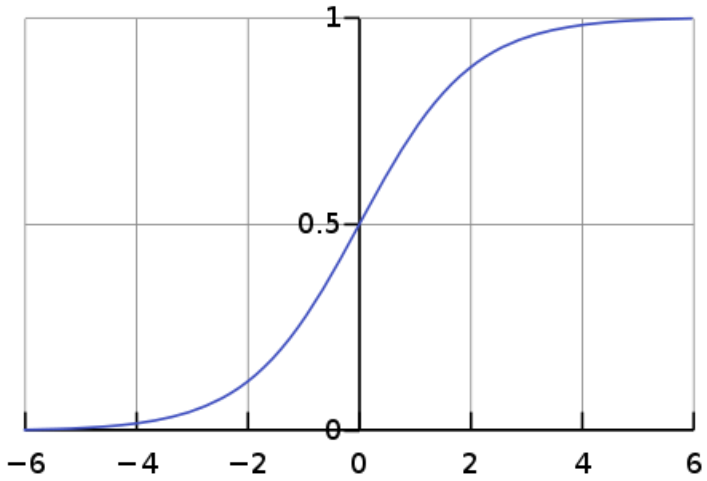


Imagen 34 - Curva de regresión logística fuente www.en.wikipedia.org/wiki/Sigmoid_function

El procedimiento se realiza con las librerías de Scikit-Learn, lo primero es escalar los datos `z = (x - u) / s`, es decir variable de entrada se resta la media y dividiendo por la desviación estándar para desplazar la distribución para que tenga una media de cero y una desviación estándar de uno.

```
from sklearn.preprocessing import StandardScaler

from sklearn.model_selection import train_test_split

X = df_concat.drop('number_treatments', axis=1)

y = df_concat[['number_treatments']]

y=y.astype('int')

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)
```

También se importa el `train_test_split` puesto se dividir el conjunto de datos en training y test para la ejecución del algoritmo

```
ss = StandardScaler()

X_train_scaled = ss.fit_transform(X_train)

X_test_scaled = ss.transform(X_test)

from sklearn.linear_model import LogisticRegression

model = LogisticRegression()

model.fit(X_train_scaled, y_train)

importances = pd.DataFrame(data={

    'Feature': X_train.columns,

    'Importancia': model.coef_[0]

})

importances = importances.sort_values(by='Importancia',
ascending=False)
```

Para ver en gráfico las mejores variables se crea otra columna en el dataframe importances con el valor absoluto puesto el modelo de Logistic Regression tiene valores positivos y negativos:

```
importances['abs_Importancia']=importances['Importancia'
].abs()

max_int_abs=importances.sort_values(by='abs_Importancia'
, ascending=False)

max_int_abs.head(12)
```

	Feature	Importancia	abs_Importancia
58	ZCVLT_LS_DG1_5_RW1	0.972647	0.972647
28	HBE318_t1	0.899299	0.899299
32	HKT322_t1	-0.866317	0.866317
62	ZCVLT_WAII_RW1	-0.777118	0.777118
20	rehab_weeks_4_6	-0.696785	0.696785
16	RZExecut_2	-0.676651	0.676651
99	NZExecut_2	0.629837	0.629837
26	ERY316_t1	0.614716	0.614716
108	NZDiff_a	0.479154	0.479154
29	MCV319_t1	-0.427788	0.427788
113	RZDiff_a	-0.426961	0.426961
4	LDH357_t1	0.419762	0.419762

Imagen 35 - Dataframe ordenado por abs_importancia con valor absoluto aplicado

```
pred = max_int_abs['Feature'].head(20)

cof = max_int_abs['abs_Importancia'].head(20)

# Figure Size

fig, ax = plt.subplots(figsize =(16, 9))

# Horizontal Bar Plot

ax.barh(pred, cof)

# Show top values

ax.invert_yaxis()

ax.set_title('Top 20 - Feature Regresión Logística
number_treatments',loc ='left', )
```

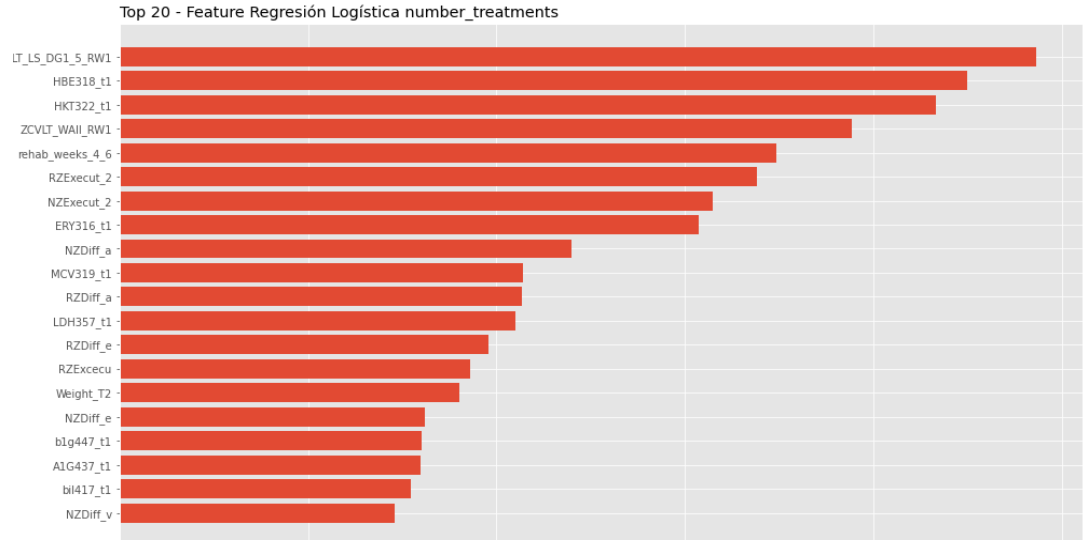


Imagen 36 - Top 20 del feature selection hecho con el modelo de regresión logística

Finalmente, el modelo será construído con las 12 variables más importantes, las más representativas de este modelo.

`ZCVLT_LS_DG1_5_RW1, HBE318_t1, HKT322_t1,ZCVLT_WAII_RW1, rehab_weeks_4_6, RZExecut_2, NZExecut_2, ERY316_t1, NZDiff_a, MCV319_t1, RZDiff_a, LDH357_t1.`

El siguiente modelo aplicado fue el de mRMR el cual significa "minimum Redundancy - Maximum Relevance", mRMR es un algoritmo de selección de características mínimamente óptimo utilizado por la plataforma de aprendizaje automático de Uber para encontrar el subconjunto de características "mínimamente óptimo". Esto significa que está diseñado para encontrar el subconjunto más pequeño de características relevantes para una tarea de aprendizaje automático determinada.

“Maximum Relevance - Minimum Redundancy” se llama así porque -en cada iteración- queremos seleccionar la característica que tenga la máxima relevancia con respecto a la variable objetivo y la mínima redundancia con respecto a las características que se han seleccionado en iteraciones anteriores. En la práctica, en cada iteración i, se calcula una puntuación para cada característica a evaluar (f):

$$score_i(f) = \frac{relevance(f \mid target)}{redundancy(f \mid features\ selected\ until\ i-1)}$$

Imagen 37 - Idea general del modelo mRMR fuente: www.towardsdatascience.com

La relevancia de una característica f en la iteración i-ésima (numerador) se calcula como el estadístico F entre la característica y la variable objetivo (aquí para saber más sobre la prueba F). La redundancia (denominador) se calcula como la correlación media (Pearson) entre la característica y todas las características que se han seleccionado en iteraciones anteriores. Finalmente, esta es la fórmula

$$score_i(f) = \frac{F(f, target)}{\sum_{s \in features\ selected\ until\ i-1} |corr(f,s)| \ / \ (i-1)}$$

Imagen 38 - Ecuación del modelo mRMR fuente: www.towardsdatascience.com

donde i es la iteración i-ésima, f es la característica que se está evaluando, F es el estadístico F y corr es la correlación de Pearson. Tenga en cuenta que la correlación se toma en valor absoluto. De hecho, si dos características tienen una correlación de 0,9 o de -,9, no hay ninguna diferencia: en ambos casos son muy redundantes. *(Mazzanti, Samuele , 2021)*

Lo primero es instalar el paquete, `pip install mrmr_selection` e importar `import mrmr`

Es necesario convertir el target variable a Pandas series, se aplica el método squeeze(), `y = df_concat['number_treatments'].squeeze()`

```
X = pd.DataFrame(df_concat)

y = pd.Series(y)

dfbh_features = mrmr_classif(X=X, y=y, K=13)

print(dfbh_features)
```

El número de parámetros requerido se indica en `K,` igual que en el método anterior se eligen 12 variables, 13 más el target

El resultado es una lista: `['number_treatments', 'rehab_weeks_4_6', 'HKT322_t1', 'LDH357_t1', 'MCC320_t1', 'NZVerbal_2', 'spg419_t1', 'RZExcecu', 'FT4402_t1', 'RZVerbal', 'height_m', 'ERY316_t1', 'RZVerbal_2']`

Algunas variables en común fueron seleccionadas por los dos métodos

Logistic Regression	**mRMR**
ZCVLT_LS_DG1_5_RW1	rehab_weeks_4_6
HBE318_t1	HKT322_t1
HKT322_t1	LDH357_t1
ZCVLT_WAII_RW1	MCC320_t1
rehab_weeks_4_6	NZVerbal_2
RZExecut_2	spg419_t1
NZExecut_2	RZExcecu
ERY316_t1	FT4402_t1
NZDiff_a	RZVerbal
MCV319_t1	height_m
RZDiff_a	ERY316_t1
LDH357_t1	RZVerbal_2

El otro modelo de feature selection es random forest el cual será utilizado con un modelo de random forest, éstos ofrecen un buen rendimiento predictivo, un bajo nivel de sobreajuste y una fácil interpretabilidad. Esta interpretabilidad viene dada por el hecho de que es sencillo derivar la importancia de cada variable en la decisión del árbol los bosques aleatorios o random forest están formados por centenares de árboles de decisión, cada uno de ellos construido sobre una extracción aleatoria de las observaciones del conjunto de datos y una extracción aleatoria de las características. No todos los árboles ven todas las características ni todas las observaciones, lo que garantiza que los árboles están descorrelacionados y por lo tanto sean menos propensos a un ajuste excesivo.

Para la clasificación, la medida de impureza es la impureza de *Gini* o la ganancia de información/entropía. Para la regresión, la medida de impureza es la varianza.

Por lo tanto, al entrenar un árbol, es posible calcular cuánto disminuye la impureza de cada característica. Cuanto más disminuya una característica la impureza, más importante será la característica. En los bosques aleatorios, la disminución de la impureza de cada característica puede promediarse entre los árboles para determinar la importancia final de la variable. *(Dubey, Akash , 2018)*

Muy parecido a los métodos anteriores donde se hace un `train_test_split` la implementación se hace con scikit-learn la función `RandomForestRegressor`

La variable target es `number_treatments` la cual se asigna en `y`, el resto del dataframe `df_concat` en `X`

```
X = df_concat.drop('number_treatments', axis=1)

y = df_concat[['number_treatments']]

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)
```

Aquí se hace el fit del modelo usando el `RandomForestRegressor` de scikit-learn, `n_estimators` es el número de árboles por defecto es 100, 150 se suele utilizar mucho.

```
rf = RandomForestRegressor(n_estimators=150)

rf.fit(X_train, y_train)
```

De esta manera se aprecian todas las variables del dataframe en orden descendente, pero como son tantas la imagen no es clara, (ver imagen 39) y no se usarán todas se hace un nuevo gráfico con menos variables. Las importancias de las características son proporcionadas por el atributo ajustado `feature_importances_` y se calculan como la media y la desviación estándar de la acumulación de la disminución de impurezas dentro de cada árbol.

Las características situadas más arriba del árbol contribuyen a la predicción final de una fracción mayor de las muestras de entrada que las características descendentes en el árbol. La importancia de las características es una estimación de la fracción de la clasificación de las muestras de entrada a la que contribuye una característica.

1.0 significa que tiene una característica que por sí sola clasifica todas las muestras, 0.0 indicaría una característica que no puede añadir ningún valor (adicional) para la clasificación.

Random forest calcula una media de estas estimaciones para reducir la varianza de las estimaciones de importancia.

```
plt.figure(figsize=(25,22))

plt.yticks(fontsize=6)

plt.title('Feature Random Forest number_treatments',
size=20)

sort = rf.feature_importances_.argsort()

plt.barh(X.columns[sort], rf.feature_importances_[sort])

plt.xlabel("Features")

plt.savefig("output.jpg")
```

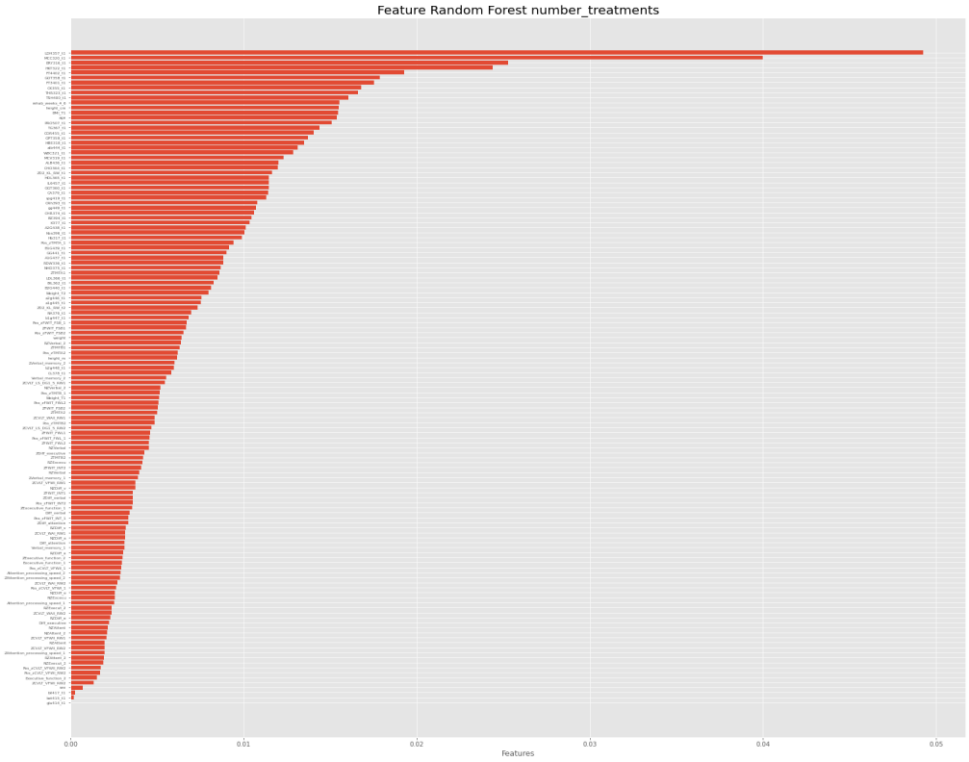


Imagen 39 - Variables del feature selection, random forest

Las variables más importantes se agrupan en otro dataframe con el coeficiente y su importancia y se ordenan en orden descendente para ver su importancia

```
coef_numb_treat = pd.DataFrame({'predictor':
X.columns,'coef':rf.feature_importances_})
```

Aquí se asigna al dataframe `max_int`

```
max_int = coef_numb_treat.sort_values(by='coef',
ascending=False)
```

Con la librería `matplotlib` de Python se hace el diagrama horizontal de barras ordenado

```
pred = max_int['predictor'].head(20)

cof = max_int['coef'].head(20)

 # Figure Size

fig, ax = plt.subplots(figsize =(16, 9))

# Horizontal Bar Plot

ax.barh(pred, cof)

# Show top values

ax.invert_yaxis()

ax.set_title('Top 20 - Feature Random Forest
number_treatments',loc ='left',)
```

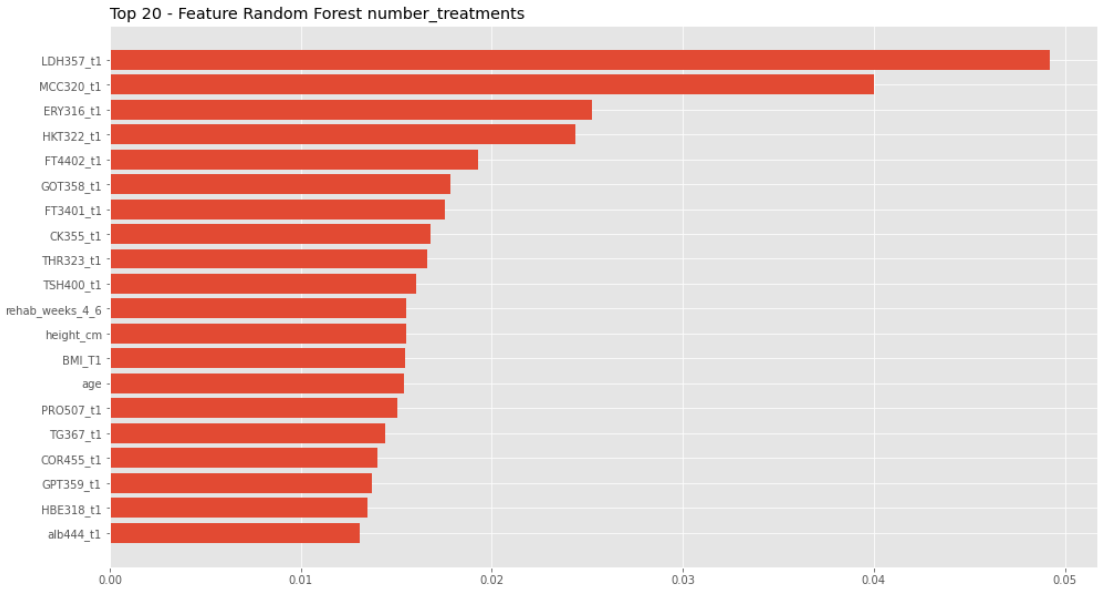


Imagen 40 - Primeras 20 features más importantes del feature selection random forest en orden descendente

De acuerdo con lo anterior se crean tres dataframes para los modelos

`df_rl`: dataframe creado a partir del feature selection de Regresión Logística, se utilizan 12 variables más el target.

```
df_rl = df_concat[['number_treatments',
'ZCVLT_LS_DG1_5_RW1', 'HBE318_t1', 'HKT322_t1',
'ZCVLT_WAII_RW1', 'rehab_weeks_4_6', 'RZExecut_2',
'NZExecut_2', 'ERY316_t1', 'NZDiff_a', 'MCV319_t1',
'RZDiff_a', 'LDH357_t1']]
```

`df_mr`: dataframe creado a partir del feature selection de mRMR , se utilizan 12 variables más el target.

```
df_mr = df_concat[['number_treatments',
'rehab_weeks_4_6', 'HKT322_t1', 'LDH357_t1',
'MCC320_t1', 'NZVerbal_2', 'spg419_t1', 'RZExcecu',
'FT4402_t1', 'RZVerbal', 'height_m', 'ERY316_t1',
'RZVerbal_2']]
```

`df_rf`: dataframe creado a partir del feature selection de Random Forest , se utilizan 12 variables más el target.

```
df_rf = df_concat[['number_treatments', 'LDH357_t1',
'MCC320_t1', 'ERY316_t1', 'HKT322_t1', 'FT4402_t1',
```

```
'GOT358_t1', 'FT3401_t1', 'CK355_t1', 'THR323_t1',
'TSH400_t1', 'BMI_T1', 'age']]
```

Definición de las Variables

De acuerdo con los resultados anteriores cada Dataframe tiene un grupo de variables las cuales algunas son comunes entre los dataframes, se definen a continuación.

`ZCVLT_LS_DG1_5_RW1`: es el valor z-score del California verbal learning test (sum1-5) t1. California Verbal Learning Test es un test de aprendizaje de listas de palabras utilizado para la memoria verbal, la lista A de palabras (learning list) se presenta un total de cinco veces (DG 1-5). Z-score es el número de desviaciones estándar de la media de un punto de información $Z = (X - \mu) / \sigma$, donde X es el puntaje del test, μ la media de todos los puntajes y σ la desviación estándar de todos los puntajes.

`HBE318_t1`: nivel total de hemoglobina al ingresar el paciente.

`HKT322_t1`: hematocritos, es el volumen de glóbulos rojos en la sangre.

`ZCVLT_WAII_RW1`: California Verbal Learning Test es un test de aprendizaje de listas de palabras utilizado para la memoria verbal. Corresponde al valor Z-score del recuerdo diferido a largo plazo al ingresar el paciente, WA-se refiere a la recuperación con asistencia.

`rehab_weeks_4_6`: más o menos de 4 semanas (sin especificación exacta). Tiene dos valores: 1 hasta 4 semanas, 2 más de 4 semanas.

`RZExecut_2`: rango del valor z-score de Executive_function_2. Executive_function es un conjunto de habilidades mentales que incluyen la memoria de trabajo, el pensamiento flexible y el autocontrol.

`NZExecut_2`: puntaje normal ZExecutive_function_2 usando el método Ranking.

`ERY316_t1`: eritrocitos o glóbulos rojos, es un tipo de células sanguíneas.

`NZDiff_a`: puntaje normal de ZDiff_attention usando el metodo Ranking, Diff_attention es la diferencia entre T1 y T2 (admisión y salida) Atención y velocidad de procesamiento Puntuaciones compuestas.

`MCV319_t1`: volumen corpuscular medio, la prueba MCV mide el tamaño medio de los glóbulos rojos.

`RZDiff_a`: rango de ZDiff_attention, puntaje z-zcore - diferencia entre T1 y T2 (admisión y salida) Atención y velocidad de procesamiento Puntuaciones compuestas

`LDH357_t1`: nivel de ácido láctico deshidrogenasa en la sangre.

`MCC320_t1`: concentración media de hemoglobina corpuscalar.

`NZVerbal_2`: puntaje normal de ZVerbal_memory_2 usando el método Ranking.

`spg419_t1`: densidad relativa de la orina.

`RZExcecu`: rango de ZExcecutive_function_1

`FT4402_t1`: tiroxina libre (T4), el test de tiroxina es un análisis de sangre que ayuda a diagnosticar las enfermedades de la tiroides.

`RZVerbal`: rango de ZVerbal_memory_1

`RZVerbal_2`: rango de ZVerbal_memory_2

`GOT358_t1`: ASAT (GOT) aspartato aminotransferasa es una enzima que se encuentra en varios tejidos del cuerpo y se utiliza principalmente para diagnosticar y controlar las enfermedades del hígado.

`FT3401_t1`: tiroxina libre (T3), La tiroxina es una hormona que la glándula tiroides segrega en el torrente sanguíneo.

`CK355_t1`: creatina quinasa, es una enzima expresada por varios tejidos y tipos celulares.

`THR323_t1`: trombocitos o plaquetas, componente de la sangre cuya función es reaccionar a las hemorragias por lesiones de los vasos sanguíneos

`TSH400_t1`: TSH hormona estimulante de la tiroides basal. La prueba de TSH es un análisis de sangre que mide esta hormona

`height_m`: altura de cada paciente expresda en metros.

`age`: edad de cada paciente está indicada en años y días.

`BMI_T1`: índice de masa corporal es una medida de adiposidad u obesidad.

Estas son las variables comunes en los tres datasets *HKT322_t1*, *ERY316_t1*, *LDH357_t1*, las cuales son somáticas referentes a valores físicos corporales.

Lo siguiente es ver con gráficos los datos y cifras de los dataframes, por lo tanto, se realiza el siguiente `pairplot` del paquete `seaborn` del dataframe `df_rl`

```
sns.set()

cols = ['number_treatments', 'ZCVLT_LS_DG1_5_RW1',
'HBE318_t1', 'HKT322_t1', 'ZCVLT_WAII_RW1',
'rehab_weeks_4_6', 'RZExecut_2', 'NZExecut_2',
'ERY316_t1', 'NZDiff_a', 'MCV319_t1', 'RZDiff_a',
'LDH357_t1']

sns.pairplot(df_rl[cols], height = 2.5)

plt.show();
```

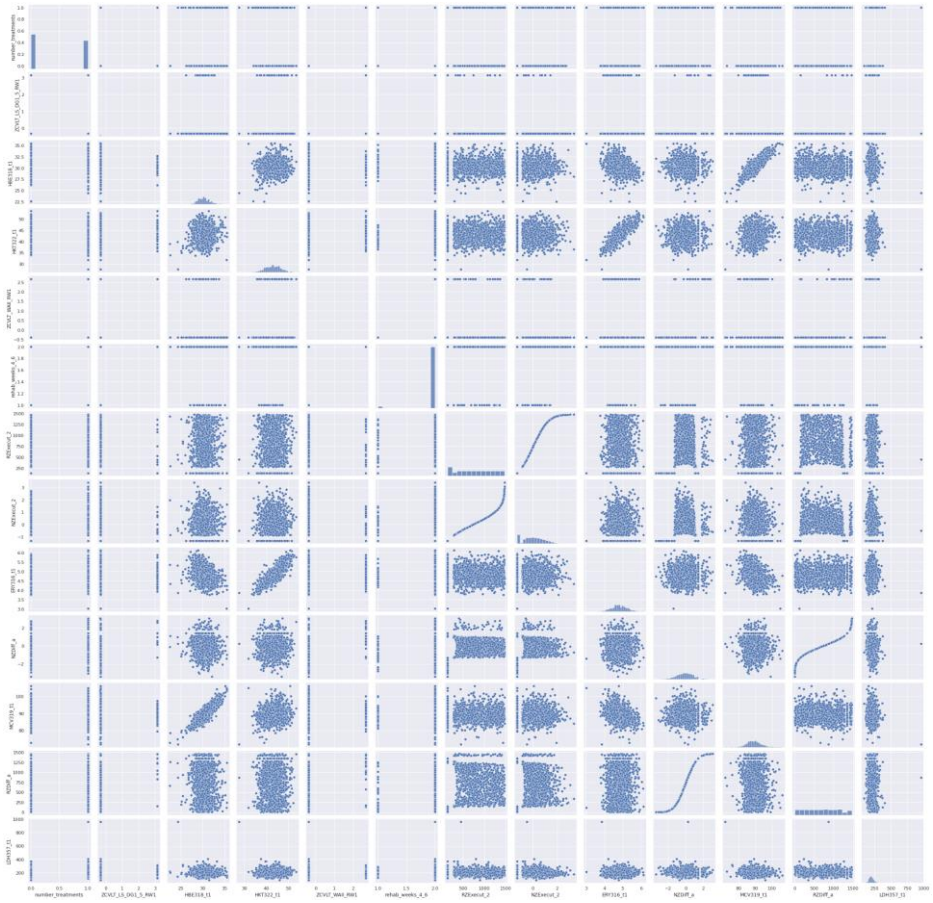


Imagen 41 - Pairplot del dataframe df_rl

En el pairplot se aprecia cómo las variables están concentradas en un rango aunque presentan algunos atípicos fuera de ese grupo. En general los datos mantienen un rango y unos niveles entre ellos, no hay tanta dispersión de datos ya que estos son datos psicológicos dentro de los cuales se tienen rangos y puntuaciones que finalmente son similares entre muchos pacientes.

En el caso de `HBE318_t1` - `HKT322_t1` la mayoría de los datos están en un rango entre 27.5 y 32.5 y 35 - 50 aproximadamente. También con la variable `HBE318_t1` hay tendencia al alta con `MCV319_t1`, igualmente entre `HKT322_t1` y `ERY316_t1` lo que indica que estas variables están correlacionadas y pueden ser redundantes en el modelo.

El siguiente `pairplot` del dataframe df_mr

```
sns.set()cols = ['number_treatments', 'rehab_weeks_4_6',
'HKT322_t1','LDH357_t1','MCC320_t1','NZVerbal_2',
'spg419_t1','RZExcecu','FT4402_t1','RZVerbal','height_m'
,'ERY316_t1', 'RZVerbal_2']

sns.pairplot(df_mr[cols], height = 2.5)

plt.show();
```

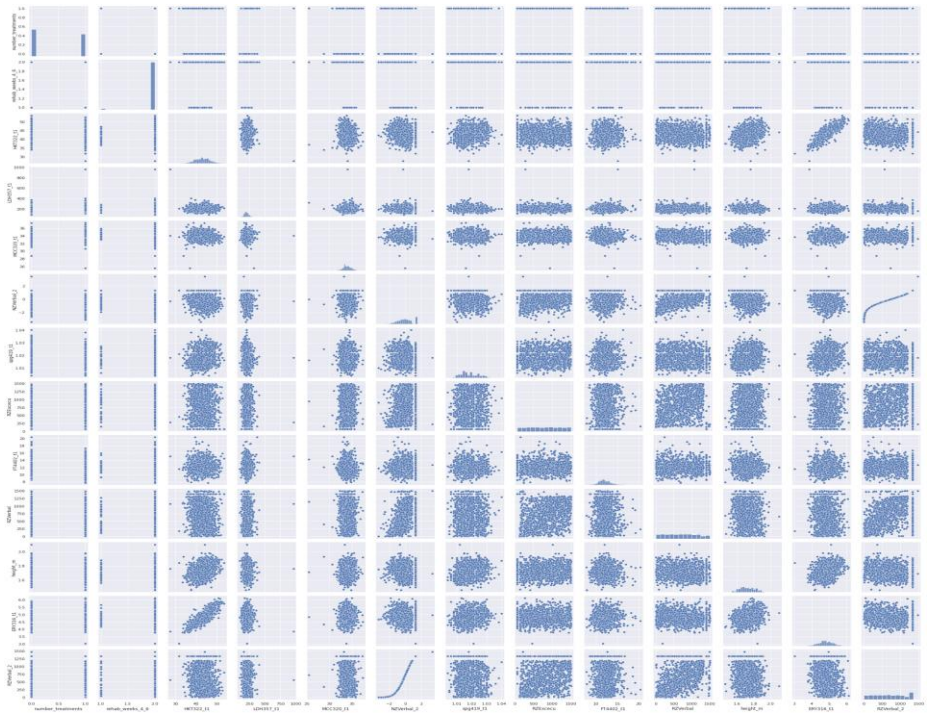


Imagen 42 - Pairplot del dataframe df_mr

Igual con el dataframe anterior los datos se encuentran agrupados en en rango de cifras, hay alguna tendencia o correlación entre variables `HKT322_t1` - `ERY316_t1` lo cual puede ser redundante para el modelo.

El siguiente `pairplot` del dataframe df_rf

```
sns.set()

cols = ['number_treatments', 'LDH357_t1', 'MCC320_t1',
'ERY316_t1','HKT322_t1','FT4402_t1','GOT358_t1','FT3401_
t1','CK355_t1','THR323_t1','TSH400_t1', 'BMI_T1', 'age']

sns.pairplot(df_rf[cols], height = 2.5)

plt.show();
```

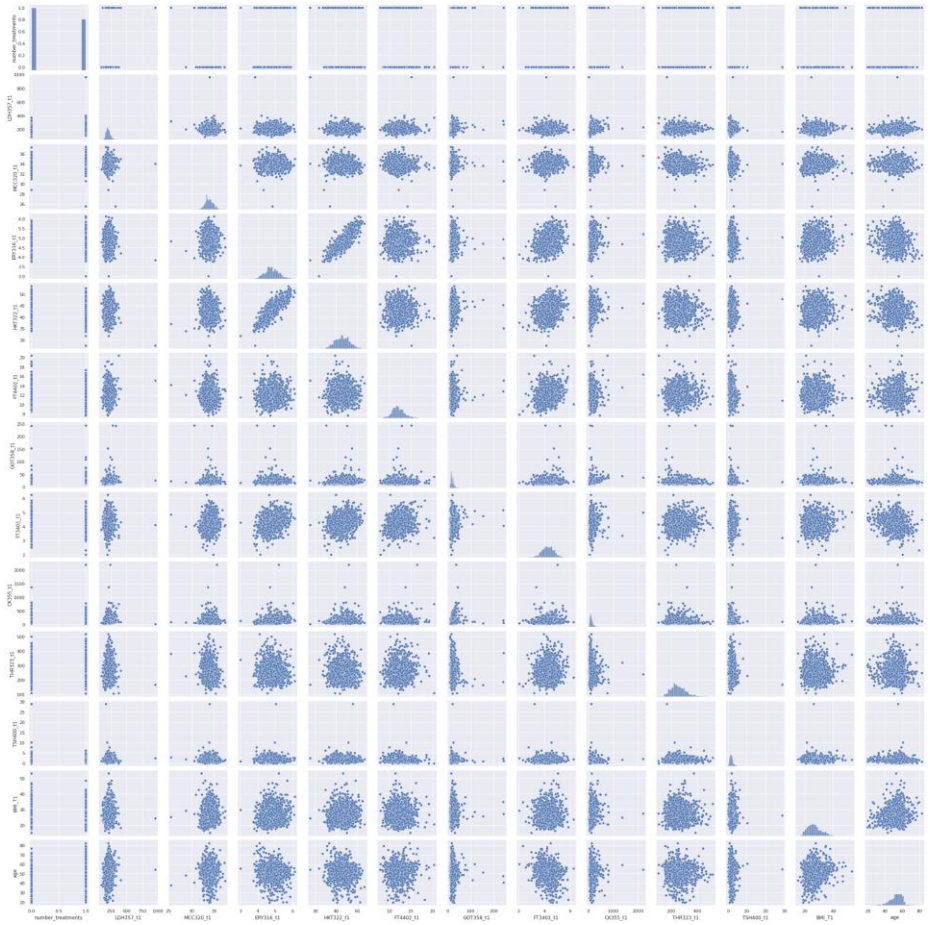


Imagen 43 - Pairplot del dataframe df_rf

Igual con los anteriores dataframes los datos se encuentran agrupados en en rango de cifras, hay alguna tendencia o correlación entre variables `HKT322_t1` - `ERY316_t1` lo cual puede ser redundante para el modelo.

5.4 MODELING

El modelo elegido para este proyecto es de Regresión Logística debido a que estamos clasificando entre dos posibles respuestas basadas en un conjunto de variables independientes, este modelo pertenece a la familia de los modelos de aprendizaje automático supervisado. También se considera un modelo discriminativo, lo que significa que intenta distinguir entre clases (o categorías) es decir este modelo trata de modelar la probabilidad entre una variable binaria en función de otras variables independientes.

La regresión Logística modela la probabilidad de que la variable target pertenezca al grupo de *odds.* Este modelo de regresión modela la probabilidad de que la variable respuesta Y pertenezca a una categoría $\rho(Y = 1 \mid X)$ donde finalmente la fórmula sigmoide $\frac{p(x)}{1 - p(x)} = e^{(w_0 + w_1 X)}$ la cantidad p(x)/[1−p(x)] se conoce como odds y pueden tomar cualquier valor entre 0 que es una probabilidad muy baja y *infinito* (probabilidad muy alta), odds son la relación entre algo que ocurre y algo que no ocurre o el ratio entre la probabilidad de evento verdadero y la probabilidad de evento falso $\frac{p}{q}$.

En el caso de la Regresión Logística múltiple el cual es nuestro caso $P(y = 1|X = x) = e\beta 0 + \beta 1x1 + \ldots + \beta pxp1 + e\beta 0 + \beta 1x1 + \ldots + \beta pxp$ / $1 + e\beta 0 + \beta 1x1 + \ldots + \beta pxp$, también donde $P(y = 1|X = x)$ puede interpretarse como la probabilidad de que la variable respuesta *y* adquiera el valor 1 dado los predictores $x_{1,\ldots.} x_p$

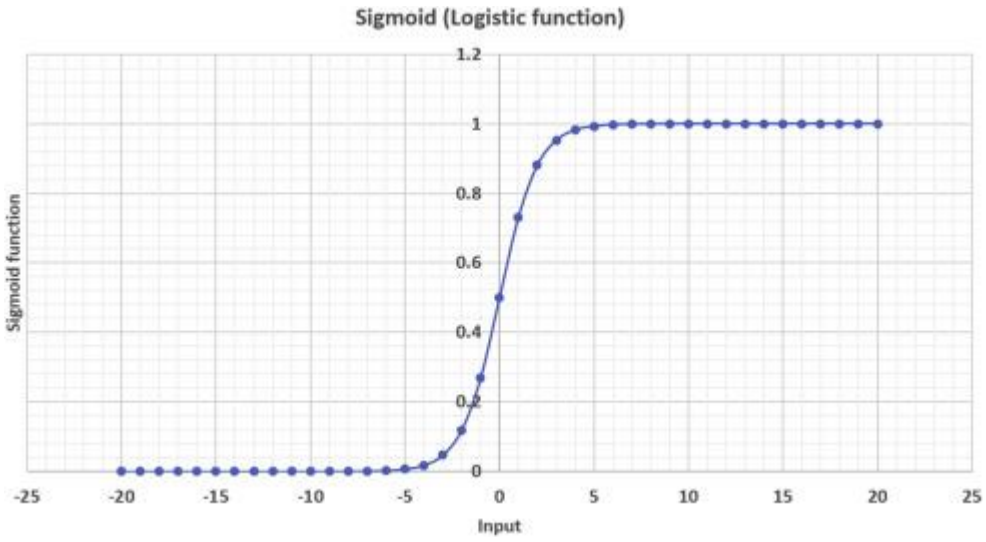


Imagen 44 - Función sigmoide con valores y entre 0 y 1

Algunas condiciones para la Regresión Logística son:

- No colinealidad o multicolinealidad: en los modelos de regresión logística múltiple, los predictores deben ser independientes, no debe de haber colinealidad entre ellos. La colinealidad ocurre cuando una variable o predictor está linealmente relacionado con una o varias variables del modelo. Cuando se presenta esta situación se excluye uno de los predictores ya que sería redundante tener dos variables que representen lo mismo
- La variable dependiente es binaria: en nuestro caso es SI y NO, otros valores binarios pueden ser 0 y 1, true or false.
- Las variables independientes deben estar relacionadas linealmente con las probabilidades logarítmicas (*log odds*)
- No autocorrelación: Los valores de cada observación son independientes de los otros
- Tamaño de la muestra: si no se dispone de suficientes observaciones, predictores que no son realmente influyentes podrían parecerlo

El otro modelo que tomamos en consideración es el de Random Forest, los Random Forests están conformados por un gran número de árboles de decisión individuales que funcionan como un conjunto.

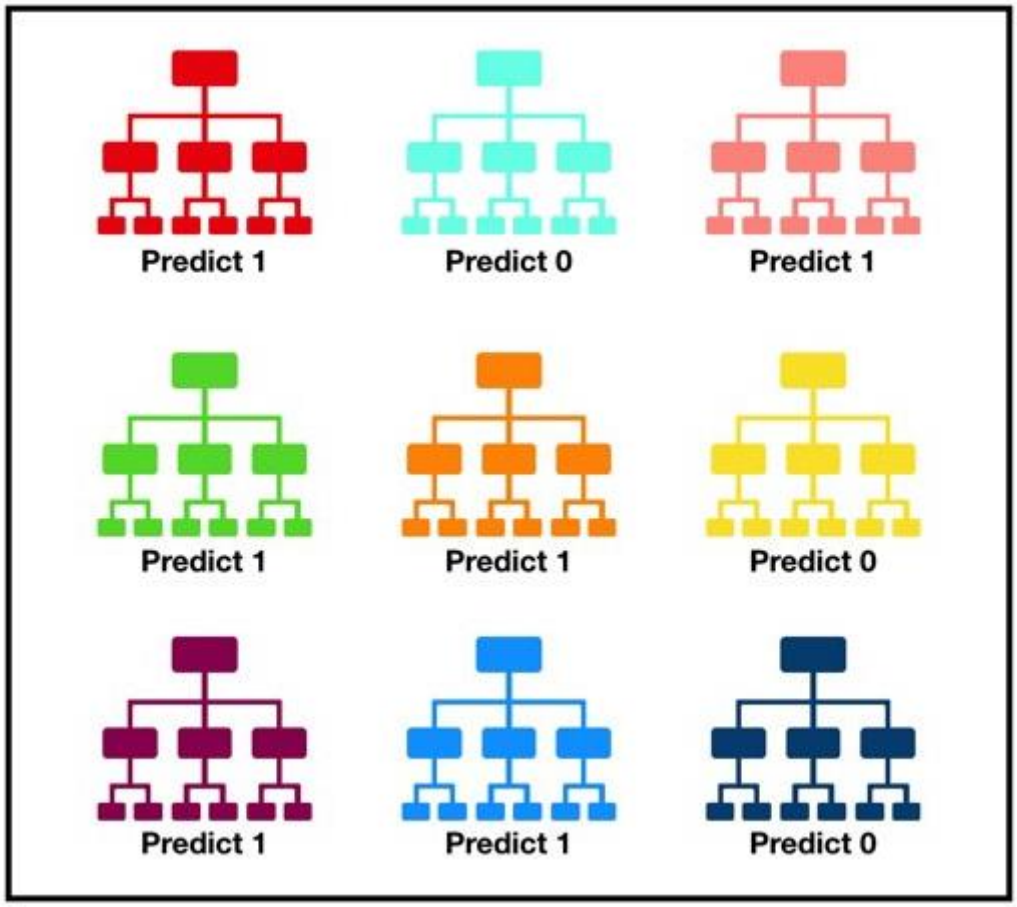


Imagen 45 - Conjunto de decision trees que funcionan como un conjunto, fuente: www.towardsdatascience.com/understanding-random-forest

Los algoritmos de random forest son una ligera modificación de bagging, ya que lo que hacen es unir muchos árboles de regresión o clasificación descorrelacionados (covarianza nula), y cuyo resultado se hace promediando los resultados de todos los árboles en caso de regresión o a votación en caso de clasificación, al mismo tiempo para obtener mejores resultados.

El algoritmo de bosque aleatorio está formado por una colección de árboles de decisión, y cada árbol del conjunto está compuesto por una muestra de datos extraída de un conjunto de entrenamiento con reemplazo, llamado muestra *bootstrap*. De esa muestra de entrenamiento, un tercio se reserva como datos de prueba, conocidos como *out-of-bag* (*oob*). A continuación, se inyecta otra instancia de aleatoriedad mediante el *bagging* de características, añadiendo más diversidad al conjunto de datos y reduciendo la correlación entre los árboles de decisión. Dependiendo del tipo de problema, la determinación de la predicción variará. Para una tarea de regresión, se promediaron los árboles de decisión individuales, y para una tarea de clasificación, un voto mayoritario -es decir, la variable

categórica más frecuente- dará como resultado la clase predicha. Por último, la muestra *oob* se utiliza para la validación cruzada, finalizando esa predicción. *(IBM Cloud Education, 2020)*

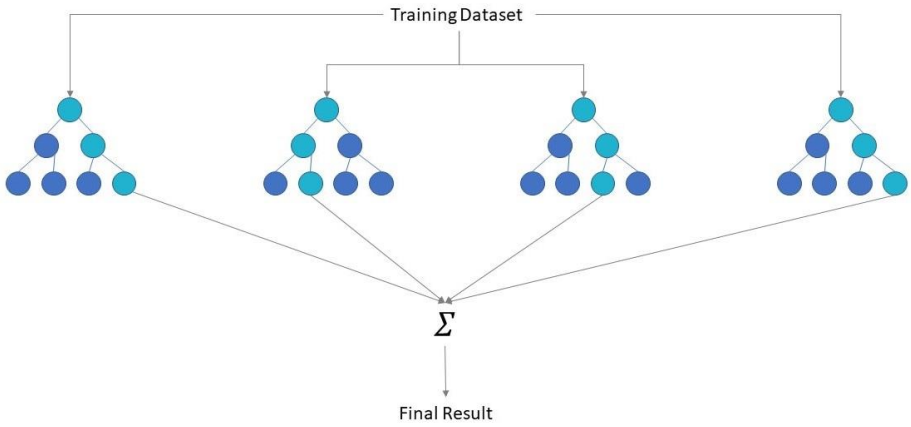


Imagen 46 - muestra final del conjunto de decision trees fuente: www.ibm.com/cloud/learn/random-forest

Entre las ventajas de Random Forest se tienen:

- Reduce el overfitting: los árboles de decisión corren el riesgo de sobreajustarse, ya que tienden a ajustarse a todas las muestras de los datos de entrenamiento
- Proveen flexibilidad: dado que Random Forest puede manejar tanto las tareas de regresión como de clasificación con un alto grado de precisión, es un método popular entre los científicos de datos.
- Útil para el feature importance: Random Forest facilita la evaluación de la importancia de las variables, o su contribución al modelo. Hay pocas formas de evaluar la importancia de las características. www.ibm.com

Donde para cada $b = 1\ to\ B$:

a. Dibujar una muestra bootstrap Z^* de tamaño *N* en el set de training.
b. Hacer crecer un random forest aleatorio con los datos obtenidos por bootstrap, repitiendo recursivamente los siguientes pasos para cada nodo terminal del árbol, hasta el tamaño mínimo del nodo $\mathrm{n}_{\mathrm{min}}$ sea alcanzado.
    - seleccionar m variables aleatorias de *p* variables.
    - elegir la mejor variable/punto de corte entre las *m*
    - dividir el nodo en dos nodos hijos

Salida del conjunto de árboles $\{T_b\}_1^B$

hacer una predicción en un nuevo punto *x:*

Regresion: $\hat{f}_{rf}^{B}(X) = \frac{1}{B}\sum_{b=1}^{B} Tb(x)$

Clasificación: Let $\hat{C}_b(X)$ sea la predicción de clase del b-ésimo random forest aleatorio.

Entonces $\hat{C}_{rf}^{B}(X) = voto\ mayoría\ \{\hat{C}_b(X)\}_1^B$

(T. Hastie, 2009)

Se necesita calcular las correlaciones de los dataframes para lo cual se recurre al metodo `corr()` de `Pandas` de forma *Pearson* el cual mide la relación estadística entre dos variables continuas cuyo rango va de 1 a -1.

```
corr_matrix = df_rl.corr(method='pearson')
```

Representado como heatmap:

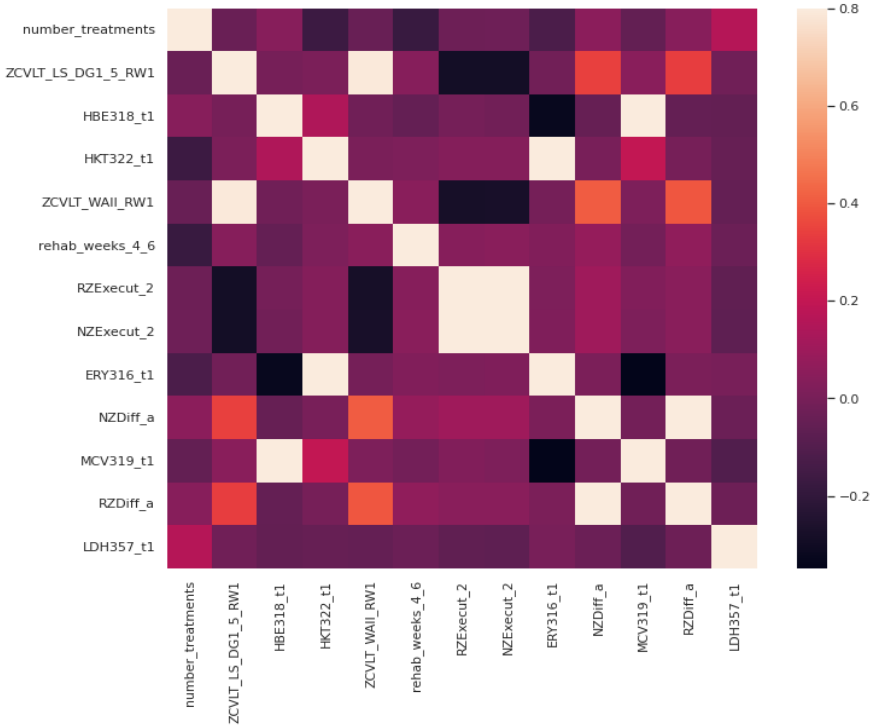


Imagen 47 - Heatmap y correlaciones del dataframe df_rl

Hay variables las cuales están totalmente correlacionadas, una de esas variables no serán incluidas en el modelo por las mismas asunciones de la Regresión Logística y por redundancia, en general al ver el color del heatmap parece estar bastante correlacionado.

Respecto a las correlaciones del target `number_treatments`

```
print(corrmat['number_treatments'].sort_values(ascending=False)[:20], '\n') #

print('----------------------')

print(corrmat['number_treatments'].sort_values(ascending=False)[-5:]) #
```

```
number_treatments     1.000000
LDH357_t1             0.164224
NZDiff_a              0.050566
HBE318_t1             0.044681
RZDiff_a              0.041752
NZExecut_2           -0.023079
RZExecut_2           -0.030569
ZCVLT_LS_DG1_5_RW1   -0.036103
ZCVLT_WAII_RW1       -0.044348
MCV319_t1            -0.056124
ERY316_t1            -0.124126
HKT322_t1            -0.166160
rehab_weeks_4_6      -0.179026
Name: number_treatments, dtype: float64

----------------------
ZCVLT_WAII_RW1    -0.044348
MCV319_t1         -0.056124
ERY316_t1         -0.124126
HKT322_t1         -0.166160
rehab_weeks_4_6   -0.179026
Name: number_treatments, dtype: float64
```

Imagen 48 - Correlaciones más altas y bajas de number_treatments del dataframe df_rl

El mismo método para el dataframe df_mr,

```
corr_matrix = df_mr.corr(method='pearson')
```

Igual que en el otro dataframe se presentan correlación muy alta entre variables, dichas variables se excluyen del modelo

```
print(corrmat['number_treatments'].sort_values(ascending=False)[:20], '\n') #

print ('----------------------')

print(corrmat['number_treatments'].sort_values(ascending=False)[-5:]) #
```

```
number_treatments    1.000000
MCC320_t1             0.171685
LDH357_t1             0.164224
FT4402_t1            -0.089166
RZExcecu             -0.089378
spg419_t1            -0.096183
height_m             -0.116016
ERY316_t1            -0.124126
RZVerbal             -0.124147
NZVerbal_2           -0.125746
RZVerbal_2           -0.126706
HKT322_t1            -0.166160
rehab_weeks_4_6      -0.179026
Name: number_treatments, dtype: float64

----------------------
RZVerbal          -0.124147
NZVerbal_2        -0.125746
RZVerbal_2        -0.126706
HKT322_t1         -0.166160
rehab_weeks_4_6   -0.179026
Name: number_treatments, dtype: float64
```

Imagen 49 - Correlaciones más altas y bajas de number_treatments del dataframe df_mr

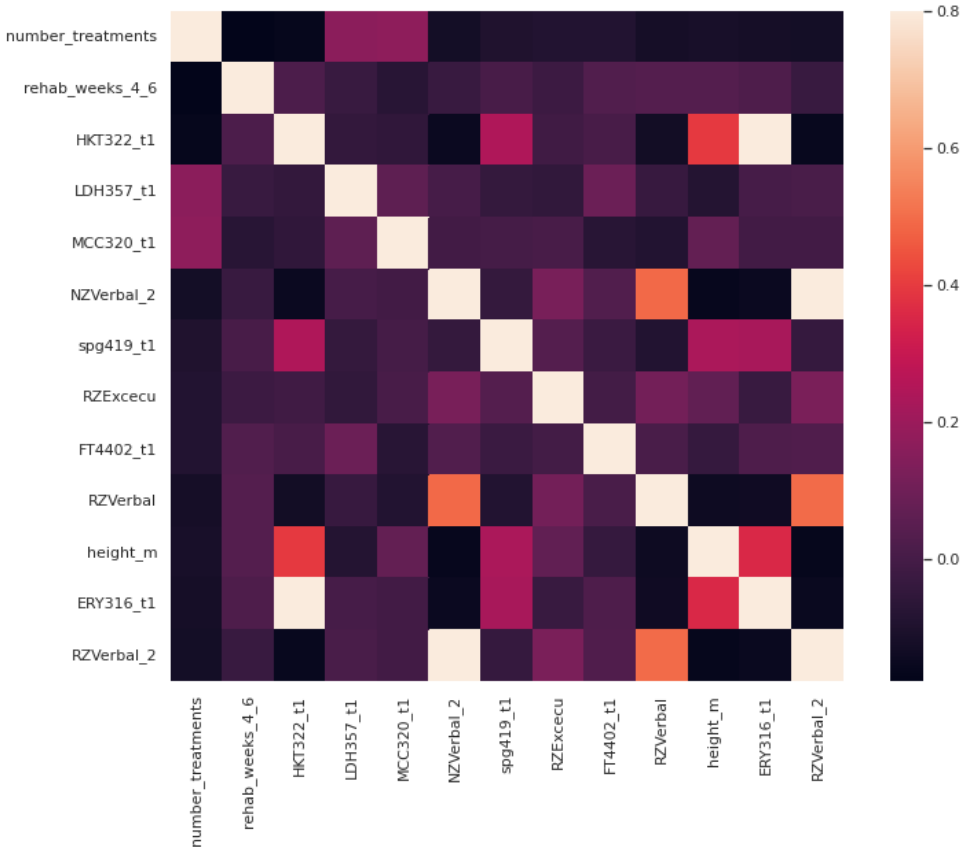


Imagen 50 - heatmap y correlaciones del dataframe df_mr

En este caso en general no hay tanta correlación entre las variables los valores en el heatmap son más bajos y el target `number_treatments` no está correlacionado con ninguna de las variables.

El mismo método para el dataframe `df_rf`

```
corr_matrix = df_rf.corr(method='pearson')

print(corrmat['number_treatments'].sort_values(ascending=Fals
e)[:20], '\n') #
```

Se presentan algunas correlaciones importantes

```
print ('----------------------')

print(corrmat['number_treatments'].sort_values(ascending=Fals
e)[-5:]) #
```

```
number_treatments    1.000000
MCC320_t1            0.171685
LDH357_t1            0.164224
age                  0.072333
THR323_t1            0.027297
CK355_t1             0.005410
BMI_T1               0.003400
GOT358_t1           -0.019984
TSH400_t1           -0.028233
FT3401_t1           -0.071315
FT4402_t1           -0.089166
ERY316_t1           -0.124126
HKT322_t1           -0.166160
Name: number_treatments, dtype: float64

----------------------
TSH400_t1   -0.028233
FT3401_t1   -0.071315
FT4402_t1   -0.089166
ERY316_t1   -0.124126
HKT322_t1   -0.166160
Name: number_treatments, dtype: float64
```

Imagen 51 - Correlaciones más altas y bajas de number_treatments del dataframe df_rf

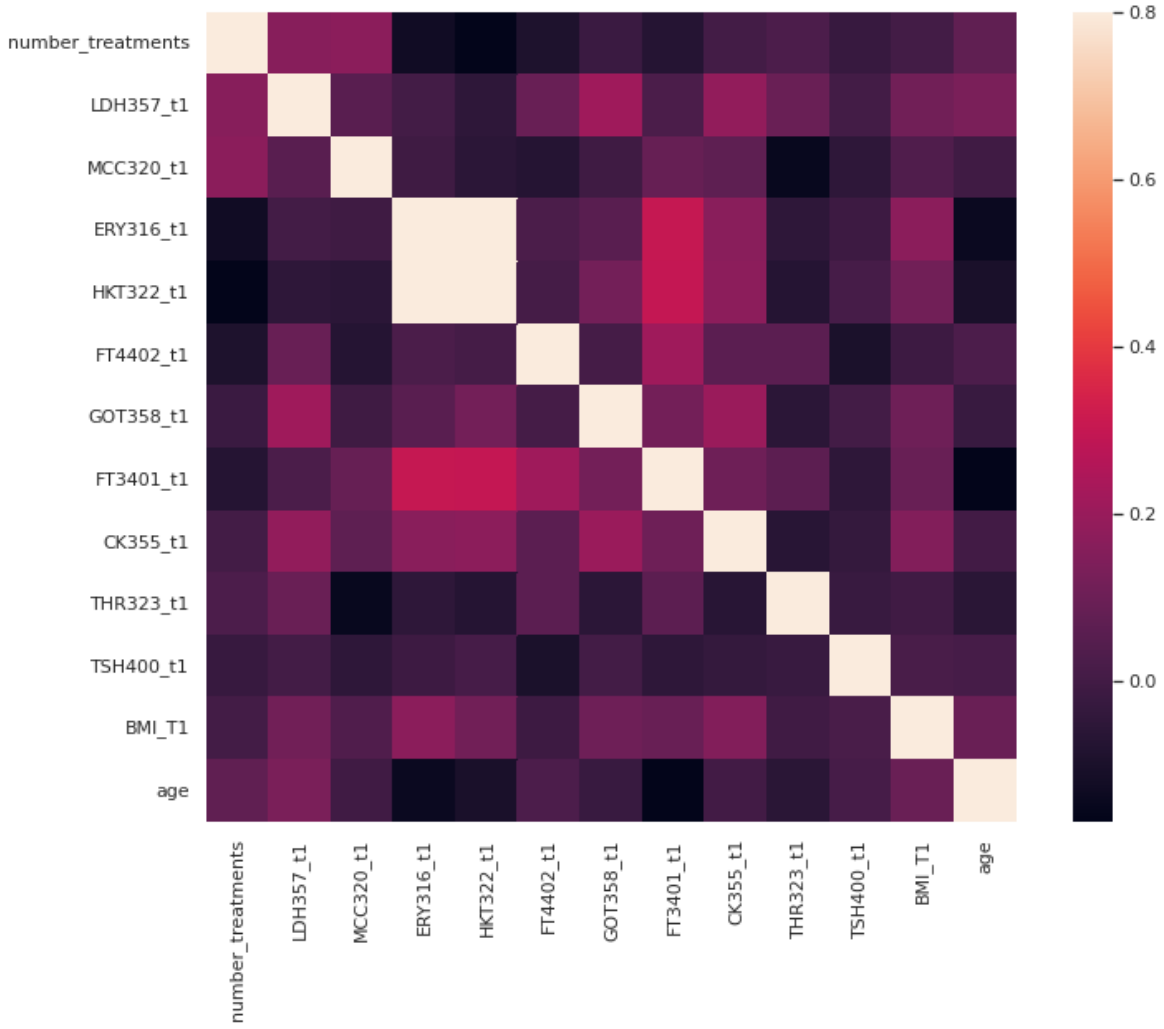


Imagen 52 - Heatmap y correlaciones del dataframe df_rf

En este caso en general no hay tanta correlación entre las variables los los valores en el heatmap son bajos quizás un poco más alto que en el modelo df_mr, el target `number_treatments` no está correlacionado con ninguna de las variables.

Los modelos de regresión logística no se ven muy afectados por outliers, ya que la función sigmoide reduce los valores atípicos. Los valores erróneos o extremos que tenía el dataframe tale scomo 9999, 999 ya fueron eliminados previamente.

Mediante un boxplot del dataframe `df_rl` se tienen estos outliers

```
df_rl.boxplot(column=['number_treatments',
'ZCVLT_LS_DG1_5_RW1','HBE318_t1','HKT322_t1','ZCVLT_WAII_RW1'
, 'rehab_weeks_4_6', 'RZExecut_2', 'NZExecut_2','ERY316_t1',
'NZDiff_a', 'MCV319_t1', 'RZDiff_a', 'LDH357_t1'], grid=True,
color='black', rot=90, fontsize=15, figsize=(15,12) )
```

Se utiliza el método `boxplot()` de `pandas` se pueden apreciar datos importantes, ejemplo las variables `RZExecut_2` y `RZDiff_a` son simétricas puesto la mediana está en el centro del rango intercuartílico Q1 - Q3.

Algunas variables por ejemplo `NZDiff_a`, `ERY316_t1` el rango intercuartílico, es bastante reducido puesto estas variables tienen un rango de valores muy pequeños y esta

En la siguiente imagen del diagrama de distribución se aprecia cómo se distribuye la variable `NZDiff_a` la mayoria de datos están entre -1 y +1 por lo que son cifras muy pequeñas.

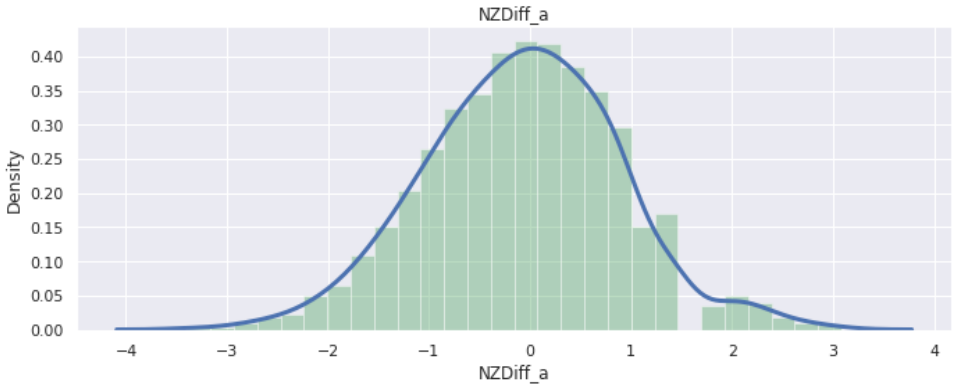


Imagen 53 - Distplot de la variable NZDiff_a del dataframe df_rl

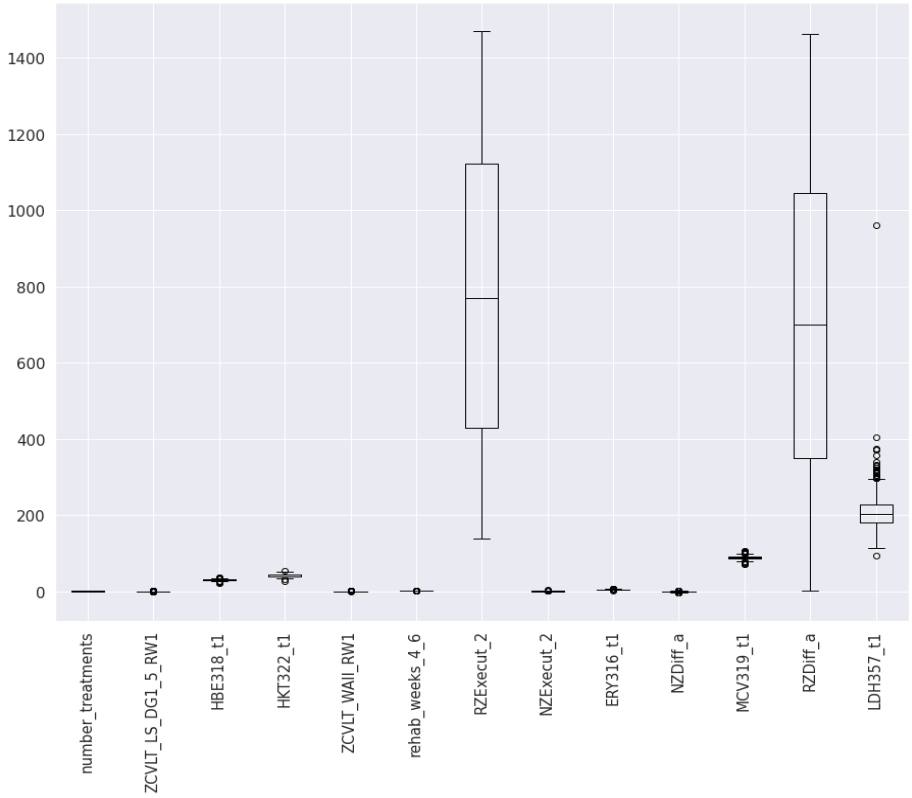


Imagen 54 - Diagrama boxplot con rangos intercuartílicos dataframe df_rl

En el caso del dataframe df_mr

```
df_mr.boxplot(column=['number_treatments', 'rehab_weeks_4_6',
'HKT322_t1', 'LDH357_t1','MCC320_t1', 'NZVerbal_2',
'spg419_t1', 'RZExcecu', 'FT4402_t1','RZVerbal', 'height_m',
'ERY316_t1', 'RZVerbal_2'], grid=True, color='black', rot=90,
fontsize=15, figsize=(15,12))
```

Las variables `RZVerbal`, `RZVerbal_2`, `RZExcecu` no presentan outliers se puede ver simetría en estas variables puesto la mediana está en el centro del rango

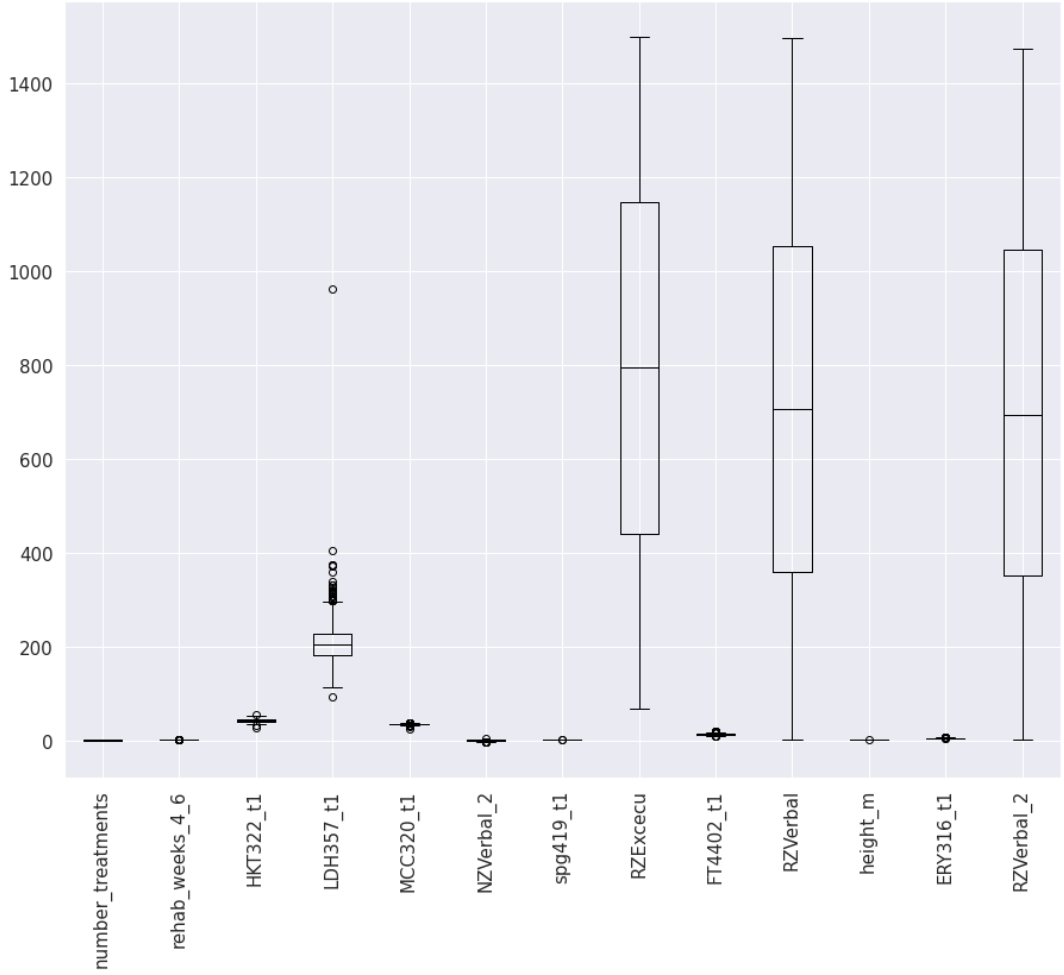


Imagen 55 - Diagrama boxplot con rangos intercuartílicos dataframe df_mr

Continuando con los Boxplots del dataframe df_rf

```
df_rf.boxplot(column=['number_treatments', 'LDH357_t1',
'MCC320_t1', 'ERY316_t1', 'HKT322_t1','FT4402_t1',
'GOT358_t1', 'FT3401_t1', 'CK355_t1','THR323_t1','TSH400_t1',
'BMI_T1', 'age'], grid=True, color='black', rot=90,
fontsize=15, figsize=(15,12) )
```

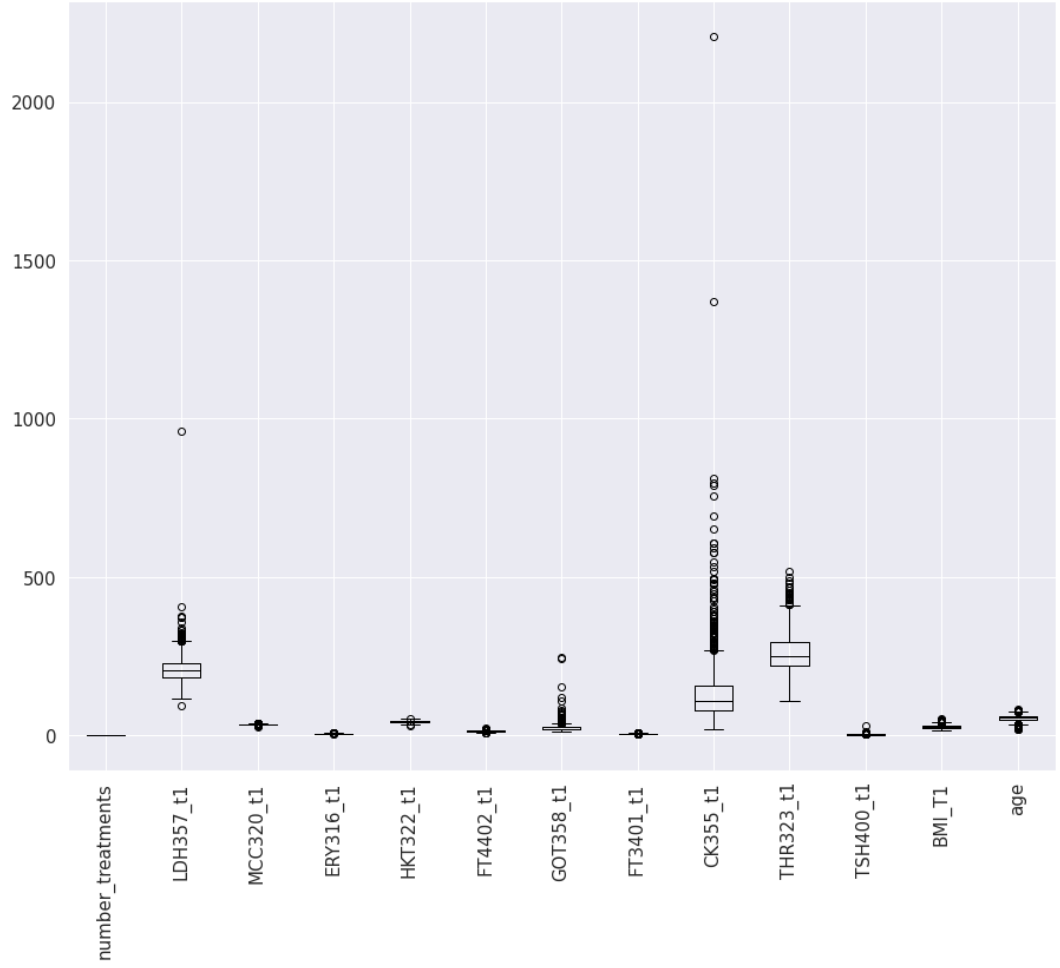


Imagen 56 - Diagrama boxplot con rangos intercuartílicos dataframe df_rf

En general el algoritmo de Random Forest es resistente a outliers ya que se promedian mediante la agregación de los resultados de varios árboles, los métodos de árbol son métodos "contributivos", en los que sólo los puntos locales -los del mismo nodo de la hoja- afectan a la estimación de un punto determinado. Por tanto, los valores atípicos de salida tienen un efecto de "cuarentena". Por lo tanto, los outliers que distorsionan la precisión de algunos algoritmos tienen un efecto menor en la predicción de un Random Forest.

En la siguiente imagen se pueden ver los histogramas de cada variable junto al target `number_treatments` del dataframe `df_rl`, en general los valores 0 es decir aquellos pacientes que no ingresaron a tratamiento tienen los valores más altos respecto a las variables.

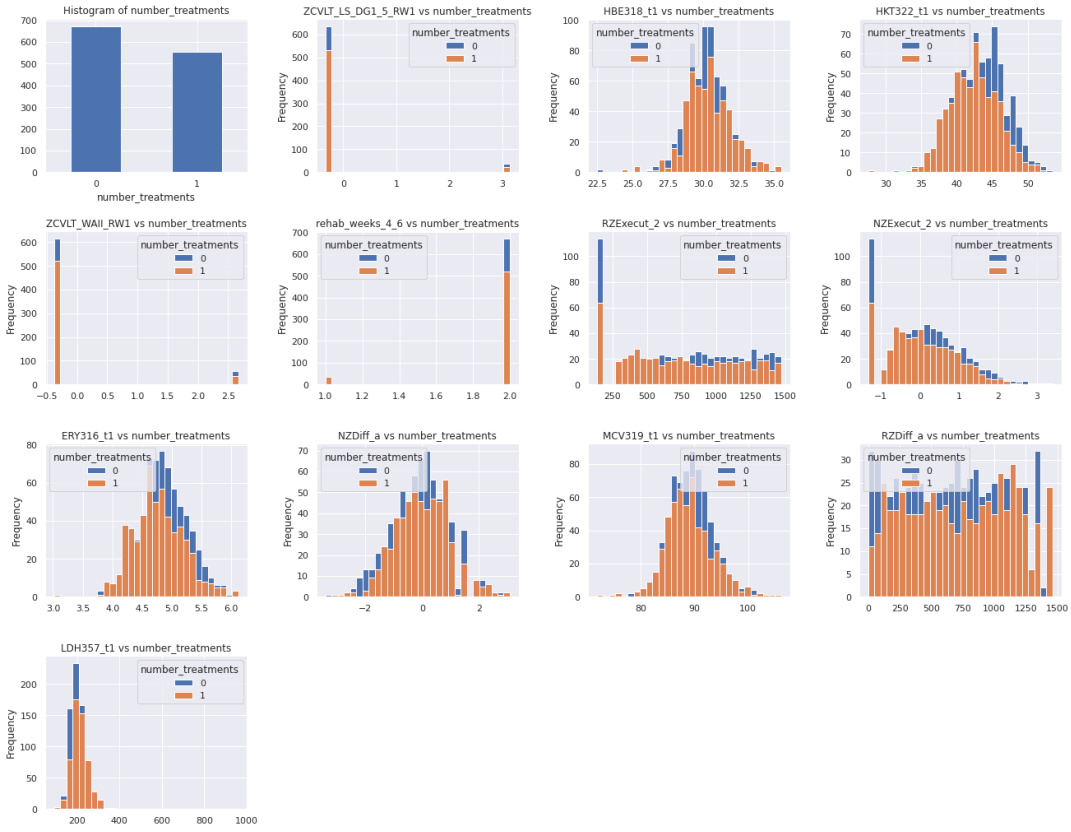


Imagen 57 -Histogramas del dataframe df_rl comparados con el target valores 0 y 1

En el caso del dataframe `df_mr` la tendencia es similar los valores 0, son los más altos con respecto a las otras variables, en primer lugar, se aprecia que el target ya está balanceado cuando al principio del ejercicio no tenía un balance adecuado.

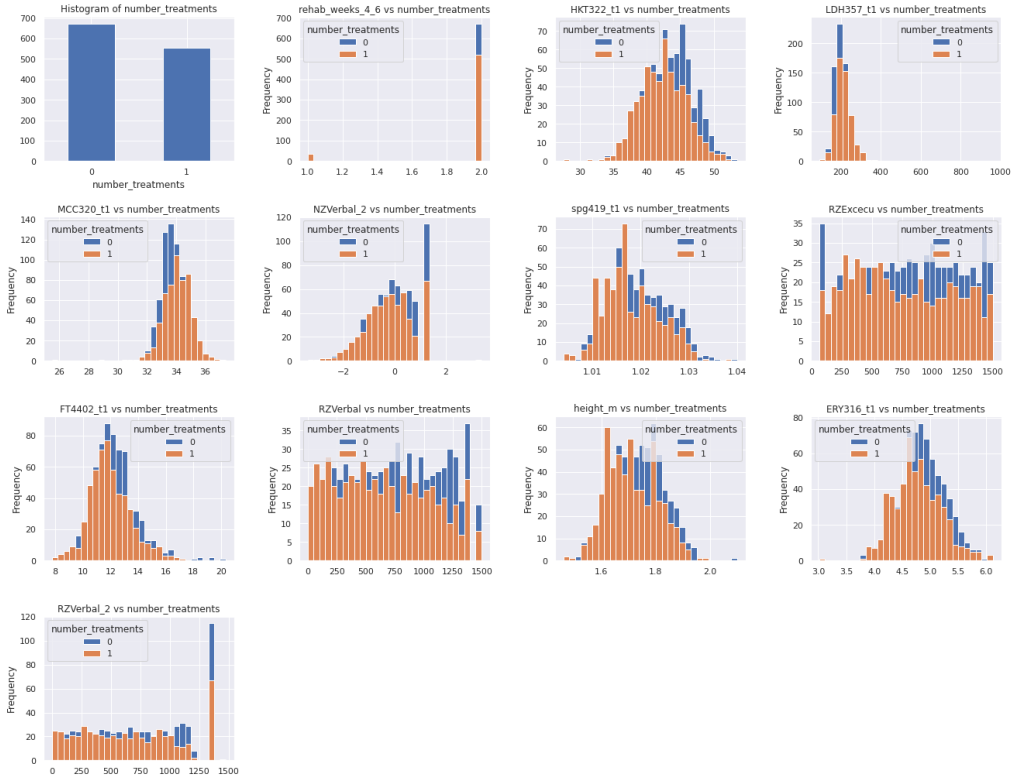


Imagen 58 - Histogramas del dataframe df_mr comparados con el target valores 0 y 1

En el caso del dataframe `df_rf` la tendencia es similar los valores 0, son los más altos con respecto a las otras variables, en primer lugar, se aprecia que el target ya está balanceado cuando al principio del ejercicio no tenía un balance adecuado.

En el caso de la edad por ejemplo el volumen más alto está entre los 43 y 60 años aproximadamente, el BMI por ejemplo no es determinantes para ingresar un paciente a tratamiento la clase dominante es 0, algunos casos donde el BMI tiene valor en 33, las clases SI y No son casi iguales.

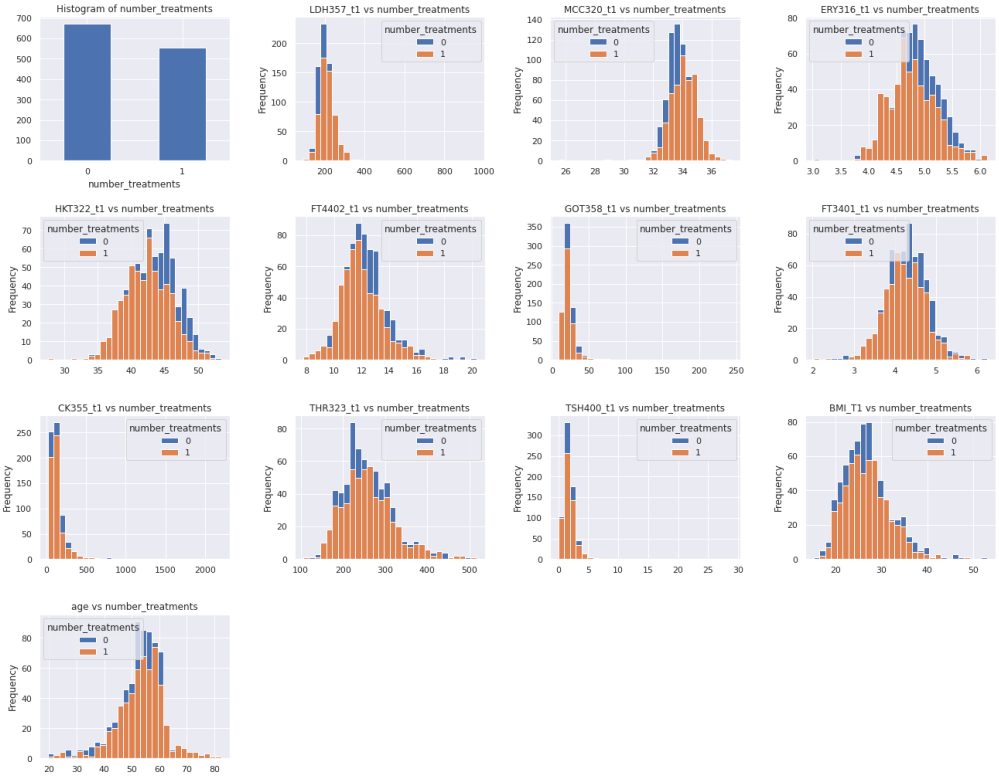


Imagen 59 - Histogramas del dataframe df_rf comparados con el target valores 0 y 1

Finalmente, el modelo a utilizar es Regresión Logística ya previamente se hizo una introducción teórica del mismo, debido a que estamos clasificando entre 0 y 1, tenemos un modelo de clasificación binaria, en Machine Learning el término Clasificación corresponde a el proceso de reconocer, entender y agrupar categorías, en este tipo de algoritmos los datos están categorizados o etiquetados según el problema a resolver.

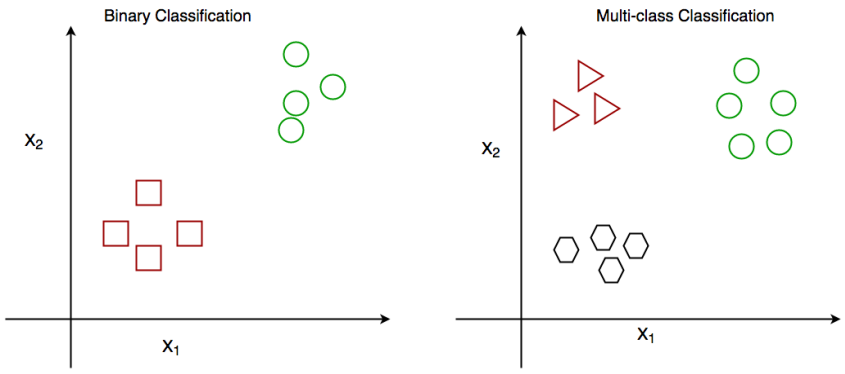


Imagen 60 - Ejemplos de clasificación binaria y multiclase

Los modelos de Clasificación a su vez hacen parte del Supervised Machine Learning, la mayoría de los algoritmos de Machine Learning son de tipo supervisados esto quiere decir que se tienen variables de entrada *X* y una variable de salida *Y*, por lo tanto, estas técnicas se usan para aprender desde la entrada a la salida *Y=f(X)*. Mediante el conjunto de datos de entrada se debe predecir la salida *Y*. Las técnicas de los algoritmos de aprendizaje automático supervisado incluyen la regresión lineal y logística, la clasificación multiclase, Decision Trees y support vector machines (SVM). El aprendizaje supervisado requiere que los datos utilizados para entrenar el algoritmo ya están etiquetados en este caso es nuestro dataset el cual ya viene etiquetado con los datos correctos, aquellos datos erróneos ya fueron retirados.

Las ventajas de utilizar Regresión Logística son:

- Funciona bien cuando el conjunto de datos es linealmente separable
- Menos propensos a over-fitting
- Da una medida de la relevancia de un predictor (tamaño del coeficiente) y la dirección de la asociación (positiva o negativa)
- Fácil de aplicar, interpretar y muy eficiente para formar

La implementación se hace con el paquete de Logistic Regression de Sciki-learn

Dataframe df_rl:

Lo primero es convertir el target variable a category y cambiar los 0 a 'NO', 1 a 'SI' en el dataframe `df_rl`. Procedimiento que se hace con `replace()` y `astype()`

```
df_rl['number_treatments'] =
df_rl['number_treatments'].replace([0], 'NO')
```

```
df_rl['number_treatments'] =
df_rl['number_treatments'].replace([1], 'YES')

df_rl.number_treatments =
df_rl.number_treatments.astype('category')

df_rl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1227 entries, 0 to 1518
Data columns (total 13 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   number_treatments   1227 non-null   category
 1   ZCVLT_LS_DG1_5_RW1  1227 non-null   float32
 2   HBE318_t1           1227 non-null   float32
 3   HKT322_t1           1227 non-null   float32
 4   ZCVLT_WAII_RW1      1227 non-null   float32
 5   rehab_weeks_4_6     1227 non-null   int32
 6   RZExecut_2          1227 non-null   int32
 7   NZExecut_2          1227 non-null   float32
 8   ERY316_t1           1227 non-null   float32
 9   NZDiff_a            1227 non-null   float32
 10  MCV319_t1           1227 non-null   float32
 11  RZDiff_a            1227 non-null   float32
 12  LDH357_t1           1227 non-null   int32
dtypes: category(1), float32(9), int32(3)
memory usage: 100.7 KB
```

Imagen 61 - Data types del dataframe df_rl

Estas variables están muy correlacionadas no las incluyo en el modelo ya se comprobó en el heatmap hecho previamente

```
#ZCVLT_WAII_RW1, MCV319_t1, ERY316_t1, NZExecut_2, RZDiff_a#
```

El modelo se lanza inicialmente con siete variables que se asigna a *X*, el target variable `number_treatments` se asigna a *y*

```
X = df_rl[['ZCVLT_LS_DG1_5_RW1', 'HBE318_t1', 'HKT322_t1',

'rehab_weeks_4_6', 'RZExecut_2','NZDiff_a', 'LDH357_t1']]

y = df_rl['number_treatments']
```

Como todo algoritmo de lenguaje supervisado requiere una parte de entrenamiento y otra parte de test, con el training set se va a verificar el performance con datos nuevos. Lo que se

hace es dividir en partes por lo general se usa 80% para training y 20% para test otros estudios lo hacen con 70 - 30, nuestro ejercicio se hace 80 - 20. El split de los datos se hace con `train_test_split()` de Scikit-learn se toma un 80% de muestras aleatorias sin reemplazo y el 20% restante es para el test el cual es usado para evaluar el fit hecho en el training muy importante que el test evalúe el performance con datos nuevos o sea aquellos que no fueron usados para entrenar el modelo.

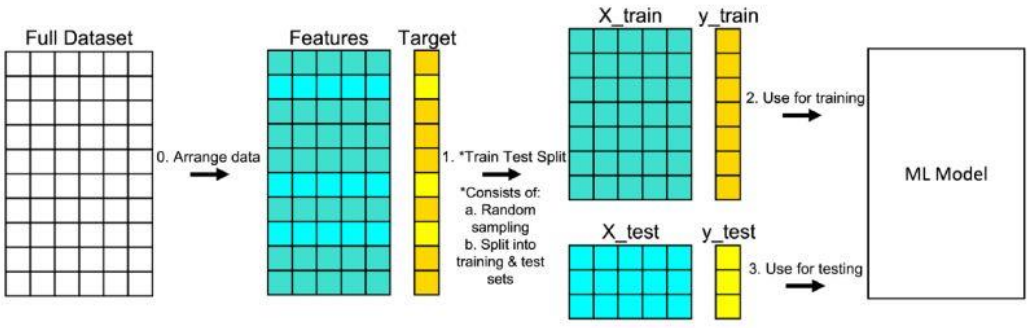


Imagen 62 - Procedimiento train test split Fuente www.builtin.com/data-science/train-test-split

```
## Divide the data into training and test sets --------------
-------------------------------------
```

Se asigna un `random_state = 5` para que sea reproducible y no cambien los datos cada que se ejecute el código. `stratify = y` se usa para que el método `train_test_split()` devuelve subconjuntos de entrenamiento y de prueba que tienen las mismas proporciones de etiquetas de clase que el conjunto de datos de entrada.

```
## Create random 80/20 % split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 5, stratify = y)
```

En este punto se asignan a otros elementos para ser utilizados más adelante

```
## Create dataset to store model predictions

dfTR_eval = X_train.copy()

dfTR_eval['Y'] = y_train

dfTS_eval = X_test.copy()
```

```
dfTS_eval['Y'] = y_test
```

El training del modelo empieza con la función `pipeline()` el cual permite crear un flujo de trabajo en orden y hace el código más ordenado.

Lo primero es normalizar los datos con el `StandardScaler()` también de `scikit-learn` el cual realiza la estandarización de datos es decir mean = 0, standard deviation = 1, el centrado y el escalado se realizan de forma independiente en cada característica, calculando los estadísticos pertinentes en las muestras del conjunto de entrenamiento La estandarización de un conjunto de datos es un requisito común para muchos estimadores de aprendizaje automático: podrían comportarse mal si las características individuales no se parecen más o menos a los datos estándar distribuidos normalmente.

En el pipeline se incluye el modelo, la función de `LogisticRegression` con el random_state = 5 cómo se hizo previamente también.

Para validar el performance del modelo se utiliza el metodo estadistico de Cross Validation el cual es un método de resampling que utiliza diferentes partes del dataset para testear y entrenar el modelo en diferentes iteraciones.

Este enfoque consiste en dividir aleatoriamente el conjunto de observaciones en k grupos, o pliegues, de tamaño aproximadamente igual. El primer pliegue se trata como un conjunto de validación, y el método se ajusta a los restantes $k - 1$ folds. A continuación, se calcula el error cuadrático medio, MSE, en las observaciones del pliegue retenido. Este procedimiento se repite *k* veces; cada vez, un grupo diferente de observaciones es tratado como un conjunto de validación. Este proceso da lugar a *k* estimaciones del error de la prueba, MSE1, MSE2, . . . , MSE*k*. El CV *k*-fold se calcula promediando estos valores. *(Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirami, 2013)*

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i$$

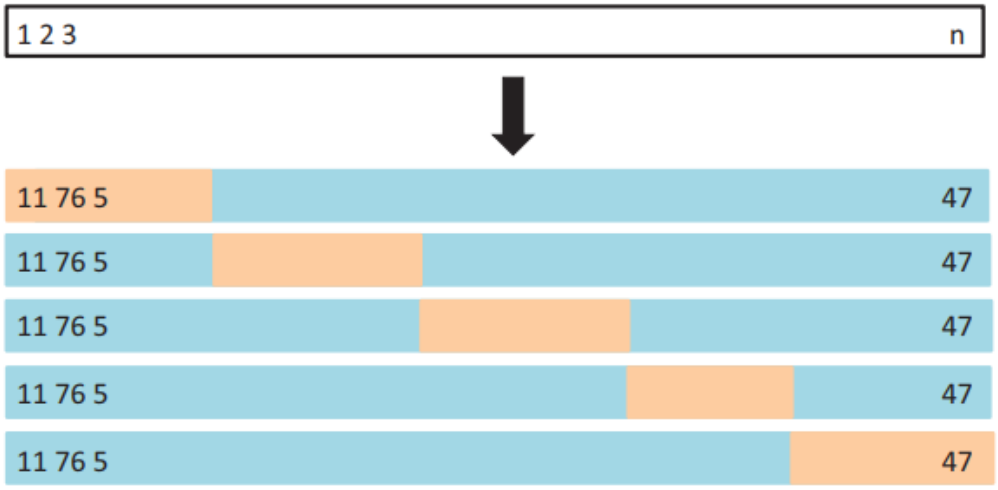


Imagen - una muestra CV 5-folds Un conjunto de n observaciones se divide aleatoriamente en cinco grupos que no se solapan. Cada uno de estos quintos actúa como conjunto de validación (color beige), y el resto como conjunto de entrenamiento (color azul). El error de prueba se calcula promediando las cinco estimaciones de MSE resultantes. Fuente: (Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirami, 2013)

El valor de k deber ser propiamente elegido normalmente se utiliza *k*=10 suele dar un sesgo bajo y varianza regular, en otros modelos de resampling como LOOCV se utliza *k*=5

Continuando con la validación del modelo se utiliza la función `GridSearchCV` también de Scikit-learn la cual permite evaluar y seleccionar de forma sistemática los parámetros de un modelo.

La función `GridSearchCV` se guardan en `LogReg_fit` con los siguientes parámetros

- `param_grid:` dict or list of dictionaries this enables searching over any sequence of parameter settings, the empty dictionary is defined `param`
- `n_jobs:` number of jobs to run in parallel -1 means using all processors
- `scoring:` strategy to evaluate the performance of the cross-validated model on the test set, Accuracy option is used because this is a classification model
- `cv:` number of folds defined in nFolds = 10

```
## Train model ----------------------------------------

pipe = Pipeline(steps=[('scaler',StandardScaler()), #
Preprocess the variables when training the model
```

```
('LogReg',LogisticRegression(random_state=5))]) # Model to use
in the pipeline

# Search Cross Validation to find the best parameter for the model in
the grid defined

nFolds = 10

param = {}

LogReg_fit = GridSearchCV(estimator=pipe, # Structure of the
model to use

             param_grid=param, # Defined grid to search in

             n_jobs=-1, # Number of cores to use (parallelize)

             scoring='accuracy', # Accuracy

             cv=nFolds) # Number of Folds

LogReg_fit.fit(X_train, y_train) # Train both the scaler and the
model

scores = cross_val_score(LogReg_fit, X_train, y_train, cv =
10) # Cross validation scores

for i in range(0,len(scores)):

     print("Resample %i: Accuracy - %0.2f" % (i+1, scores[i])
     )

print("Mean accuracy - %0.2f (+/- %0.2f)" % (scores.mean(),
scores.std() * 2))

ax = sns.boxplot(y = scores).set_title('Boxplot for summary
metrics of test samples')

plt.xlabel('Accuracy')

plt.show()
```

Con el método fit se entrena el scaler y el modelo finalmente en el ciclo `for` se muestran las accuracies de cada fold en diagrama `boxplot`,las accuracies del modelo tienen poca diferencia entre ellas lo que indica una robustez en el modelo, esto indica que el modelo es robusto, predice de forma correcta sin sobreentrenamiento, la media es de 64% de accuracy.

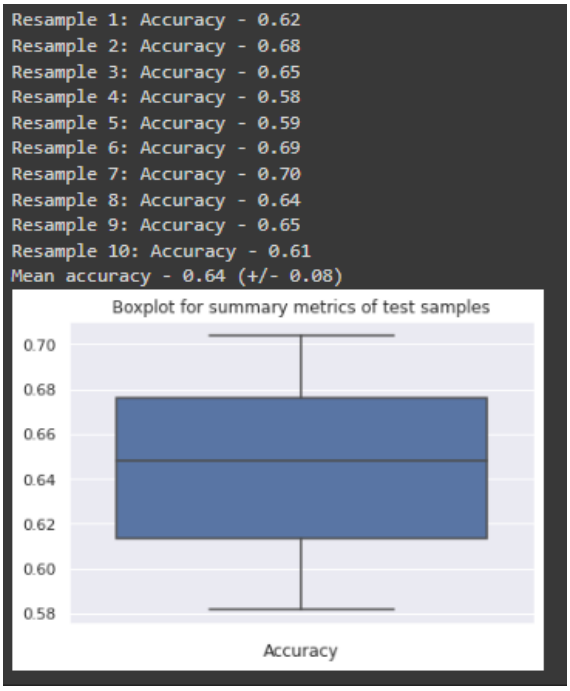


Imagen 63 - Boxplot de accuracies, 10 k-folds en el cross-validation del dataframe df_rl

Los resultados del modelo y coeficientes se hacen con la función `summaryLogReg`

```
CT.summaryLogReg(LogReg_fit.best_estimator_['LogReg'],
X_train, y_train)
```

De acuerdo con los coeficientes se aprecia que las más significativas aquellas con los *pvalues* el residuo de desviación es la medida de desviación aportada por cada observación entre más bajo es mejor donde la mediana es de -5.519850e-16

```
Deviance Residuals:
   Min   1Q       Median   3Q  Max
0 -1.0 -1.0 -5.519850e-16  0.0  1.0

Coefficients:
                  Estimate  Std. Err   t-value  Pr(>|t|) Signif
Intercept         -0.158290  9.355212 -0.016920  0.986500
ZCVLT_LS_DG1_5_RW1 -0.110748  1.029355 -0.107590  0.914321
HBE318_t1          0.135422  0.238337  0.568197  0.569901
HKT322_t1         -0.396124  0.162438 -2.438625  0.014743      *
rehab_weeks_4_6   -0.605695  2.885084 -0.209940  0.833714
RZExecut_2        -0.083613  0.020049 -4.170381  0.000030    ***
NZDiff_a           0.200058  0.683359  0.292756  0.769708
LDH357_t1          0.510389  0.120704  4.228442  0.000024    ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Imagen 64 - Coeficientes y Summary del dataframe df_rl

t-value o t -statistic es la relación entre la desviación del valor estimado de un parámetro con respecto a su valor hipotético y su error estándar. La columna Pr(>|t|) representa el valor p asociado al valor de la columna del valor t.

- Un valor bajo de p-value (< 0.05 significa que es probable que el coeficiente no sea igual a cero.
- Un valor alto de p-value (> 0.05) significa que no podemos concluir que la variable explicativa afecta a la variable dependiente.

En esta iteración del modelo las variables que cumplen con la null hypothesis dónde quiere decir que hay una relación entre variable por eso el p-value <0.05, de acuerdo con esto las variables `HKT322_t1`, `RZExecut_2`, `LDH357_t1` están relacionadas.

Dataframe df_mr:

El procedimiento para el dataframe df_mr es igual al anterior

Estas variables están muy correlacionadas no las incluyo en el modelo ya se comprobó en el heatmap hecho previamente

```
#ERY316_t1, RZVerbal_2#
```

El modelo se lanza inicialmente con siete variables que se asigna a *X*, el target variable `number_treatments` se asigna a *Y*

```
X = df_mr[['rehab_weeks_4_6', 'HKT322_t1', 'LDH357_t1',
'MCC320_t1', 'NZVerbal_2', 'spg419_t1', 'RZExcecu',
'FT4402_t1','RZVerbal', 'height_m']]

y = df_mr['number_treatments']

## Divide the data into training and test sets ----------------------
------------------------------

## Create random 80/20 % split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 5, stratify = y)

## Create dataset to store model predictions

dfTR_eval = X_train.copy()

dfTR_eval['Y'] = y_train

dfTS_eval = X_test.copy()

dfTS_eval['Y'] = y_test

## Train model ------------------------------------------------
```

El mismo Pipeline con el scaler y el modelo

```
pipe = Pipeline(steps=[('scaler',StandardScaler()), #
Preprocess the variables when training the model
('LogReg',LogisticRegression(random_state=5))]) # Model to use
in the pipeline
```

Se utiliza Cross Validation para validar el modelo, los mismos parámetros anteriores

```
# Grid Search Cross Validation to find the best parameter for the model
in the grid defined

nFolds = 10

param = {}

LogReg_fit = GridSearchCV(estimator=pipe, # Structure of the
model to use

param_grid=param, # Defined grid to search in
```

```
n_jobs=-1, # Number of cores to use (parallelize)

                        scoring='accuracy', # Accuracy

                        cv=nFolds) # Number of Folds

LogReg_fit.fit(X_train, y_train) # Train both the scaler and the
model

scores = cross_val_score(LogReg_fit, X_train, y_train, cv =
10) # Cross validation scores

for i in range(0,len(scores)):

     print("Resample %i: Accuracy - %0.2f" % (i+1, scores[i])
     )

print("Mean accuracy - %0.2f (+/- %0.2f)" % (scores.mean(),
scores.std() * 2))

ax = sns.boxplot(y = scores).set_title('Boxplot for summary
metrics of test samples')

plt.xlabel('Accuracy')

plt.show()
```

El mismo caso anterior las accuracies están en un rango parecido, en este caso se alcanzaron algunos valores un poco más altos, la media es de 66% mejor que el modelo anterior.

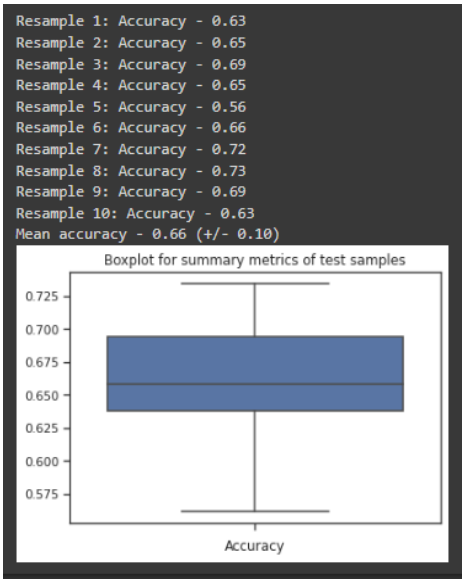


Imagen 65 - Boxplot de accuracies,10 k-folds en el cross-validation del dataframe df_mr

En este caso todos los coeficientes tuvieron p-values más altos que 0.05, lo que indica que las variables explicativas afectan a la variable dependiente.

```
Deviance Residuals:
   Min            1Q        Median    3Q  Max
0 -1.0 -7.831031e-57 -3.989721e-118  1.0  1.0

Coefficients:
                Estimate    Std. Err   t-value  Pr(>|t|) Signif
Intercept       -0.167837  380.702761 -0.000441  0.999648
rehab_weeks_4_6 -0.601881  130.327565 -0.004618  0.996315
HKT322_t1       -0.342128    0.318084 -1.075592  0.282110
LDH357_t1        0.454362    0.286349  1.586744  0.112571
MCC320_t1        0.360686    2.221299  0.162376  0.871010
NZVerbal_2      -0.246497    1.450233 -0.169971  0.865033
spg419_t1       -0.131920  254.025774 -0.000519  0.999586
RZExcecu        -0.159236    0.098428 -1.617791  0.105708
FT4402_t1       -0.209507    1.263370 -0.165832  0.868289
RZVerbal        -0.124193    0.075981 -1.634528  0.102148
height_m        -0.129332   23.718141 -0.005453  0.995649
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Imagen 66 - Coeficientes y Summary del dataframe df_mr

El otro algoritmo elegido para el modelo es el de Random Forest, el cual es ampliamente utilizado en proyectos de Clasificación y Regresión, ya se utilizó previamente en el Feature

Selection con buenos resultados construye árboles de decisión sobre diferentes muestras y toma su voto mayoritario para la clasificación y la media en caso de regresión.

Random forest es lo que se llama un método de conjunto (o *Ensemble* method, en inglés), es decir que "pone junto" o combina resultados para obtener un resultado final, es decir el resultado de los diferentes árboles de decisión que lo componen.

Es necesario tener en cuenta algunos conceptos básicos de las técnicas de *Ensemble*

- Bagging: Se construyen muchos modelos del mismo tipo, pero entrenando con diferentes submuestras del dataset. Ej: hacemos submuestras del training set, hacemos un arbol diferente con cada ese subconjunto de la muestra y combinamos
- Boosting: Se construyen modelos del mismo tipo, pero cada uno de ellos busca aprender de los errores de predicción del anterior, es decir busca aprender aquello que el anterior no pudo por lo tanto, *bagging* es aprendizaje en paralelo mientras que boosting no.

Tal como se indicó previamente Random Forest es un método Ensemble aquí detallaremos el concepto de *Bagging* en el cual se basa Random Forest

- Bagging: también conocido como *Bootstrap Aggregation* es la técnica de ensamblaje utilizada por Random Forest. Bagging elige una muestra aleatoria del conjunto de datos. Por lo tanto, cada modelo se genera a partir de las muestras (muestras *Bootstrap*) proporcionadas por los datos originales con reemplazo conocido como *row sampling*. Este paso de muestreo de filas con reemplazo se llama *bootstrap*. Ahora cada modelo se entrena de forma independiente, lo que genera resultados. El resultado final se basa en la votación por mayoría tras combinar los resultados de todos los modelos. Este paso, que implica la combinación de todos los resultados y la generación de resultados basados en la votación por mayoría, se conoce como *aggregation*.

  Este es un procedimiento de propósito general para reducir la varianza de un método de aprendizaje estadístico, una forma de reducir la varianza y por tanto, de aumentar la precisión de las predicciones de un método de ML, es tomar muchos conjuntos de entrenamiento (TR) (con bootstrap) de la población, construir un modelo independiente utilizando cada TR y promediar las predicciones resultantes.

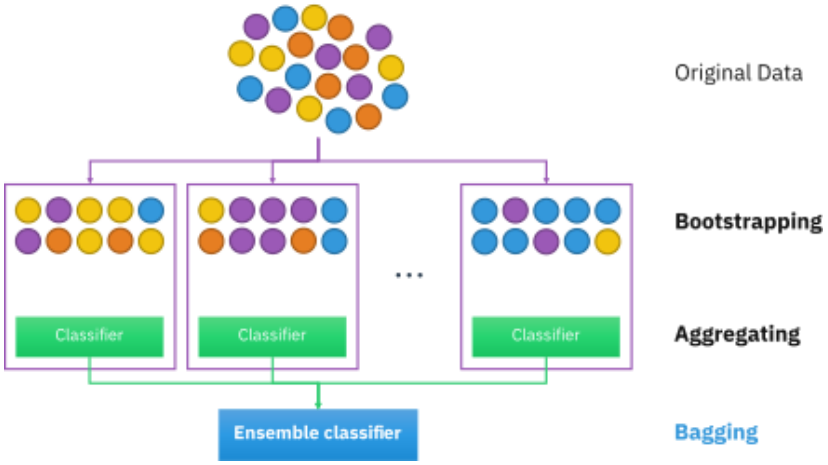


Imagen - el resultado final es el bagging o ensemble previa selección de muestras Bootstrap, fuente: /www.analyticsvidhya.com

Bootstrap step (se crean muestras bootstrap con reemplazo)

$$Z = (Z_1, Z_2, \ldots, Z_N)$$

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$Z^{*b}, \; b = 1, \ldots, B$$

Fitting step (se utilizan muestras de entrenamiento bootstrap**)**

$$Z^{*b} \rightarrow \hat{f}^{*b}(x) \quad b = 1, \ldots, B$$

Aggregation step (promediar los modelos o por votación de mayoría para clasificación)

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$$

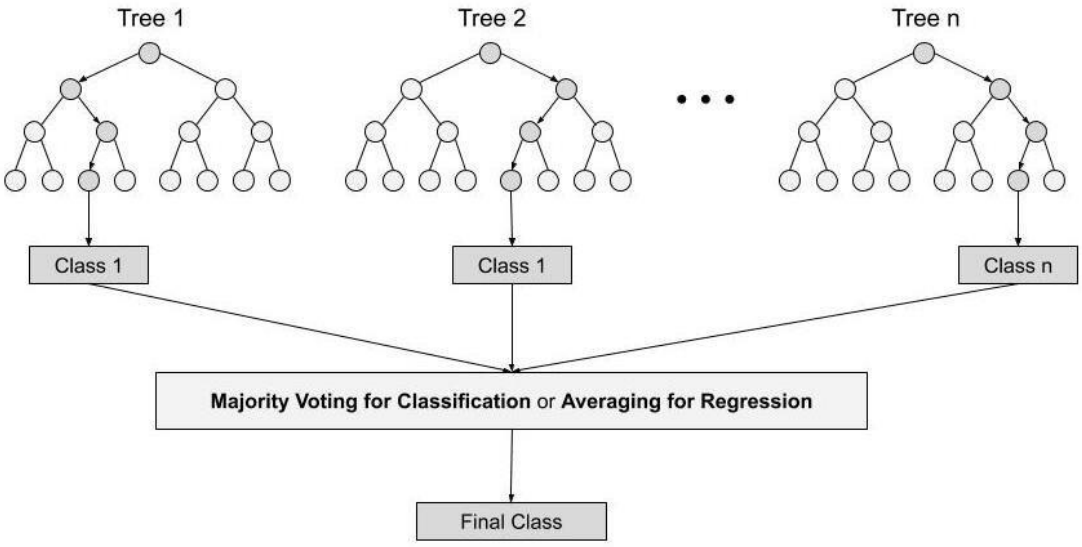


Imagen - el resultado final es por votación en clasificación o por promedio para regresión, fuente: /www.analyticsvidhya.com

Este algoritmo tiene muchas ventajas entre las cuales tenemos:

- Es diverso no todos los atributos o variables son utilizados al hacer cada árbol, cada árbol es diferente
- La dimensionalidad no es un problema para Random Forest dado que no todas las features se utilizan en los árboles
- Los resultados se basan en mayoría o promedio por lo tanto el overfitting está controlado

Dataframe df_rf:

Igual que en los procedimientos anteriores lo primero es convertir el target variable a category y cambiar los 0 a 'NO', 1 a 'SI' en el dataframe `df_rf`. Procedimiento que se hace con `replace()` y `astype()`

```
df_rf['number_treatments']=df_rf['number_treatments'].replace
([0], 'NO')

df_rf['number_treatments']=df_rf['number_treatments'].replace
([1], 'YES')
```

La variable target se convierte a category

```
df_rf.number_treatments=df_rf.number_treatments.astype('categ
ory')
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1227 entries, 0 to 1518
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   number_treatments  1227 non-null   category
 1   LDH357_t1          1227 non-null   int32
 2   MCC320_t1          1227 non-null   float32
 3   ERY316_t1          1227 non-null   float32
 4   HKT322_t1          1227 non-null   float32
 5   FT4402_t1          1227 non-null   float32
 6   GOT358_t1          1227 non-null   int32
 7   FT3401_t1          1227 non-null   float32
 8   CK355_t1           1227 non-null   int32
 9   THR323_t1          1227 non-null   int32
 10  TSH400_t1          1227 non-null   float32
 11  BMI_T1             1227 non-null   float32
 12  age                1227 non-null   float32
dtypes: category(1), float32(8), int32(4)
memory usage: 68.4 KB
```

Imagen 67 - dtypes de cada variable, number_treatments es categorical

Previamente en el análisis de correlaciones las variables `HKT322_t1` y `ERY316_t1` están muy correlacionadas por lo tanto una de ellas no se utiliza.

```
X = df_rf[['LDH357_t1', 'MCC320_t1', 'ERY316_t1','FT4402_t1',
'GOT358_t1', 'FT3401_t1', 'CK355_t1','THR323_t1','TSH400_t1',
'BMI_T1', 'age']]

y = df_rf['number_treatments']
```

Se realiza la división train / test tal como se viene haciendo

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 5, stratify = y)
```

Estos datasets se crean para guardar las predicciones de los modelos

```
dfTR_eval = X_train.copy()

dfTR_eval['Y'] = y_train

dfTS_eval = X_test.copy()

dfTS_eval['Y'] = y_test
```

La implementación de este algoritmo se hace con el paquete scikit-learn de Python la función `RandomForestClassifier.`

El parámetro `'RF__n_estimators'` se guarda en la variable `param` lo que indica es el número de árboles en el bosque de 10 en 10 máximo 200

El pipeline igual como antes se escalan los datos con el método `StandardScaler()`

El criterio seleccionado es *Gini*, `criterion='gini'` el cual viene por defecto en scikit-learn cuya función es medir la calidad de las divisiones. El índice de Gini mide el grado de pureza de un nodo. mide la probabilidad de no sacar dos registros de la misma clase del nodo. A mayor índice de Gini menor pureza, por lo que seleccionaremos la variable con menor Gini ponderado. Los rangos de pureza de Gini van de 0 a 0.5 siendo 0.5 el peor y 0 la mejor puntuación.

$$Gini(t) = 1 - \sum_{i=1}^{j} P\,(i|t)^2$$

donde *j* representa el número de clases en la etiqueta y *P* representa la proporción de la clase en el i-ésimo nodo.

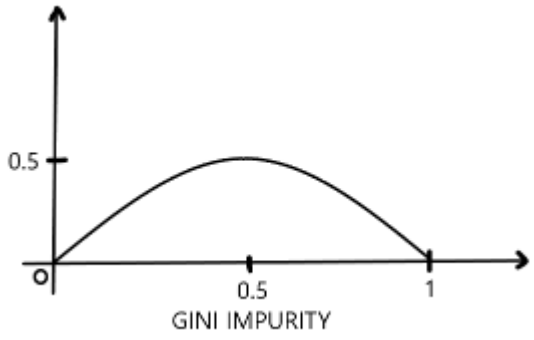


Imagen - índice gini con los rangos del 0 al 0.5 fuente: www.analyticsindiamag.com/

Otros parámetros son:

- max_features: el número de características a tener en cuenta a la hora de buscar el mejor split, se utiliza el tamaño del dataframe de entrenamiento con el método `len()` de esta forma `len(X_train.columns)`

- min_samples_split: el número mínimo de muestras necesarias para dividir un nodo interno
- min_samples_leaf: el número mínimo de muestras requerido para estar en un nodo hoja
- max_depth: es la profundidad máxima del árbol. Si es None, los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos muestras que min_samples_split se deja en 5
- random_state: se usa para replicación

```
param = {'RF__n_estimators': range(10, 200, 10)} #Number of
trees to grow

pipe = Pipeline(steps=[('scaler', StandardScaler()),

('RF',RandomForestClassifier(criterion='gini', #impuriry
measure

max_features=len(X_train.columns),  #number of variables randomly
sampled as candidates at each split.

min_samples_split=5,  #Minimum number of obs in node to keep cutting

min_samples_leaf=5, #Minimum number of obs in a terminal node

max_depth=5, #

random_state=150))]) # For replication
```

Se utiliza Cross Validation para validar el modelo, los mismos parámetros anteriores, se utiliza esta técnica de Gridsearch para obtener el mejor modelo de Random Forest y con este se hacen las predicciones del test set.

```
nFolds = 10

rf_fit = GridSearchCV(estimator=pipe, #

param_grid=param, # Defined grid to search in

n_jobs=-1,

scoring='accuracy', # Accuracy

cv=nFolds) # Number of Folds

rf_fit.fit(X_train, y_train) # Search in grid
```

Seguidamente se van a graficar la mejor accuracy junto con el mejor número de árboles que indique el modelo para lo cual se utiliza el atributo `cv_results_` del Cross Validation el cual es un diccionario con las claves como encabezados y los valores como columnas y el cual se puede importar a `dataframe pandas`.

```
{
'param_kernel': masked_array(data = ['poly', 'poly', 'rbf', 'rbf'],
                             mask = [False False False False]...)
'param_gamma': masked_array(data = [-- -- 0.1 0.2],
                            mask = [ True  True False False]...),
'param_degree': masked_array(data = [2.0 3.0 -- --],
                             mask = [False False  True  True]...),
'split0_test_score'  : [0.80, 0.70, 0.80, 0.93],
'split1_test_score'  : [0.82, 0.50, 0.70, 0.78],
'mean_test_score'    : [0.81, 0.60, 0.75, 0.85],
'std_test_score'     : [0.01, 0.10, 0.05, 0.08],
'rank_test_score'    : [2, 4, 3, 1],
'split0_train_score' : [0.80, 0.92, 0.70, 0.93],
'split1_train_score' : [0.82, 0.55, 0.70, 0.87],
'mean_train_score'   : [0.81, 0.74, 0.70, 0.90],
'std_train_score'    : [0.01, 0.19, 0.00, 0.03],
'mean_fit_time'      : [0.73, 0.63, 0.43, 0.49],
'std_fit_time'       : [0.01, 0.02, 0.01, 0.01],
'mean_score_time'    : [0.01, 0.06, 0.04, 0.04],
'std_score_time'     : [0.00, 0.00, 0.00, 0.01],
'params'             : [{'kernel': 'poly', 'degree': 2}, ...],
}
```

Imagen 68 - diccionario cv_results_ de scikit-learn fuente www.scikit-learn.org/

```
nFolds = 10

cv_errors=np.empty([nFolds,len(rf_fit.cv_results_['split0_tes
t_score'])])
```

```
for split in range(nFolds):

     cv_errors[split,:] = rf_fit.cv_results_['split'+
     str(split) + '_test_score']

meanAcc = cv_errors.mean(0)

stdAcc = cv_errors.std(0)

scores_rf = meanAcc # Store cv-scores for later
```

Finalmente con el uso de Matplotlib se grafican el accuracy y el número de árboles del modelo donde los resultados de cada iteración se utiliza en forma de dataframe esta opción `cv_results_['param_RF__n_estimators']` la cual muestra el accuracy el cual se acerca a 64% previamente definido en `meanAcc` y donde las líneas azules muestran el error estándar de cada árbol guardado en esta variable `stdAcc`. El mejor modelo se construye con 160 árboles.

```
plt.figure(figsize=(12, 6))

plt.plot(rf_fit.cv_results_['param_RF__n_estimators'],
meanAcc, marker='o', markersize=10)

plt.errorbar(rf_fit.cv_results_['param_RF__n_estimators'],
meanAcc, yerr=stdAcc, linestyle="None", ecolor='lightblue')

plt.plot(rf_fit.best_params_['RF__n_estimators'],
rf_fit.best_score_,marker='o', markersize=15, color='red')

plt.title('Accuracy Rate number of trees Value. Best model
with ' + str(rf_fit.best_params_['RF__n_estimators']) + '
trees')

plt.xlabel('Number of trees')

plt.ylabel('Accuracy')

plt.show()
```

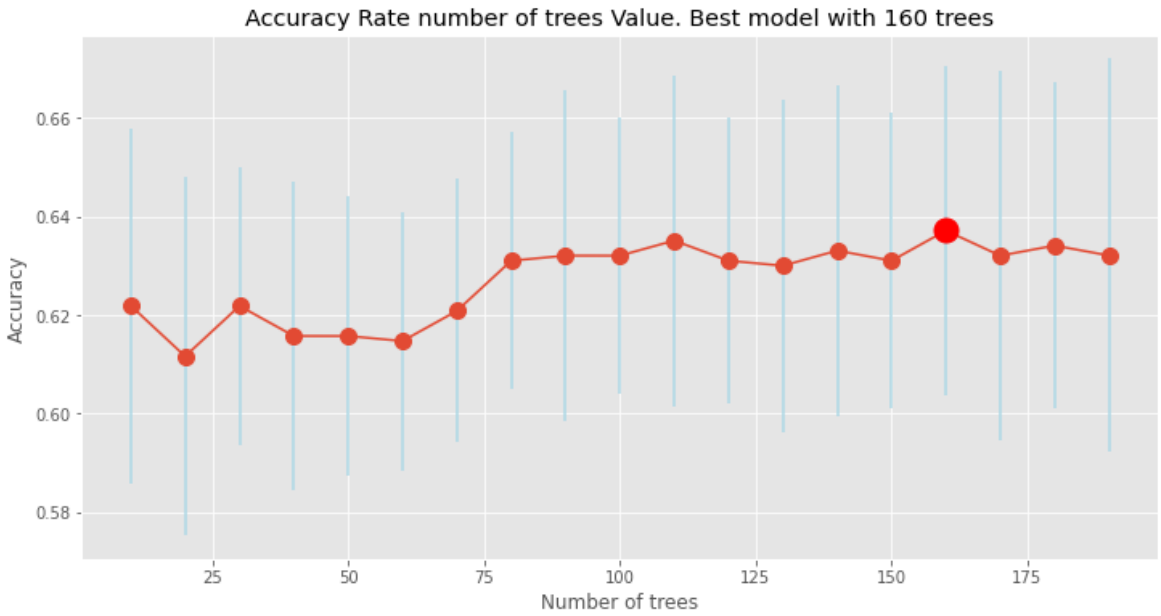


Imagen 69 - Accuracy con el mejor modelo y número ideal de árboles ideal modelo df_rf

Es importante también conocer las features más importantes utilizadas en el modelo se utiliza el atributo `best_estimator_` el cual devuelve los mejores resultados del `GridSearchCV`.

Se utiliza la librería matplotlib como se viene haciendo, en un gráfico de barras

```
plt.figure(figsize=(9 * 1.618, 6))

index = np.arange(len(X_train.columns))

bar_width = 0.35

plt.bar(index,rf_fit.best_estimator_['RF'].feature_importance
s_,color='black', alpha=0.5)

plt.xlabel('features')

plt.ylabel('importance')

plt.title('Feature importance')

plt.xticks(index, X_train.columns)

plt.tight_layout()
```

```
plt.show()
```

Se aprecia como las más importantes son `MCC320_t1`, `LDH357_t1`, `ERY316_t1` y `age`

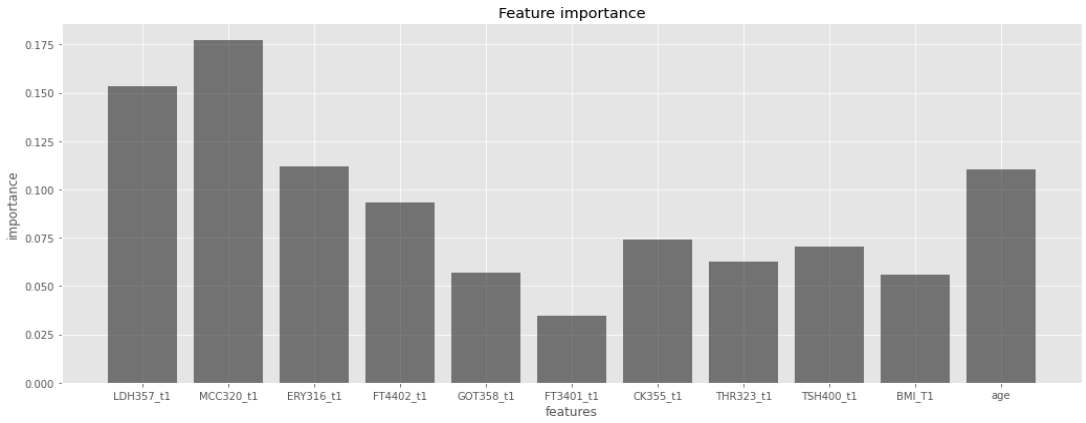


Imagen 70 - Gráfico de barras con las variables más importantes del modelo df_rf

5.5 EVALUATION

Para la evaluación del modelo se usarán varias técnicas para ideales para modelos de Clasificación por ejemplo la Matriz de Confusión que es una medida de rendimiento para un problema de clasificación de aprendizaje automático en el que la salida puede ser de dos o más clases. Es una tabla con 4 combinaciones diferentes de valores predichos y reales.

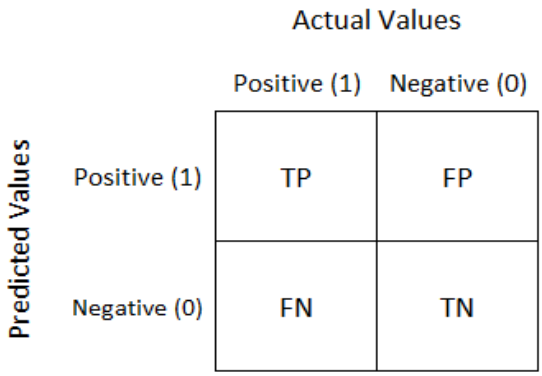


Imagen 71 - Clases predichas y reales en matriz de confusión fuente: www./towardsdatascience.com

A continuación, se detallan los conceptos principales de la matriz de confusión

- Precision: la precisión se define como la relación entre los verdaderos positivos y la suma de los verdaderos y falsos positivos. Es decir, de todas las predicciones positivas hechas, cuántas son realmente positivas
- Recall: es todos los ejemplos positivos reales que hay, cuántos de ellos se predijeron correctamente que serían positivos.
- F1 Score: es la media armónica de la precisión y la recuperación para un resumen más equilibrado del rendimiento del modelo. Cuanto más se acerque el valor de la puntuación F1 a 1.0 mejor será el rendimiento esperado del modelo

Continuando con el proceso de evaluación para obtener las predicciones en train y test se hacen las siguientes asignaciones a los dataframes creados previamente

```
## Obtain a report of the model based on predictions

dfTR_eval['Y_LR_pred'] = LogReg_fit.predict(X_train)
```

```
dfTR_eval['Y_LR_prob_neg']=LogReg_fit.predict_proba(X_train)[
:,0]

dfTR_eval['Y_LR_prob_pos']=LogReg_fit.predict_proba(X_train)[
:,1]

### Scale test using preprocess in training

dfTS_eval['Y_LR_pred'] = LogReg_fit.predict(X_test)

dfTS_eval['Y_LR_prob_neg']=LogReg_fit.predict_proba(X_test)[:
,0]

dfTS_eval['Y_LR_prob_pos']=LogReg_fit.predict_proba(X_test)[:
,1]
```

Los siguientes son los datos de evaluación para el conjunto de training `dfTR_eval`

- Accuracy: tiene un valor de 65% el cual es la relación de la suma de los verdaderos positivos con los verdaderos negativos con la suma de los elementos verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos.
- No Information Rate: es la precisión que se puede conseguir prediciendo siempre la etiqueta de la clase mayoritaria, es la mayor proporción de las clases observadas en este caso 0.52
- P-Value [Acc > NIR]: el valor es 0.0 el accuracy es más alto que el NIR
- Kappa: el coeficiente *kappa* mide la concordancia entre la clasificación y los valores de verdad, tiene una escala de -1 a 1, donde 1 representa una concordancia perfecta, 0 es exactamente lo que se esperaría por azar y los valores negativos indican una concordancia menor que el azar este caso fue de 0,27 más cercano a 0
- Mcnemar's Test P-Value: p-values de la técnica no paramétrica utilizada para las variables categóricas. Puede utilizarse con dos medidas dicotómicas en los mismos sujetos (medidas repetidas). La hipótesis nula muestra que las dos probabilidades de

cada resultado son iguales: $Ptp + Pfp = Ptp + Pfn$ and $Pfn + Ptn = Pfp + Ptn$ Mcnemar's equation $X^2 = \frac{(fp - fn)\text{^}2}{fp + fn}$

- Sensitivity: tasa de verdaderos positivos, valor de 0.47
- Specificity: tasa de verdaderos negativos, valor de 0.79
- Pos pred value: tasa de positivos capturados entre el total de positivos previstos, el valor es de 0.65
- Neg pred value: tasa de negativos capturados entre el total de negativos previstos, el valor es de 0.65
- Prevalence: es la tasa de "Todos los Positivos REALES" en toda la población el valor es de 0.45
- Detection Rate: es entre la población total, cuánto se detecta el valor es de 0.21
- Detection prevalence: es la tasa de "Todos los PREDICTIVOS" en toda la población cuyo valor es 0.33
- Balanced accuracy: es la media de la sensibilidad y la especificidad, valor es 0.63
- F Score: es la media armónica de la precisión y la recuperación para un resumen más equilibrado del rendimiento del modelo valor de 0.55
- Positive class: la clase positiva representa la clase o comportamiento no normal, por lo que suele estar menos representada que la otra clase, el valor es "SI", en este caso el modelo se ajusta mejor a los “NO” predice mejor cuando un paciente no es ingresado

El siguiente comando ejecuta el `summary` con los datos de la Confusion Matrix para el dataframe de training

```
CT.confusion_matrix(dfTR_eval['Y'], dfTR_eval['Y_LR_pred'])
```

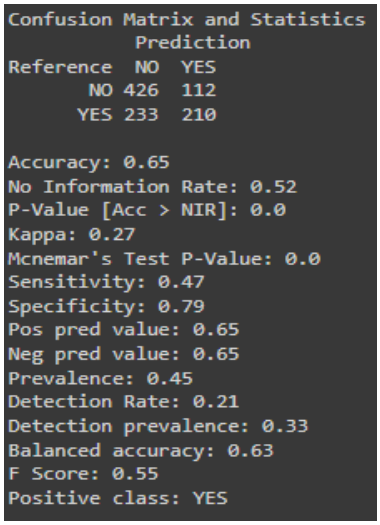


Imagen 72 - Resultados del dataframe en training dfTR_eval del modelo df_rl

Los siguientes son los datos de evaluación para el conjunto de test `dfTS_eval` del modelo `df_rl`, los parámetros de esta función ya fueron definidos previamente, se indican sólo los valores.

- Accuracy: 0.63
- No Information Rate: 0.52
- P-Value [Acc > NIR]: 0.01
- Kappa: 0.23
- Mcnemar's Test P-Value: 0.0
- Sensitivity: 0.79
- Specificity: 0.43
- Pos pred value: 0.63
- Neg pred value: 0.63
- Prevalence: 0.55
- Detection Rate: 0.43
- Detection prevalence: 0.69
- Balanced accuracy: 0.61

- F Score: 0.7
- Positive class: NO

Los datos de test de la Confusion Matrix son los siguientes y se ejecutan con el comando

```
CT.confusion_matrix(dfTS_eval['Y'], dfTS_eval['Y_LR_pred'])
```

```
Confusion Matrix and Statistics
          Prediction
Reference  YES  NO
      YES   48  63
       NO   28 107

Accuracy: 0.63
No Information Rate: 0.52
P-Value [Acc > NIR]: 0.01
Kappa: 0.23
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.79
Specificity: 0.43
Pos pred value: 0.63
Neg pred value: 0.63
Prevalence: 0.55
Detection Rate: 0.43
Detection prevalence: 0.69
Balanced accuracy: 0.61
F Score: 0.7
Positive class: NO
```

Imagen 73 - Resultados en testing dataframe dfTS_eval del modelo df_rl

- Calibration Plot: es una herramienta visual para evaluar la concordancia entre las predicciones y las observaciones en diferentes percentiles (principalmente deciles) de los valores predichos.

  De acuerdo con el calibration plot las predicciones del modelo de training ajusta bastante a la recta ideal donde el acierto de probabilidad se encuentra estable a lo largo de los valores de predicción.

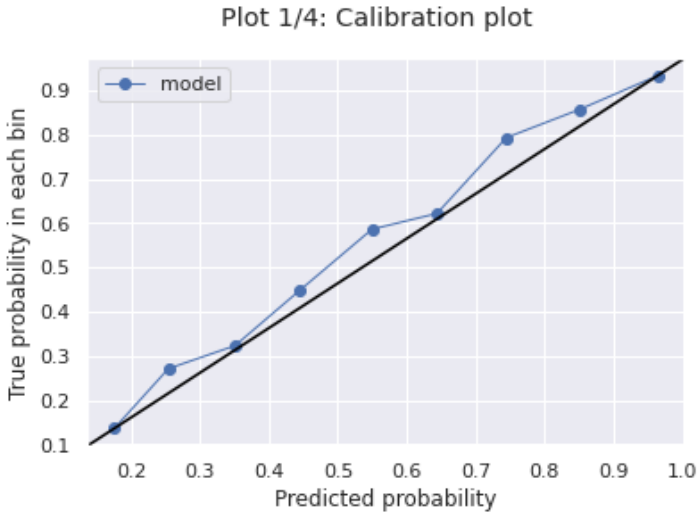


Imagen 74 - Calibration plot del dataframe training dfTR_eval del modelo df_rl

- Histograma: un histograma es la representación gráfica de frecuencias de una variable, el histograma muestra como la frecuencia más alta está en la categoría de NO, los valores para SI son más pequeños, puesto el modelo de training predice mejor la categoría de NO.

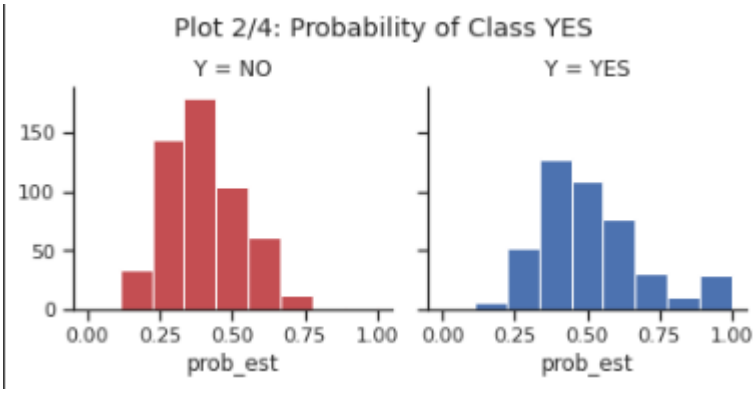


Imagen 75 - Histograma del dataframe training dfTR_eval del modelo df_rl

- ROC curve: la curva ROC se crea evaluando las probabilidades de clase del modelo a través de un continuo de umbrales ROC es una curva de probabilidad y AUC representa el grado o medida de separabilidad. Indica en qué medida el modelo es

capaz de distinguir entre clases. Cuanto más alto sea el AUC, mejor será el modelo para predecir las clases 0 como 0 y las clases 1 como 1. Esta es la ecuación $dROC = \sqrt{\frac{(TPR - FPR)\text{^}2}{2}}$.

El modelo predice en buena forma de 0.686 los verdaderos positivos, no llega al hasta el extremo izquierdo que es lo ideal para la tasa de TP pero supera el mínimo de 0.5

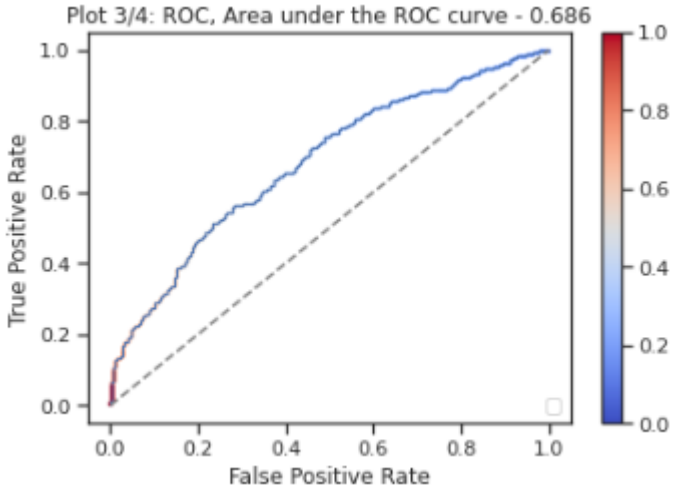


Imagen 76 - Curva ROC ROC del dataframe training dfTR_eval del modelo df_rl

- Accuracy cutoffs chart: gráfico el cual muestra las accuracies más altas, este modelo tiene el accuracy más alto en el corte 0.5

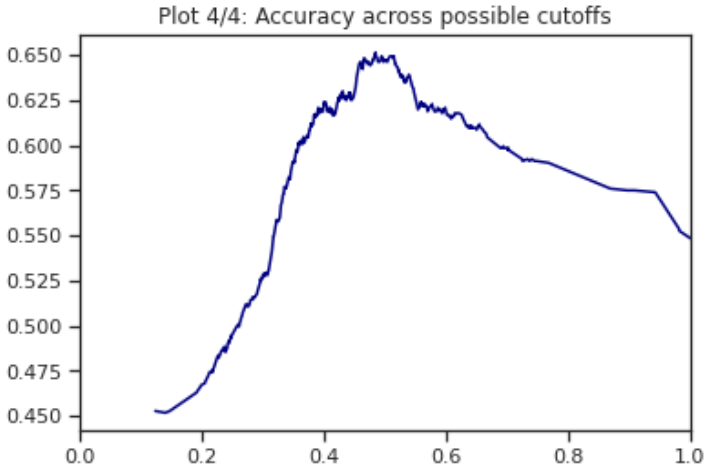


Imagen - Las accuracies más altas están entre los puntos 0.4 y 0.6 del dataframe dfTR_eval del modelo df_rl

Continuando con el otro modelo del dataframe `df_mr` se sigue el mismo proceso con las asignaciones para obtener los reportes de los modelos

```
## Obtain a report of the model based on predictions

dfTR_eval['Y_LR_pred']=LogReg_fit.predict(X_train)

dfTR_eval['Y_LR_prob_neg']=LogReg_fit.predict_proba(X_train)[
:,0]

dfTR_eval['Y_LR_prob_pos']=LogReg_fit.predict_proba(X_train)[
:,1]

dfTS_eval['Y_LR_pred'] = LogReg_fit.predict(X_test)

dfTS_eval['Y_LR_prob_neg']=LogReg_fit.predict_proba(X_test)[:
,0]

dfTS_eval['Y_LR_prob_pos']=LogReg_fit.predict_proba(X_test)[:
,1]
```

Los conceptos de Confusion Matrix ya fueron definidos previamente, los resultados e imagen en training son estos

Se lanza el comando para la Confusion Matrix en training

```
CT.confusion_matrix(dfTR_eval['Y'], dfTR_eval['Y_LR_pred'])
```

- Accuracy: 0.67, en este modelo el accuracy es un poco más alto
- No Information Rate: 0.51
- P-Value [Acc > NIR]: 0.0
- Kappa: 0.33 mas alto que el modelo anterior
- Mcnemar's Test P-Value: 0.0
- Sensitivity: 0.53 más alto que el modelo anterior, es la tasa de verdaderos positivos
- Specificity: 0.78 es la tasa de verdaderos negativos
- Pos pred value: 0.67 un poco mayor que el anterior modelo

- Neg pred value: 0.67 un poco mayor que el anterior modelo
- Prevalence: 0.45
- Detection Rate: 0.24
- Detection prevalence: 0.36
- Balanced accuracy: 0.66
- F Score: 0.6
- Positive class: YES

En general los valores mejoran un poco en este modelo la positive class sigue siendo YES

```
Confusion Matrix and Statistics
          Prediction
Reference  NO  YES
       NO 422  116
      YES 206  237

Accuracy: 0.67
No Information Rate: 0.51
P-Value [Acc > NIR]: 0.0
Kappa: 0.33
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.53
Specificity: 0.78
Pos pred value: 0.67
Neg pred value: 0.67
Prevalence: 0.45
Detection Rate: 0.24
Detection prevalence: 0.36
Balanced accuracy: 0.66
F Score: 0.6
Positive class: YES
```

Imagen 77 - Resultados del dataframe dfTR_eval del modelo df_mr

Los siguientes son los datos de evaluación para el conjunto de test `dfTS_eval` del modelo `df_mr`, los parámetros de esta función ya fueron definidos previamente, se indican sólo los valores.

- Accuracy: 0.65
- No Information Rate: 0.51
- P-Value [Acc > NIR]: 0.0
- Kappa: 0.28

- Mcnemar's Test P-Value: 0.07
- Sensitivity: 0.75
- Specificity: 0.53
- Pos pred value: 0.66
- Neg pred value: 0.63
- Prevalence: 0.55
- Detection Rate: 0.41
- Detection prevalence: 0.62
- Balanced accuracy: 0.64
- F Score: 0.7
- Positive class: NO

Los valores son parecidos al test anterior el accuracy es un poco más alto.

```
Confusion Matrix and Statistics
          Prediction
Reference  YES  NO
      YES   59  52
       NO   34 101

Accuracy: 0.65
No Information Rate: 0.51
P-Value [Acc > NIR]: 0.0
Kappa: 0.28
Mcnemar's Test P-Value: 0.07
Sensitivity: 0.75
Specificity: 0.53
Pos pred value: 0.66
Neg pred value: 0.63
Prevalence: 0.55
Detection Rate: 0.41
Detection prevalence: 0.62
Balanced accuracy: 0.64
F Score: 0.7
Positive class: NO
```

Imagen 78 - Resultados en testing dataframe dfTS_eval del modelo df_mr

Los diagramas utilizados en training para este modelo `df_mr` se ejecutan con el siguiente comando, igual que el modelo anterior se usa el dataframe `dfTR_eval`

```
CT.plotClassPerformance(dfTR_eval['Y'],LogReg_fit.predict_pro
ba(X_train), selClass='YES')
```

- Calibration plot: en este modelo la tendencia es bastante buena ya que el modelo se ajusta bastante a la predicción

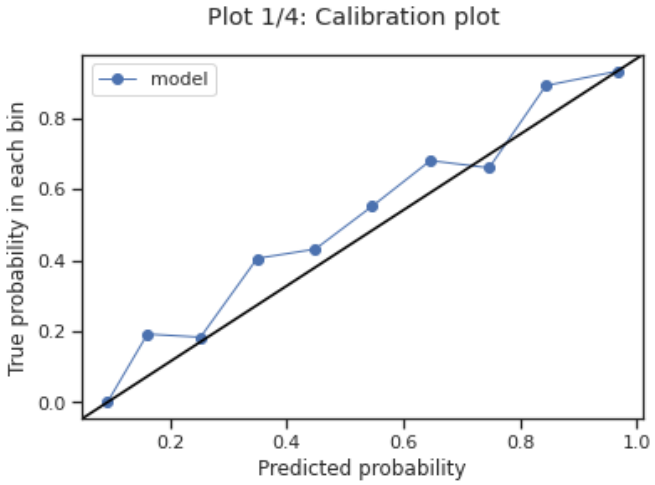


Imagen 79 - Calibration plot dataframe dfTR_eval model df_mr el modelo ajustado

- Histograma: los valores del histograma de la clase YES son parecidos indica un modelo robusto puesto las accuracies están en un rango parecido, el número mas alto de la clase NO, tuvo mala predicción de un 0.25

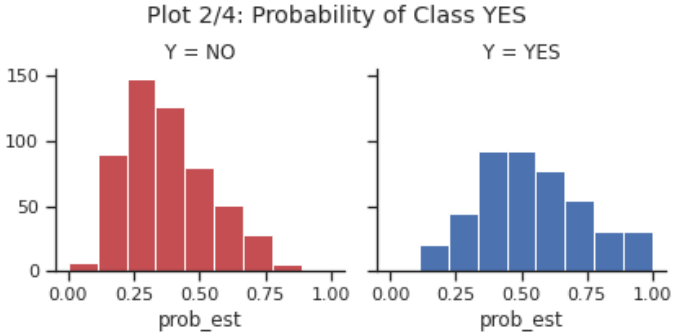


Imagen 80 - Histograma del dataframe dfTR_eval model df_mr

- ROC curve: el modelo predice bien con un valor alto de 0.732 los verdaderos positivos la clase YES con sus verdaderos positivos tiene buena predicción

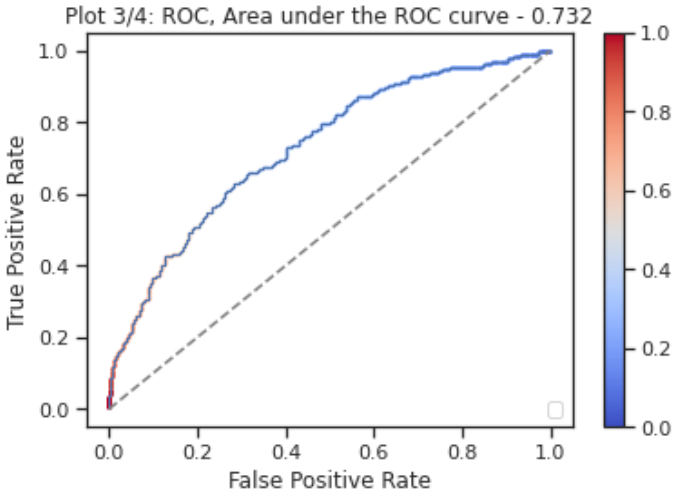


Imagen 81 - Área bajo la curva valor de 0.732 dataframe dfTR_eval model df_mr

- Accuracy cutoffs chart: el nivel de accuracy y cutoff es parejo hay un valor estable entre los cortes 0.5 y 0.6 el valor del accuracy 0.65 es parecido al modelo anterior

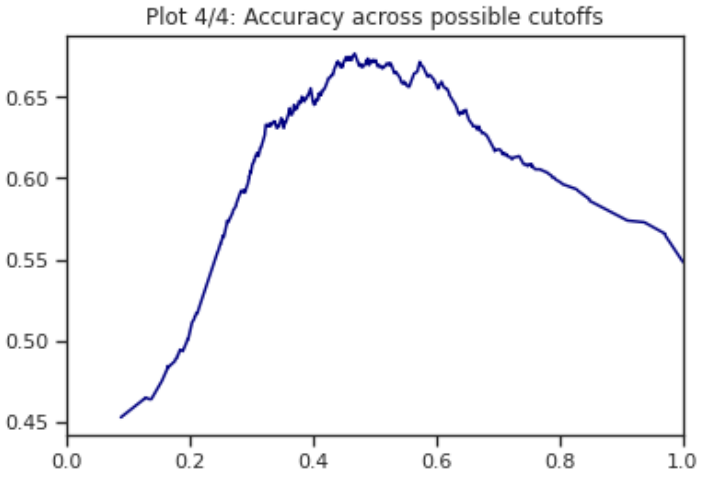


Imagen - Cutoff y accuracy del dataframe dfTR_eval model df_mr

Siguiendo con el modelo de Random Forest del dataframe `dr_rf`, es esta etapa de evaluación se continúa con el mismo proceso donde se utilizan las mismas métricas

Se crean los datasets para los reportes de predicciones

```
dfTR_eval['Y_RF_pred'] = rf_fit.predict(X_train)
dfTR_eval['Y_RF_prob_neg']=rf_fit.predict_proba(X_train)[:,0]
dfTR_eval['Y_RF_prob_pos']=rf_fit.predict_proba(X_train)[:,1]
dfTS_eval['Y_RF_pred'] = rf_fit.predict(X_test)
dfTS_eval['Y_RF_prob_neg']= rf_fit.predict_proba(X_test)[:,0]
dfTS_eval['Y_RF_prob_pos']= rf_fit.predict_proba(X_test)[:,1]
```

Se lanza el comando para la Confusion Matrix en training del dataframe `df_rf`

```
CT.confusion_matrix(dfTR_eval['Y'], dfTR_eval['Y_LR_pred'])
```

- Accuracy: 0.8 en este modelo el accuracy es más alto
- No Information Rate: 0.51
- P-Value [Acc > NIR]: 0.0
- Kappa: 0.6 entre más cercano a 1 mejor
- Mcnemar's Test P-Value: 0.0
- Sensitivity: 0.69 es la tasa de verdaderos positivos
- Specificity: 0.9 es la tasa de verdaderos negativos
- Pos pred value: 0.85 más alto que los modelos anteriores
- Neg pred value: 0.78 más alto que los modelos anteriores
- Prevalence: 0.45
- Detection Rate: 0.31
- Detection prevalence: 0.36
- Balanced accuracy: 0.79
- F Score: 0.76
- Positive class: YES

En este caso los valores fueron más altos que los anteriores modelos ya que Random Forest es un algoritmo bastante eficiente, dado su naturaleza *Bagging* donde se unen muchos árboles de decisión es la reducción de las correlaciones entre los árboles bootstrap individuales y permitiendo así que haya una reducción mayor de la varianza. Por lo tanto el modelo tiene un sesgo bajo y una varianza moderada.

El accuracy fue mucho mayor que los modelos anteriores el F Score también fue alto debido a la tasa de verdaderos positivos y verdaderos negativos.

```
Confusion Matrix and Statistics
          Prediction
Reference  NO  YES
       NO 485   53
      YES 139  304

Accuracy: 0.8
No Information Rate: 0.51
P-Value [Acc > NIR]: 0.0
Kappa: 0.6
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.69
Specificity: 0.9
Pos pred value: 0.85
Neg pred value: 0.78
Prevalence: 0.45
Detection Rate: 0.31
Detection prevalence: 0.36
Balanced accuracy: 0.79
F Score: 0.76
Positive class: YES
```

Imagen 82 - Resultados del training dataframe dfTR_eval modelo df_rf

Los siguientes son los datos de evaluación para el conjunto de test `dfTS_eval` del modelo `df_rf`, ya se explicaron los resultados en training para este modelo de Random Forest lo siguiente es ver el comportamiento en el conjunto de test.

```
CT.confusion_matrix(dfTS_eval['Y'], dfTS_eval['Y_RF_pred'])
```

- Accuracy: 0.6
- No Information Rate: 0.51

- P-Value [Acc > NIR]: 0.05
- Kappa: 0.18
- Mcnemar's Test P-Value: 0.02
- Sensitivity: 0.73
- Specificity: 0.45
- Pos pred value: 0.62
- Neg pred value: 0.57
- Prevalence: 0.55
- Detection Rate: 0.4
- Detection prevalence: 0.65
- Balanced accuracy: 0.59
- F Score: 0.67
- Positive class: NO

En este caso hay una diferencia importante entre accuracies pero no determinante, a pesar que los Random Forest no tienen overfitting según Breiman y Cutler *(Breiman, Leo; Cutle, Adele , 2022)* hay ocasiones en que si hay overfitting, cuando se agregan árboles al Randon Forest, la tendencia al overfitting debería disminuir (gracias al bagging y al random feature selection). Sin embargo, el error de generalización no llegará a cero. La varianza del error de generalización se acercará a cero con más árboles agregados, pero el sesgo no lo hará. Cuantos más árboles haya Random Forest mejor.

Por lo tanto, uno de los parámetros importantes al lanzar el algoritmo es `max_depth` este valor es bueno dejarlo bajo se deja en 5 puesto al dejarlo muy alto el árbol empieza a sobreentrenar el conjunto de entrenamiento y por lo tanto, no es capaz de generalizar sobre los puntos no vistos del test set. Este parámetro indica el camino más largo entre el nodo raíz y el nodo hoja, el valor por defecto es *None* por lo tanto los nodos se expanden hasta que todas las hojas sean puras o hasta que todas las hojas contengan menos muestras que `min_samples_split`.

```
Confusion Matrix and Statistics
          Prediction
Reference  YES  NO
      YES   50  61
       NO   37  98

Accuracy: 0.6
No Information Rate: 0.51
P-Value [Acc > NIR]: 0.05
Kappa: 0.18
Mcnemar's Test P-Value: 0.02
Sensitivity: 0.73
Specificity: 0.45
Pos pred value: 0.62
Neg pred value: 0.57
Prevalence: 0.55
Detection Rate: 0.4
Detection prevalence: 0.65
Balanced accuracy: 0.59
F Score: 0.67
Positive class: NO
```

Imagen 83 - Resultados del dataframe en testing dfTS_eval del modelo df_rf

Continuando con los gráficos de evaluación del modelo `df_rf` se ejecutan de la misma forma anterior

```
CT.plotClassPerformance(dfTR_eval['Y'],rf_fit.predict_proba(X_train)[:,], selClass='YES')
```

- Calibration plot: se aprecia como el modelo ajusta bastante bien al principio y finalmente acierta en el predicho, no es perfecto a lo largo pero ajusta bien.

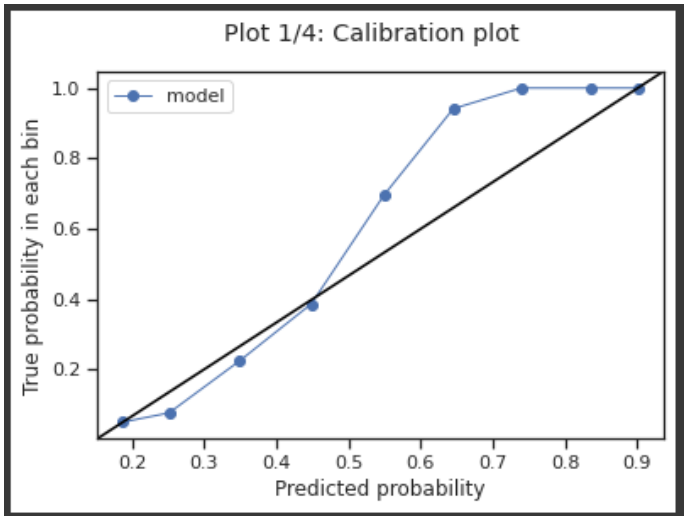


Imagen 84 - Calibration plot del dataframe de training dfTR_eval modelo df_rf

- Histograma: la probabilidad de la clase YES tiene un valor importante de 0.75 una frecuencia alta el modelo en training acierta más con la clase YES como se mostró en imagen anterior, la clase NO tiene probabilidad más baja de acierto con 0.25.

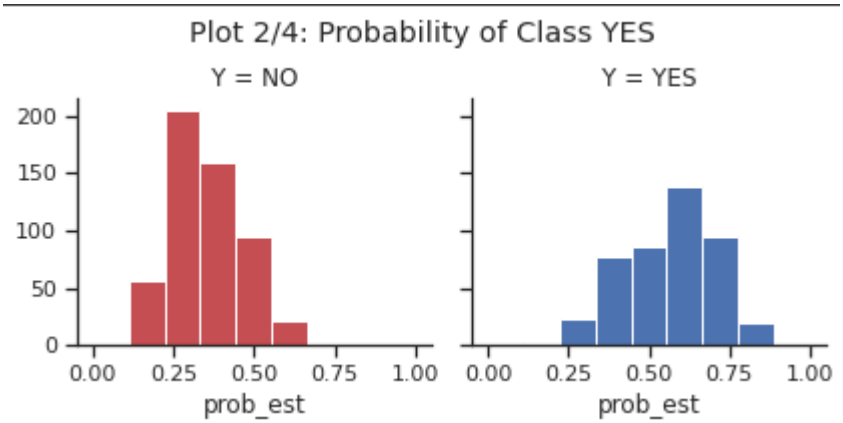


Imagen 85 - Histograma del dataframe de training dfTR_eval modelo df_rf

- ROC curve: la tasa de verdaderos positivos es bastante alta el performance del modelo es bastante ideal prediciendo verdaderos positivos.

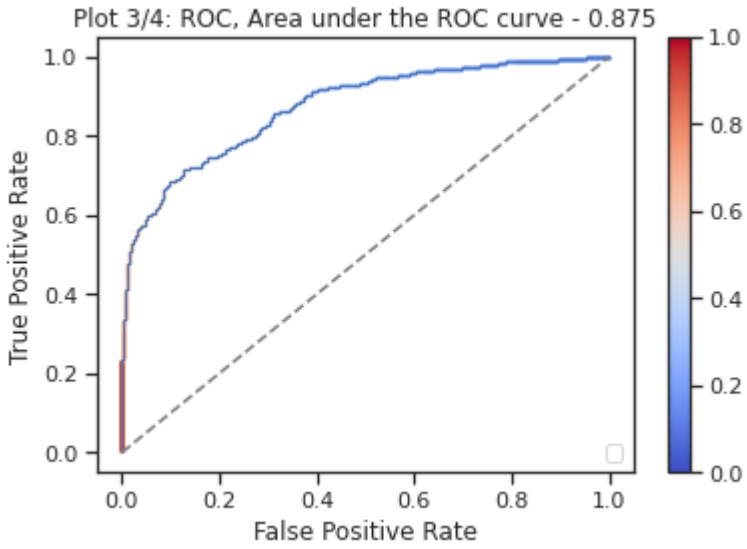


Imagen 86 - ROC chart del dataframe de training dfTR_eval modelo df_rf

- Accuracy cutoffs chart: este gráfico indica en qué punto se encuentra el accuracy más alto, en el 0.5 se encuentra el accuracy más alto

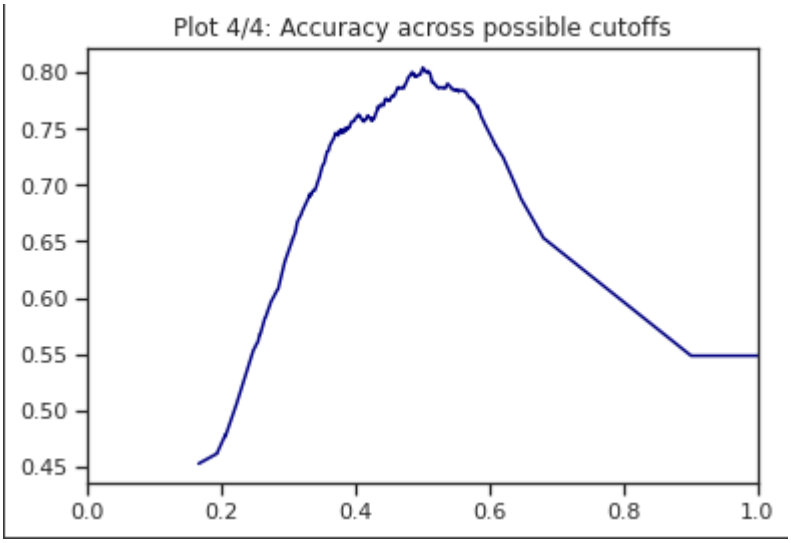


Imagen - El accuracy más entre los puntos posibles del modelo del dataframe de training `dfTR_eval` *modelo* `df_rf`

Además de los modelos creados anteriormente, también se creó otro modelo con la unión de los tres marcos de datos, utilizando también el modelo final de regresión logística y el bosque aleatorio.

El modelo se compone de las variables de los tres dataframes anteriores, algunas en común con los otros dataframes. El nuevo marco de datos es df_join y tiene 28 columnas.

```
Index(['number_treatments', 'ZCVLT_LS_DG1_5_RW1', 'HBE318_t1', 'HKT322_t1',
       'ZCVLT_WAII_RW1', 'rehab_weeks_4_6', 'RZExecut_2', 'NZExecut_2',
       'ERY316_t1', 'NZDiff_a', 'MCV319_t1', 'RZDiff_a', 'LDH357_t1',
       'MCC320_t1', 'NZVerbal_2', 'spg419_t1', 'RZExcecu', 'FT4402_t1',
       'RZVerbal', 'height_m', 'RZVerbal_2', 'GOT358_t1', 'FT3401_t1',
       'CK355_t1', 'THR323_t1', 'TSH400_t1', 'BMI_T1', 'age'],
      dtype='object')
```

Imagen 87 – Columnas del dataframe df_join con las columnas utilizadas de los otros modelos

Este nuevo marco de datos está más correlacionado como se muestra en el mapa de calor, hay variables que serán excluidas del modelo debido a su alta correlación, porque todas las variables de los tres marcos de datos son muy similares porque representan valores o información similar.

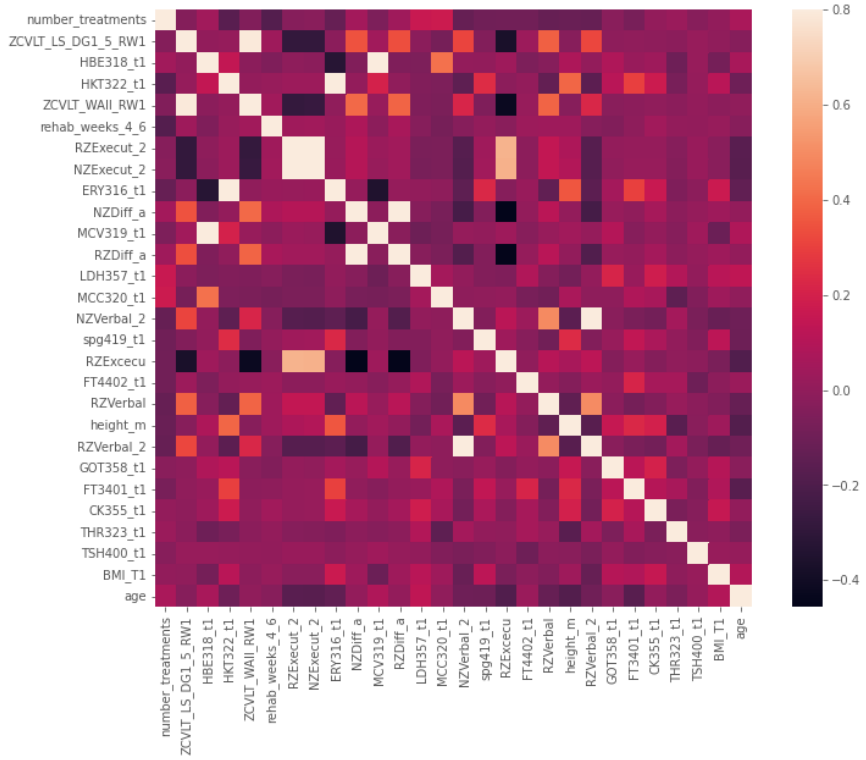


Imagen 88 - Heatmap del dataframe df_join con algunas features altamente correlacionadas

Modelo final Regresión Logística

Este modelo al igual que los anteriores se utiliza en clasificación, en esta nueva iteración todas las variables de los otros modelos son utilizadas, este modelo debe tener mejores accuracies, el procedimiento es el mismo, se entrena el modelo, se valida con técnicas de Validación Cruzada y se analizan los resultados con matriz de confusión.

La siguiente imagen muestra la precisión en cada uno de los pliegues de la técnica de Validación Cruzada, la media es de 0,67 - 67%, este valor es muy similar al de los otros modelos, los valores son parecidos, no hay grandes diferencias entre ellos, lo que denota un buen modelo robusto, sin sobreajuste y que predice correctamente, como se indicó anteriormente, las accuracies de los modelos están dentro del 64% - 67%.

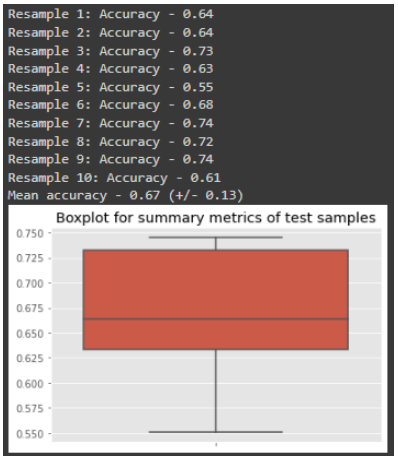


Imagen 89 - Boxplot con cada accuracy de Cross Validation

Los valores de este nuevo modelo son:

- Accuracy: 0.68
- F-score: 0.6

Este modelo predice mejor los verdaderos negativos que los verdaderos positivos, la especificidad o Specificity es superior a 0,78, Balanced Accuracy por encima de 0,5 lo que indica un valor aceptable y la precisión de 0,68 es un valor aceptable, los valores son similares a los modelos de regresión logística anteriores.

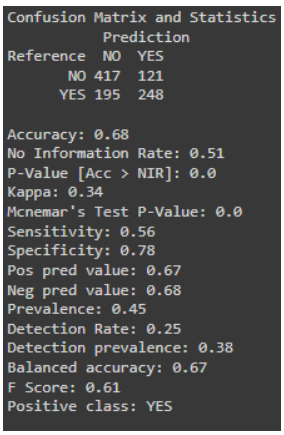
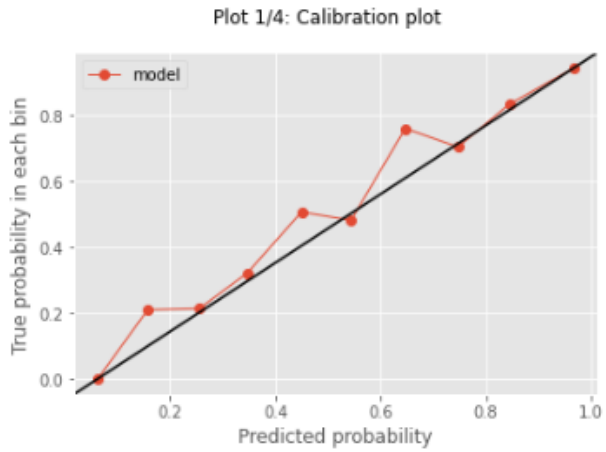
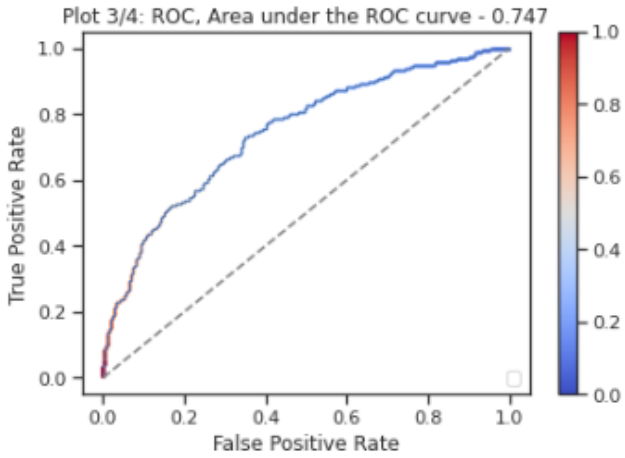


Imagen 90 - Resultados del dataframe dfTR_eval modelo df_join - regresión logística

En cuanto a las métricas de evaluación, el gráfico de calibración muestra una tendencia bastante cercana a la real, el modelo ajusta bastante bien su predicción de entrenamiento.



La curva ROC mejoró con respecto a los modelos de regresión logística anteriores, pero sigue manteniendo un nivel similar de predicciones de verdaderos positivos en 0,747, el objetivo es que esta curva esté lo más cerca posible de la esquina superior izquierda



Modelo final Random Forest

En este nuevo modelo se implementó la misma técnica con el modelo de random forest anterior, esta vez se ejecuta con el nuevo dataframe df_join utilizando la misma configuración con el modelo anterior, en este caso el número de árboles fue ligeramente

menor de 160 a 150, en general entre más árboles en el modelo se obtienen mejores resultados, sin embargo, el desempeño disminuye conforme aumenta el número de árboles, en este caso el accuracy aumentó un poco más a 0.66 anteriormente era de 0.64.

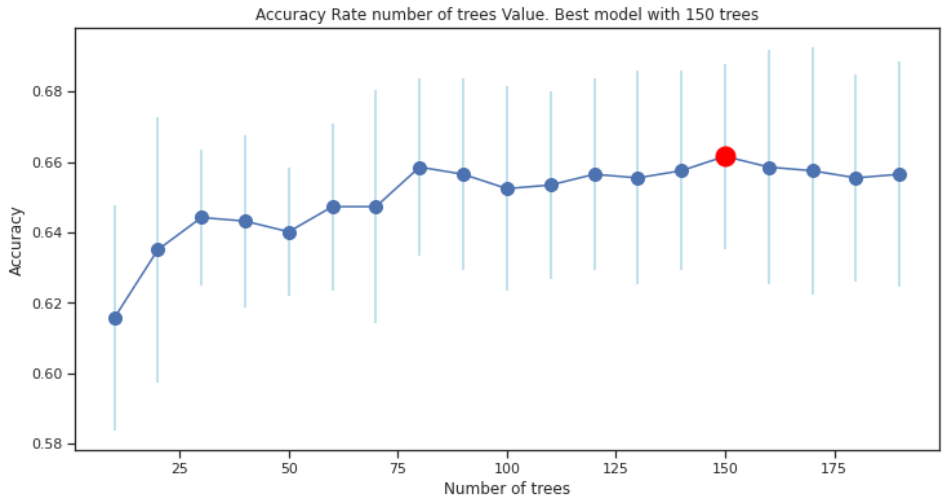


Imagen 91 - El mejor modelo se construye con 150 árboles del dataframe df_join

Los valores en este nuevo modelo aumentaron un poco, balanced accuracy es alto en 0,81, el conjunto de las variables más representativas del marco de datos converge mejor en este modelo, los números son más altos y se utilizan los mismos parámetros de ajuste para evitar el overfitting.

- Accuracy: 0.82
- F-score: 0.77

```
Confusion Matrix and Statistics
          Prediction
Reference  NO  YES
       NO 496   42
      YES 138  305

Accuracy: 0.82
No Information Rate: 0.51
P-Value [Acc > NIR]: 0.0
Kappa: 0.62
Mcnemar's Test P-Value: 0.0
Sensitivity: 0.69
Specificity: 0.92
Pos pred value: 0.88
Neg pred value: 0.78
Prevalence: 0.45
Detection Rate: 0.31
Detection prevalence: 0.35
Balanced accuracy: 0.81
F Score: 0.77
Positive class: YES
```

Imagen 92 - Resultados del dataframe dfTR_eval modelo df_join – random forest

Siguiendo con las métricas de evaluación de este modelo, el gráfico de calibración del primer modelo de random forest es muy similar a este, al principio converge bien, luego se aleja del ideal y termina bien, esto indica un buen entrenamiento del modelo sin overfitting.

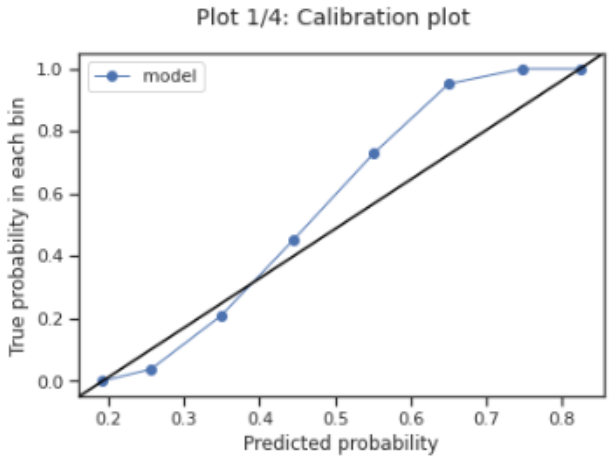


Imagen 93 - Calibration plot del dataframe en training dfTR_eval modelo df_join

La mejor tasa de verdaderos positivos es con los modelos de bosque aleatorio, este modelo mejoró incluso el primer modelo de bosque aleatorio llegando a 0,908 lo que significa que este modelo predice mejor los verdaderos positivos.

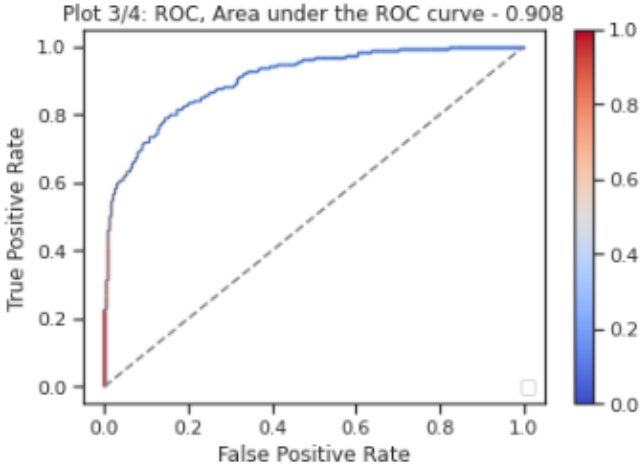


Imagen 94 - Diagrama ROC en training del dataframe dfTR_eval model df_join

Resultados de los modelos

Centrándonos en los resultados obtenidos, se puede observar que son satisfactorios en base a la similitud de las accuracies obtenidas en los conjuntos de training y test.

Sin embargo, en el modelo random forest aparentemente podría haber algún overfitting que se controló previamente utilizando el parámetro de ajuste max_depth igual a 5, además que el algoritmo de random forest es conocido por manejar el overfitting adecuadamente. El mejor rendimiento en el entrenamiento es el último modelo con todas las características, la unión de todas las variables hace que tenga un mejor rendimiento en el entrenamiento a pesar de tener un F-score menor que los demás, lo que se traduce en el número de registros ya que este conjunto de datos tiene más características, estos modelos obtienen un mejor rendimiento de generalización para un rendimiento similar durante el entrenamiento, esta mejora en la generalización se consigue compensando los errores de las predicciones de los diferentes árboles de decisión.

Así que para este proyecto estos modelos son parecidos, los valores entre ellos son casi iguales, esto demuestra que no fueron sobre entrenados y las características fueron seleccionadas con precisión.

Modelo	F-Score train	F-Score test	Accuracy CrossVal	Accuracy train	Accuracy test
Logistic Regression model df_rl	0.55	0.70	0.64	0.65	0.63
Logistic Regression model df_mr	0.60	0.70	0.66	0.67	0.65
Random Forest model df_rf	0.76	0.67	0.64	0.80	0.60
Logistic Regression model df_join	0.61	0.68	0.67	0.68	0.63
Random Forest model df_join	0.77	0.66	0.66	0.82	0.60

Comparación de modelos y rendimiento

5.6 DEPLOYMENT

Esta etapa el modelo, puede ser usado con nuevos datos y seguir la misma secuencia, nuevas interacciones en esta fase podrían revelar nuevas variables y necesidades para el conjunto de datos y el modelo. Estos nuevos retos podrían iniciar la revisión de las necesidades y acciones empresariales, o del modelo y los datos, o de ambos.

Los modelos aquí descritos y trabajados sirven para modelos de clasificación por lo cual en una posible implementación de estos se debe tener en cuenta el target a predecir que se ajuste a los conceptos de Lenguaje Supervisado y Machine Learning.

La metodología aquí expuesta puede cambiar de acuerdo con los requerimientos que se tengan y las fuentes de datos de donde provenga la información base para el modelo.

5.7 MODELOS ADICIONALES

Adicional al modelo previamente aquí expuesto de Regresión Logística, se hizo otro análisis de otras variables con Regresión Lineal, este es un modelo estadístico que trata de modelar la relación lineal entre una variable continua (variable dependiente) y una o más variables independientes (predictores) mediante el ajuste de una ecuación lineal.

Esto es que, en un conjunto de observaciones, la media μ de la variable respuesta Y se relaciona de forma lineal con la o las variables regresoras X según ecuación:

$$\mu Y=\beta 0+\beta 1x1+\beta 2x2+\ldots+\beta pxp$$

Las variables son las siguientes las cuales ya fueron definidas previamente:

SCLR_GSI_t2: SCLR es un cuestionario ampliamente utilizado para determinar una serie de síntomas psicológicos.

HAMD_sum_t2: es un cuestionario de múltiples ítems que se utiliza para proporcionar una indicación de la depresión, y como guía para evaluar la recuperación.

GAF_t2: escala utilizada para la planificación del tratamiento y la evaluación de los resultados en el Eje V del sistema de evaluación multiaxial del DSM-IV-TR

WHOQOL_psych_t2: es una evaluación de la calidad de vida.

Los mejores resultados fueron con WHOQOL_psych_t2 y SCLR_GSI_t2, estas variables son muy importantes en tratamientos psiquiátricos puesto son pruebas ampliamente utilizadas que ayudan al diagnóstico de pacientes, particularmente WHOQOL_psych_t2 mide factores psicológicos (ver imagen 11) tales como sentimientos positivos, negativos, concentración, autoestima, etc.

En este modelo se tuvo en cuenta la imputación de valores nulos, con el fin de obtener un mejor modelo con las mejores variables representativas, los valores nulos o faltantes son un

problema común en investigaciones psiquiátricas la técnica MICE (Multiple Imputation by Chained Equation) se ha convertido en un método muy efectivo para los valores faltantes, creando múltiples imputaciones a diferencia de las imputaciones únicas, tiene en cuenta la incertidumbre estadística de las imputaciones. Además, esta técnica es muy flexible y puede manejar variables de distintos tipos, este modelo asume que los datos faltantes son MAR (Missing at Random) es decir que los datos faltantes no están relacionados con los datos faltantes sino por los datos observados. *(Azur, M. J., Stuart, E. A., Frangakis, C., & Leaf, P. J. (2011), 2011)*

La implementación se hace también con el paquete *Scikit learn* y el módulo *sklearn.impute. IterativeImputer*, el cual es un imputador multivariante que estima cada característica a partir de todas las demás una estrategia de imputación de valores perdidos mediante la modelización de cada característica con valores perdidos en función de otras características de forma rotatoria. Esta técnica se utiliza junto con un modelo de regresión lineal para imputar los datos faltantes en cada columna del dataframe.

Es necesario los siguientes imports

```
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
```

Los parámetros más importantes son max_iter, que es el número de iteraciones que hace el modelo para devolver el valor imputado por defecto es 10, imputation_order es el orden en que se imputan los valores, ‘roman’ indica de izquierda a derecha.

min_value y max_value son los valores mínimos y máximos que puede tener el valor imputado muy importantes estos dos parámetros, ya que si no se configuran se imputa cualquier valor aleatorio. En este punto se ingresaron los valores mínimos y máximos que tiene el Dataframe con las funciones de Excel del archivo csv y se ingresaron en el notebook.

```
imp = IterativeImputer(estimator=lr,missing_values=np.nan,
max_iter=10,verbose=2,imputation_order='roman',random_state=0
,min_value=[], max_value=[])

df_int=imp.fit_transform(df_int)
```

```
[IterativeImputer] Completing matrix with shape (897, 319)
[IterativeImputer] Ending imputation round 1/10, elapsed time 14.29
[IterativeImputer] Change: 16018.45069103675, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 2/10, elapsed time 32.84
[IterativeImputer] Change: 9115.140579888108, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 3/10, elapsed time 46.22
[IterativeImputer] Change: 8800.451253874113, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 4/10, elapsed time 59.43
[IterativeImputer] Change: 7137.294677119379, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 5/10, elapsed time 72.57
[IterativeImputer] Change: 4826.124610333263, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 6/10, elapsed time 85.69
[IterativeImputer] Change: 2152.6081754613365, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 7/10, elapsed time 98.75
[IterativeImputer] Change: 3008.045306589331, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 8/10, elapsed time 112.96
[IterativeImputer] Change: 1956.7253356375286, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 9/10, elapsed time 126.03
[IterativeImputer] Change: 1263.4821923361744, scaled tolerance: 21.818
[IterativeImputer] Ending imputation round 10/10, elapsed time 139.20
[IterativeImputer] Change: 1972.9917984240026, scaled tolerance: 21.818
```

Imagen 95- Salida del IterativeImputer con cada iteración

El resultado de lo anterior es un array el cual se convierte a Dataframe con la opción de Pandas `Dataframe()` igual que en el procedimiento anterior se ejecuta en cada Dataframe con valores enteros y decimales y después se unen para formar el Dataframe *df_concat* el cual viene sin valores nulos.

Estos fueron los dataframes creados para cada variable que dieron los mejores resultados, para estos modelos se tuvo en cuenta la variable group_unipolar la cual indica pacientes con desorden depresivo de manera que el algoritmo será ejecutado solo con esta población. Para ello se filtra `df_bh['group_unipolar'] == 1`, el número 1 indica pacientes con desorden depresivo.

`WHOQOL_psych_t2:`

```
df_mrlr = df_concat[['WHOQOL_psych_t2', 'D2_F_RW1', 'Lithium',
'TMTA2', 'weight_min_age', 'WHOQOL_phys_t2', 'IL6457_t2', 'ZTMTA2',
'SCLR_DEP_t2N', 'Pos_zTMTA2', 'BDI_sum_t2R', 'TMTB2']]
```

`SCLR_GSI_t2:`

```
df_rflr = df_concat[['SCLR_GSI_t2', 'SCLR_OBS_t2', 'SCLR_PST_t2',
'SCLR_DEP_t2',   'SCLR_ANX_t2',   'SCLR_PSY_t2',   'SCLR_PSDI_t2',
'SCLR_PAR_t2', 'SCLR_INT_t2', 'SCLR_PHOB_t2']]
```

Definición de las Variables

A continuación, se definen las variables de cada dataframe, algunas ya fueron definidas en la etapa de entendimiento del negocio:

`WHOQOL_psych_t2`: es el test de calidad de vida de la OMS en salud psicológica en la salida del paciente.

`D2_F_RW1`: número de errores en admisión del paciente.

`Lithium`: lithium o litio. Las sales de litio suelen emplearse en el tratamiento del trastorno bipolar, tiene dos valores 0 no, 1 sí.

`TMTA2`: es el puntaje del test Trail Making parte A al salir el paciente, esta prueba consiste en conectar los puntos que forma parte de la Batería Neuropsicológica Halstead-Reitan.

`weight_min_age`: es el peso mínimo y la edad en años.

`WHOQOL_phys_t2:` es el test de calidad de vida de la OMS en salud física en la salida del paciente.

`IL6457_t2`: IL-6 (Interleucina-6) es una glucoproteína secretada por los macrófagos, el valor corresponde a la salida del paciente.

`ZTMTA2`: z-score del test Trail Making parte A

`SCLR_DEP_t2N`: valor estándar del cuestionario Symptom-Check-List-90-Standard – ítem depresión.

`BDI_sum_t2R`: es el puntaje del cuestionario Beck Depression Inventory

`TMTB2`: es el puntaje del test Trail Making parte B al salir el paciente, esta prueba consiste en conectar los puntos que forma parte de la Batería Neuropsicológica Halstead-Reitan

`SCLR_GSI_t2`: Symptom-Check-List-90-Standard - GSI (Global severity index) SCLR es un cuestionario ampliamente utilizado para determinar una serie de síntomas psicológicos Global Severity Index (GSI) está diseñado para ayudar a cuantificar la gravedad de la enfermedad de un paciente.

`SCLR_OBS_t2`: cuestionario SCLR escala obsesiones y compulsiones.

`SCLR_PST_t2`: cuestionario SCLR ítem síntomas positivos.

`SCLR_DEP_t2`: cuestionario SCLR escala depresión.

`SCLR_ANX_t2`: cuestionario SCLR escala ansiedad.

`SCLR_PSY_t2`: cuestionario SCLR escala psicoticismo.

`SCLR_PSDI_t2`: Symptom-Check-List-90-Standard - SDI (Positive Symptom Distress Index) ayuda a medir la intensidad de los síntomas.

`SCLR_PAR_t2`: cuestionario SCLR escala paranoia.

`SCLR_INT_t2`: cuestionario SCLR escala sensitividad interpersonal.

`SCLR_PHOB_t2`: cuestionario SCLR escala fobia.

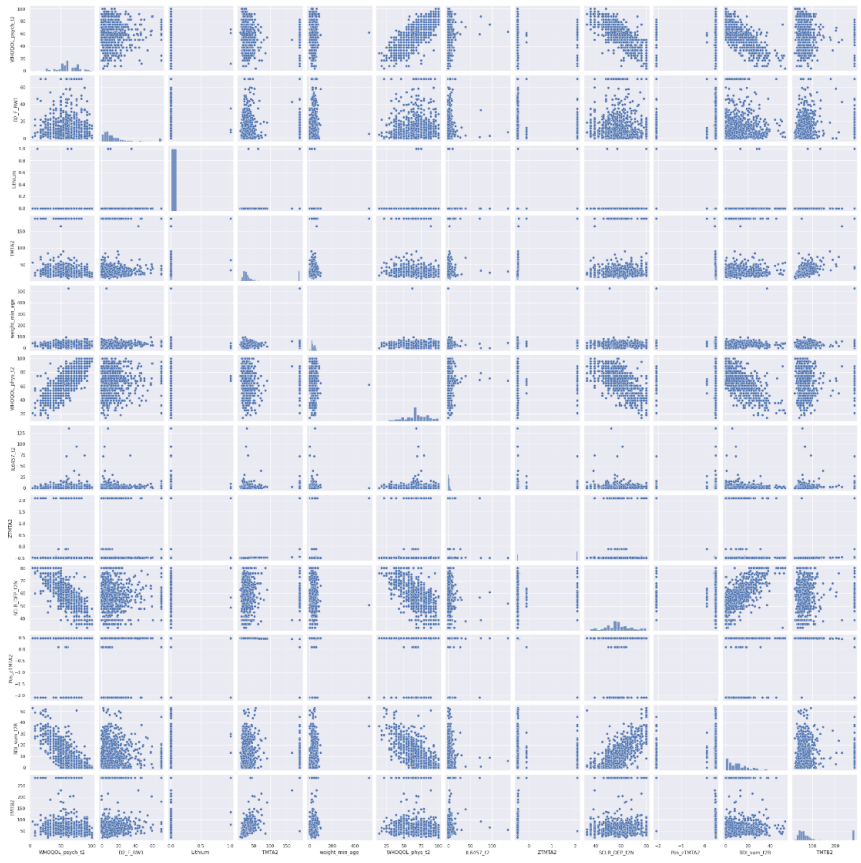


Imagen 96 - Scatter plot dataframe df_mrlr WHOQOL_psych_t2

La anterior imagen muestra la relación entre el target y el resto de variables algunas como *Lithium* muestra un desbalanceo hacia el valor 0, es una variable binaria, relación lineal entre WHOQOL_phys_t2 y el target lo cual tiene sentido puesto las dos variables miden calidad de vida, *phys* hace referencia a valores físicos (energía. Movilidad, capacidad de trabajo) el target es hacia lo psíquico (auto estima, sentimientos positivos y negativos), relación inversa entre el target y SCLR_DEP esta última es el test SCLSR en el item de depresión (DEP) entre más alto este test el paciente tiene problemas de depresión, el índice de calidad de vida WHOQOL_psych_t2 será más bajo.

Imagen 97 - Scatter plot dataframe df_rflr SCLR_GSI_t2

Estas variables son parecidas entre si puesto cada SCLR es un cuestionario para determinar síntomas psicológicos con sus respectivos ítems, ejemplo estas variables con sus ítems.

SCLR_OBS_t2	Obsesiones y compulsiones(OBS)
SCLR_DEP_t2	Depresión(DEP)
SCLR_ANX_t2	Ansiedad(ANX)
SCLR_PSY_t2	Psicoticismo(PSY)
SCLR_PAR_t2	Ideación paranoide(PAR)
SCLR_INT_t2	Sensitividad interpersonal(INT)
SCLR_PHOB_t2	Ansiedad fóbica(PHOB)

En los modelos de regresión lineal los outliers tienden a condicionar los modelos debido a pueden ser de alto impacto que pueden tener en los residuos, estos valores puede ser datos errados o simplemente valores anormales dentro de una muestra pero que son reales por lo tanto no se pueden eliminar a la ligera.

En este proyecto se tiene en cuenta el comportamiento de los residuos y las métricas de regresión lineal como R2, en las pruebas iniciales se hicieron modelos con todos los datos pero los residuos mostraron resultados dispares, los residuos son la diferencia entre los valores observados y los valores que predice el modelo de ahí su importancia en regresión lineal.

Uno de los supuestos de los residuos es que se distribuyan alrededor de la recta de regresión, en forma normal con media 0, varianza constante y los outliers que podrían alejarse de la recta, es por eso para este modelo se eliminan outliers.

Para ello se toma el método del IQR o rango intercuartil, es decir la diferencia entre el tercer y el primer cuartil (puntos tomados a intervalos regulares de la función de distribución *Q1, Q2, Q3*) de cada distribución.

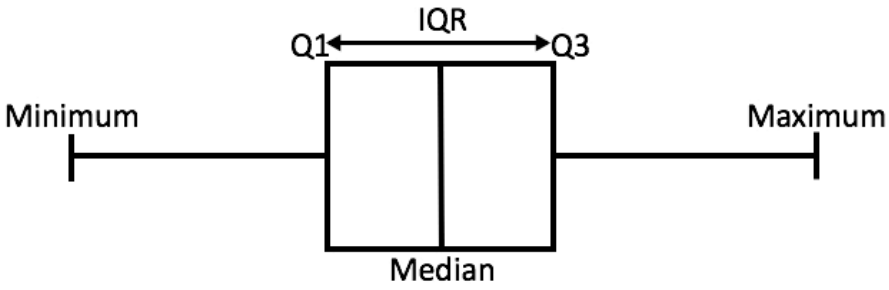


Imagen 98 - Boxplot con los rangos intercuartílicos

En la imagen 98 se indica como un diagrama de caja o boxplot muestra la distribución de datos, donde hay dos extremos mínimo y máximo de cada observación del dataset y la diferencia entre los dos es el rango de cada variable.

- Q1: es el primer cuartil de los datos o sea el 25% de los datos están entre el mínimo y Q1
- Q3: es el tercer cuartil de los datos o sea el 75% de los datos están entre el mínimo y Q3. La diferencia es el rango intercuartílico o IQR, *IQR = Q3 – Q1*

Implementacion en Python:

```
Q1 = df_mrlr.quantile(0.25)

Q3 = df_mrlr.quantile(0.75)

IQR = Q3 - Q1

df_mrlr = df_mrlr[~((df_mrlr < (Q1 - 1.5 * IQR)) |(df_mrlr > (Q3 +
1.5 * IQR))).any(axis=1)]
```

Se utiliza 1.5 en la fórmula porque una escala mayor daría como resultado que los valores atípicos se perciban como puntos de datos, mientras que una escala más pequeña daría como resultado que algunos de los puntos de datos se perciban como valores atípicos.

De igual forma como en los modelos anteriores se crea conjunto de train y test en proporción 80 – 20, tal como antes los datos se validan con Cross Validation, con k= 10 folds, se hace un preprocesado con StandardScaler() para reescalar los datos, y se utiliza el modelo LinearRegression() de Scikit-learn.

Para el modelo del target WHOQOL_psych_t2 (calidad de vida psíquica) algunas variables fueron excluidas del modelo por multicolinealidad y por IQR. La multicolinealidad es la relación de dependencia lineal fuerte entre más de dos variables explicativas en una regresión múltiple que incumple el supuesto de Gauss-Markov cuando es exacta. (Paula Rodó, 2019)

Es decir, la multicolinealidad es la correlación alta entre más de dos variables explicativas. Si existe multicolinealidad muy alta > 10 dicha variable no aporta mucha información modelo, puede ser explicada por otras variables, por lo tanto, se utiliza el VIF (Variance Inflation Factor) o factor de inflación de la varianza, por regla este valor no debe ser mayor a 10 de lo contrario el modelo es inestable y no queda bien estimado.

Variance Inflation Factor: $VIF(\beta_i) = \frac{1}{1 - R_i^2}$ donde R_i^2 es el coeficiente de determinación resultante de la regresión X_i de los $n - 1$ regresores restantes.

Las variables `TMTA2, ZTMTA2, Pos_zTMTA2` no fueron incluidas por VIF y `Lithium` por IQR. En la imagen 99 se aprecia el VIF de las variables restantes.

	VIF Factor	features
0	1.1	D2_F_RW1
1	1.0	weight_min_age
2	2.3	WHOQOL_phys_t2
3	1.0	IL6457_t2
4	2.5	SCLR_DEP_t2N
5	2.5	BDI_sum_t2R
6	1.1	TMTB2

Imagen 99- Factor VIF modelo regresión lineal target WHOQOL_psych_t2

El resumen del modelo indica algunas cosas importantes en este caso el valor R-squared o R2 o coeficiente de determinación entre más cercano a 1 es mejor el valor fue de 0.754, este coeficiente es la proporción de la varianza total de la variable explicada por la regresión. Es importante saber que mientras más cerca de 1 mayor será el ajuste del modelo a la variable que estamos intentando explicar. Cuanto más cerca a cero menos ajustado estará el modelo por tanto, el modelo es menos fiable. En este caso el modelo explica el 75.4% de la varianza de los datos.

El error estándar o desviación estándar es parecido entre las muestras lo que indica una distribución más normal sin tanto valor atípico que puede causar varianza muy alta, las variables significativas que tienen p-value < 0.05 son la mayoría 5 de 7 variables lo que indica rechazar la hipótesis nula y que existe relación entre el target y estas variables.

El coeficiente R cuadrado ajustado es la medida que define el porcentaje explicado por la varianza de la regresión en relación con la varianza de la variable explicada. Es decir, penaliza la inclusión de variables suele ser menor al R cuadrado en este caso 0.751, entre más números de variables explicativas más alejado estará R cuadrado ajustado del R cuadrado normal.

Con respecto a los residuos de este modelo la imagen 101 los muestra por variables, los residuos en regresión lineal son las diferencias entre los valores de la variable dependiente

observados y los valores que se predicen a partir de la recta de regresión, los residuos deben distribuirse alrededor de la recta de regresión en forma normal y con media 0.

El histograma muestra una distribución normal, en todas las variables los residuos están en torno la recta de regresión, este mismo modelo se probó previamente incluyendo outliers y los resultados fueron dispares con una heterocedasticidad muy alta.

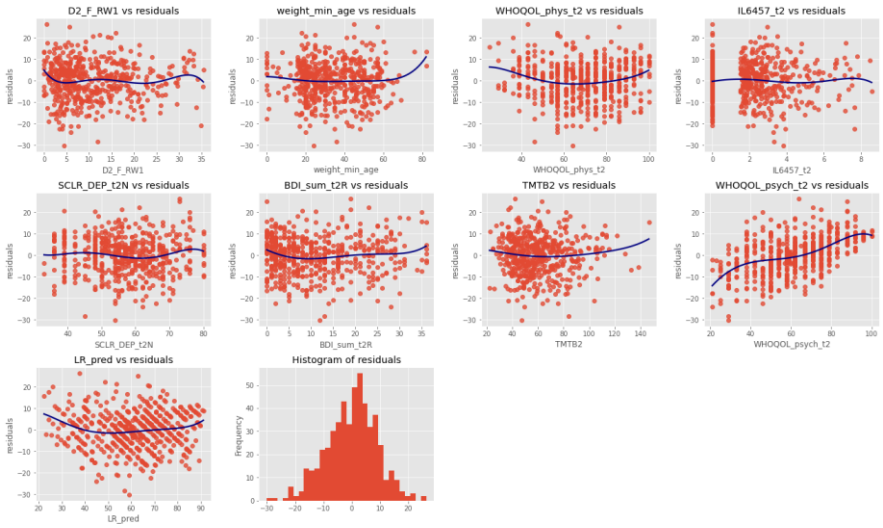


Imagen 100- Gráfico de residuos modelo regresión lineal target WHOQOL_psych_t2

En el caso del Dataframe `df_rflr` con el target `SCLR_GSI_t2` se ejecuta de la misma forma, debido a multicolinealidad alta la variable `SCLR_PST_t2` el resto de las variables tiene VIF < 5 lo cual permite un buen modelo

El modelo tiene un R cuadrado alto de 0.973 y donde todas las variables utilizadas con p-values < 0.05

```
Residuals:
        Min         1Q     Median         3Q        Max
0 -0.331189 -0.029824 -0.001613  0.031013  0.287551

Coefficients:
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.973
Model:                            OLS   Adj. R-squared:                  0.973
Method:                 Least Squares   F-statistic:                     2388.
Date:                Mon, 12 Sep 2022   Prob (F-statistic):               0.00
Time:                        12:09:58   Log-Likelihood:                778.88
No. Observations:                 541   AIC:                            -1540.
Df Residuals:                     532   BIC:                            -1501.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.4829      0.002    194.211      0.000      0.478      0.488
SCLR_OBS_t2    0.0732      0.005     15.713      0.000      0.064      0.082
SCLR_DEP_t2    0.0861      0.005     16.457      0.000      0.076      0.096
SCLR_ANX_t2    0.0764      0.004     18.059      0.000      0.068      0.085
SCLR_PSY_t2    0.0397      0.004     10.589      0.000      0.032      0.047
SCLR_PSDI_t2   0.0283      0.004      7.439      0.000      0.021      0.036
SCLR_PAR_t2    0.0266      0.004      7.406      0.000      0.020      0.034
SCLR_INT_t2    0.0548      0.004     12.749      0.000      0.046      0.063
SCLR_PHOB_t2   0.0238      0.003      6.993      0.000      0.017      0.030
==============================================================================
Omnibus:                       62.804   Durbin-Watson:                   1.931
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              465.124
Skew:                           0.076   Prob(JB):                    9.99e-102
Kurtosis:                       7.540   Cond. No.                         5.64
==============================================================================
```

Imagen 101 - Resumen del dataframe df_rflr, target SCLR_GSI_t2

En general los residuos están con media 0, varianza constante, se presentan algunas cruvas que pueden ser causadas por outliers en SCLR_DEP_t2 y SCLR_ANX_t2 (depresión y ansiedad respectivamente) y el histograma de residuos tiene una distribución normal.

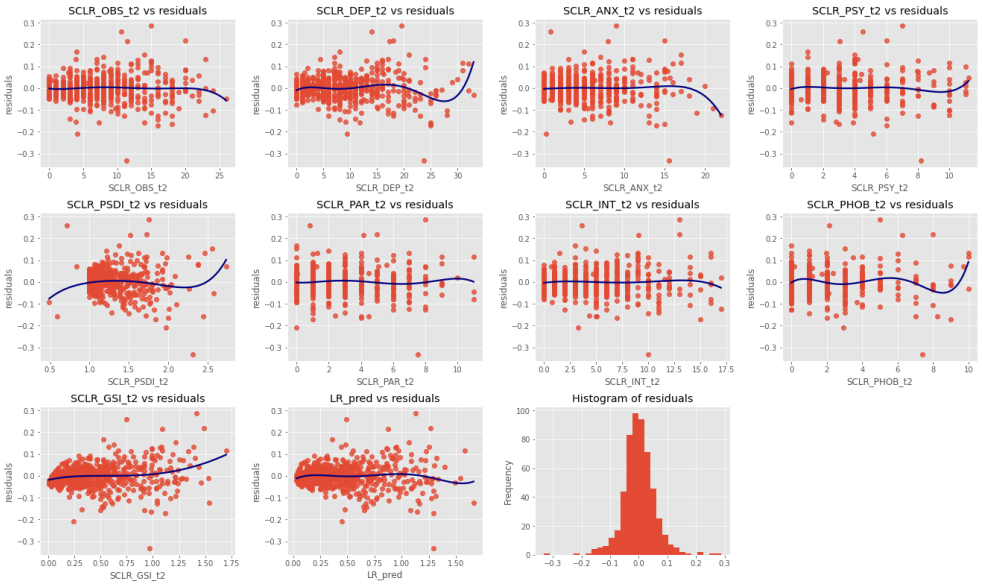


Imagen 102 - Gráfico de residuos modelo regresión lineal target SCLR_GSI_t2

Capítulo 6. ANÁLISIS DE RESULTADOS

Se implementaron y analizaron varios modelos de Machine Learning con distintos métodos de extracción de características con el fin de buscar la mejor solución posible. Los proyectos de Machine Learning requieren un flujo de trabajo por etapas el cual aquí se hizo, cada etapa puede tener pocas o varias subtareas todo depende del target y de las fuentes de trabajo. La interpretación de las métricas de precisión de las predicciones (por ejemplo, como alta o baja precisión) depende del contexto y requiere experiencia en el campo dado que se tienen muchas variables y puntos de referencia que los psicólogos utilizan y que juntas permiten establecer un diagnóstico del paciente.

Es por eso por lo que se implementaron varios modelos para tener varias opciones de trabajo, estos modelos tienen ventajas y desventajas como todo, pero hoy en día hay que tener en cuenta el soporte de ayudas en temas donde se puedan presentar inconvenientes es por eso que utilizando Python y sus librerías más importantes para el Data Science se obtienen buenos beneficios por la documentación amplia, ejemplos, material, referencias, etc.

Entre las ventajas de los modelos aquí trabajados son:

- La regresión logística es fácil de implementar, interpretar y muy eficiente al entrenar.
- Buena precisión para muchos conjuntos de datos sencillos y funciona bien cuando el conjunto de datos es linealmente separable.
- Los bosques aleatorios o Random Forest solucionan el problema overfitting, ya que el resultado se basa en la votación por mayoría o en el promedio.

Con los resultados de los modelos y las variables utilizadas, los psicólogos pueden ofrecer una mejor interpretación de síntomas y tratamientos a los pacientes, variables como California Verbal Learning Test - CVLT, Quality of Life - WHOQOL, Trail Making Test – TMTA son importantes en el diagnóstico.

En el caso del modelo Random Forest algunas de las variables más importantes son MCC320_t1, LDH357_t1, ERY316_t1, age, igualmente para regresión logística algunas de las más importantes son ZCVLT_LS_DG1_5_RW1, HBE318_t1, HKT322_t1, spg419_t1, esta importancia de variables en conjunto es un adicional para los profesionales en Psicología es una ayuda más en su diagnóstico y tratamiento a pacientes.

La precisión general de los modelos utilizando el conjunto de datos de entrenamiento y de prueba es similar entre ellos, siendo Random Forest el mejor en training. En general, la matriz de confusión ayuda a mejorar el rendimiento de los modelos de clasificación de aprendizaje automático.

De acuerdo con los resultados obtenidos los modelos mantienen los valores similares entre sí, es un modelo óptimo puesto se intentaron varios caminos y los resultados son parecidos, es decir los algoritmos ejecutados llevan a resultados parecidos.

Estos resultados son muy importantes puesto el dataset contiene mucha información de tests y otras pruebas psicológicas que en conjunto determinan el tratamiento a un paciente, muy importante la solución aquí expuesta puesto los datos del dataset son personales son más objetivos que generales por lo tanto la solución es más personal que general, habitualmente la investigación psicológica apunta a establecer los efectos causales de las variables predictoras, sobre las variables de resultado en nuestro caso por ejemplo Global severity index o Global Assessment of Functioning es decir por que un paciente obtuvo tal resultado o por que un paciente con determinada edad u otro factor obtuvo este otro resultado y los algoritmos de machine learning tienen muchas formas de medir su precisión es por esto que es indispensable contar con un equipo interdisciplinario (psicólogos, data scientists) para lograr un tratamiento adecuado y óptimo para cada paciente y mejorar los resultados de los algoritmos de manera que en una próxima iteración con nuevos datos se pueda mejorar los resultados y buscar la relación óptima entre algoritmos y datasets clínicos.

Cada dataframe mediante los métodos de features selection tuvieron distintas variables algunas comunes para los tres modelos y son estas HKT322_t1, ERY316_t1, LDH357_t1. Estas variables corresponden a conceptos somáticos y son importantes para el diagnóstico del paciente donde finalmente todas las variables son significativas para el profesional médico, pero en este trabajo y con las técnicas realizadas se selecciona un número reducido para los modelos. HKT322_t1 es el nivel de hematocritos en la sangre, la hemoconcentración por stress por causa del stress psicológico podría ser un factor de riesgo para enfermedades cardiovasculares, es probable que un paciente con enfermedad major depressive disorder y alto nivel de hematocritos pueda desarrollar enfermedad cardiovascular. Los niveles de hematocritos pueden variar según edad y sexo en hombres adultos el rango esta entre 41%-50% y en mujeres 36% - 44%. (*Wong, M. L., Dong, C., Esposito, K., Thakur, S., Liu, W., Elashoff, R. M., & Licinio, J., 2008*)

El índice de eritrocitos (ERY316_t1) en la sangre es importante en personas con trastornos depresivos la inflamación y la desregulación vascular pueden contribuir al desarrollo de la depresión e imponer una carga a la eritropoyesis (formación de glóbulos rojos en el tejido que compone la sangre), en estos casos y para un estudio a fondo es necesario examinar los valores y cambios de los niveles del volumen corpuscular medio y concentración de hemoglobina corpuscular media, es posible que los cambios en los eritrocitos pueden estar asociados con el riesgo de depresión en los adultos mayores. *(Dae Jong Oh, Ji Won Han, Jong Bin Bae, Hye Sung Kim, Seung Wan Suh, Seonjeong Byun, Tae Hui Kim, Kyung Phil Kwak, Bong Jo Kim, Shin Gyeom Kim, Jeong Lan Kim, Seok Woo Moon, Joon Hyuk Park, Seung-Ho Ryu, Jong Chul Youn, Dong Young Lee, Dong Woo Lee, Seok B, 2020)*

El nivel de Deshidrogenasa láctica (LDH357_t1) también es muy importante en el diagnóstico, pacientes que han padecido algún accidente cerebrovascular suelen padecer PSD (Post-stroke depression) o depresión después de un accidente cerebrovascular. Los niveles de deshidrogenasa láctica (LDH) están aumentados en pacientes con trastornos del sistema nervioso central (SNC), como el infarto cerebral y la encefalopatía hipóxico-isquémica, lo que puede estar relacionado con la aparición de PSD (Post-stroke depression)

en pacientes con ictus isquémico agudo (AIS). *(Li, G., Miao, J., Pan, C., Jing, P., Chen, G., Mei, J., Sun, W., Lan, Y., Zhao, X., Qiu, X., Wang, Y., Zhu, Z., Zhu, S., & Lian, L, 2021)*

En el caso del modelo df_rf con el cual se ejecutó Random Forest las cuatro primeras variables más importantes son `MCC320_t1`, `LDH357_t1`, `ERY316_t1` y `age` (ver imagen 70). Estas son variables somáticas, MCC320_t1 es la concentración media de hemoglobina corpuscalar, la insuficiencia de hemoglobina y la depresión comparten varios síntomas y suelen darse en los mismos pacientes, es probable que pacientes con una baja MCC pueden ser propensos a desarrollar síntomas depresivos. *(Lee, J. M., Nadimpalli, S. B., Yoon, J. H., Mun, S. Y., Suh, I., & Kim, H. C., 2017)*

La edad del paciente también es importante, en este estudio el promedio es de 52 años donde todos los pacientes son adultos, normalmente los síntomas depresivos empiezan entre los 30 y 40 años que es cuando la gente está en plena etapa productiva y familiar.

En general las variables del modelo df_rf son somáticas (tiroxina, edad, BMI) estas variables somáticas junto con las pruebas y resultados de todos los tests son esenciales para el diagnóstico puesto la depresión y los trastornos depresivos finalmente es una enfermedad, si la depresión psicológica no es tratada de forma adecuada los factores somáticos pueden empeorar o viceversa, la OMS en la Clasificación Internacional de Enfermedades CIE10 distingue la depresión según su gravedad en tres episodios leve, moderado y grave. El episodio moderado puede tener síntomas somáticos y mentales que impiden al paciente hacer sus actividades ordinarias. *(World Health Organization, 2019)*

En los otros datasets es importante subrayar en el feature selection, el equilibrio que hay entre variables somáticas y cognitivas puesto la importancia del puntaje en el test California verbal Learning o el Verbal Memory se juntan con el nivel de eritrocitos (`ERY316_t1`) o el total de hemoglobina (`HBE318_t1`) , la variable `rehab_weeks_4_6` que indica si el paciente se recupera en 4 o más de 6 semanas también es muy importante en el tiempo que el paciente se recupera y lo bien que responde al tratamiento de acuerdo también a los síntomas y pruebas realizadas.

Otras variables importantes son el test Aprendizaje Verbal de California (ZCVLT_LS_DG1_5_RW1) este es bastante útil en temas cognitivos contribuye al diagnóstico y tratamiento de alteraciones de la memoria, estas alteraciones pueden se originadas por trastornos psiquiátricos o neurológicos, los tests de calidad de vida de la OMS (WHOQOL) son importantes para el diagnóstico ya que evalúa la percepción del sujeto en sus dimensiones física, psicológica, social y todas las escalas que tiene esta prueba.

Las variables de la función ejecutiva (RZExecut_2, NZExecut_2, RZExcecu) también son importantes puesto las funciones ejecutivas son aquellas actividades mentales complejas necesarias para adaptarse al entorno y alcanzar metas es decir son el conjunto de capacidades cognitivas para controlar la propia conducta. Las funciones son varias: flexibilidad cognitiva, inhibición, planificación, memoria de trabajo, autoregulación esta función permite tener la habilidad de gestionar las emociones, el estado de ánimo y la motivación con el fin de lograr los objetivos. Las afectaciones de estas funciones tienen como resultado algunos trastornos tales como Alzheimer, intoxicaciones (drogadicción, alcoholismo) y TDAH.

Estas funciones se pueden evaluar con el Trail Making Test o con pruebas para la memoria verbal dichas variables (RZVerbal, RZVerbal_2) ayudan en conjunto al diagnostico de las funciones ejecutivas, la memoria verbal es aquella que guarda la información en forma de palabras, esta memoria puede verse afectada precisamente por trastornos depresivos o consumo de sustancias.

La prueba Tiroxina libre – T4 libre (*FT4402_t1*) ayuda a diagnosticar enfermedades de la tiroides, la tiroides produce hormonas que controlan la manera en que el cuerpo utiliza la energía incluso el estado de ánimo, algunos estudios sugieren que las alteraciones de esta glándula es relevante para la fisiopatología y el curso clínico del trastorno afectivo bipolar donde incluso las sales de litio (*Lithium*) ayuda a controlar la tiroides, las hormonas tiroideas (T4 y T3) se ocupan fundamentalmente de regular las acciones metabólicas, siendo imprescindibles para que se realicen la mayoría de las funciones del organismo. Por ello, el hipotiroidismo se caracteriza por una disminución global de la actividad orgánica:

metabólica, neuronal, cardiocirculatoria, digestiva, etc. Cuando los niveles de hormonas tiroideas disminuyen, como ocurre en el hipotiroidismo, la secreción de TSH (tirotropina) aumenta para intentar estimular la función tiroidea. *(Sierra,Pilar, Cámara,Rosa, Tobella,Helena, Livianos,Lorenzo, 2014)*

Los Trombocitos o plaquetas (*THR323_t1*) son aquellos componentes de la sangre que ayudan en la cicatrización cuyo nivel bajo puede ocasionar hemorragias o el exceso de estas puede ocasionar trombosis. En el ámbito psiquiátrico hay relación entre plaquetas y el cerebro, las plaquetas contienen grandes cantidades de serotonina y una disfunción del sistema serotoninérgico está implicada en el desarrollo de varios trastornos del comportamiento, como la depresión, los trastornos de ansiedad y las alteraciones autoagresivas, las anomalías plaquetarias en los pacientes deprimidos se encuentran principalmente en el sistema serotoninérgico y noradrenérgico. La serotonina interviene en el afecto positivo y el estado de ánimo en el cerebro sano, por lo que el equilibrio del metabolismo de la serotonina puede desempeñar un papel fundamental en la patogénesis de la depresión, aquí su importancia en este estudio. *(Ehrlich, D., & Humpel, C., 2012)*

La creatina quinasa es una enzima una enzima expresada por varios tejidos y tipos celulares la cual también está presente en el cerebro, esta desempeña un papel fundamental en el metabolismo de los tejidos que consumen mucha energía, como el cerebro. Dicha variable (*CK355_t1*) puede considerarse un marcador de rasgo en la depresión mayor. Los valores anormales o la detección simultánea de creatina quinasa y lactato deshidrogenasa (*LDH357_t1*) puede ser útil como indicador de los síntomas depresivos esto debido a la función que ejerce en el cerebro. *(Kato, A., Sakakibara, H., Tsuboi, H, 2014)*

Respecto a la calidad de vida WHOQOL_psych_t2 en el nivel psíquico las variables que mejor predicen el modelo de regresión lineal son las relacionadas son SCLR_DEP_t2N la cual es el cuestionario SCLR en de nivel depresión tiene sentido puesto la depresión y el nivel de vida psíquico están relacionados estas herramientas o tests en conjunto ayudan al profesional a una mejor estimación psiquiátrica del paciente , el Trail Making Test versión

B (TMTB) también es importante puesto mediante este test se mide el deterioro cognitivo del paciente, el cuestionario BDI_sum_t2R que mide gravedad de los síntomas depresivos en adolescentes y adultos también es importante para el test de calidad de vida WHOQOL_psych_t2, incluso el peso del paciente la obesidad es una patología influyente en pacientes depresivos, la interleucina-6 (*IL6457_t2)* es una citocina proinflamatoria que está presente en el plasma de personas (plasma es el líquido que transporta los componentes de la sangre por todo el cuerpo) con depresión también es importante para el estudio. (Hernández Martínez, A., Valencia Ortiz, A. I., García Cruz, R., Fernández Martínez, E., & Ortega Andrade, N. A., 2022)

El cuestionario SCLR - Global Severity Index mediante el modelo realizado tiene importancia las variables relacionadas entre si, pero no son multicolineales, dado que el test SCLR tiene varios ítems indicados previamente el modelo tiene un R cuadrado alto debido a la relación en estas variables. El profesional médico puede tener la referencia o guía en el test con ítems DEP e INT, depresión y Sensitividad interpersonal respectivamente son importantes en el modelo y en diagnóstico de pacientes.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Se puede concluir que el uso de herramientas de Machine Learning en conjunto con la ciencia médica ayuda en el tratamiento y diagnóstico en tratamientos a pacientes.

El trabajo aquí expuesto no es determinante a la hora de tratar al paciente, pero sí ayuda en algunas tareas de tomas de decisión, síntomas e importancia de atributos de cada paciente, a futuro la fuente de datos de este trabajo puede ser cambiada por otro dataset, se pueden intentar predecir otras features que se consideren importantes para tratamiento.

A futuro también se podría implementar un sistema de información con todo el pipeline aquí descrito de manera complemente los sistemas de historias clínicas o intentar crear una API a modo de servicio para que sea llamada en otros sistemas de información que pueden ser on-cloud.

Las variables más importantes utilizadas en los modelos sirven como ayuda y base para un mejor diagnóstico del paciente, a futuro también puede ser importante la relación entre imágenes de pacientes con trastorno depresivo, pacientes sanos y pacientes tratados, con el uso del Deep Learning se podría hacer análisis visual de encefalografías de pacientes y determinar diferencias, similitudes hacer un estudio completo entres los tests e imágenes diagnósticas.

Capítulo 8. BIBLIOGRAFÍA

American Psychological Association. (2020, June). *apa.org*. Retrieved from https://www.apa.org/pi/about/publications/caregivers/practice-settings/assessment/tools/beck-depression

American Psychological Association. (2022). *dictionary.apa.org*. Retrieved from https://dictionary.apa.org/concentration

American Psychological Association. (2022). *dictionary.apa.org*. Retrieved from https://dictionary.apa.org/california-verbal-learning-test

American Psychological Association. (2022). *dictionary.apa.org*. Retrieved from https://dictionary.apa.org/stroop-color-word-interference-test

American Psychological Association. (2022). *dictionary.apa.org*. Retrieved from https://dictionary.apa.org/trail-making-test

American Psychological Association. (2022). *dictionary.apa.org*. Retrieved from https://dictionary.apa.org/premorbid

American Psychological Association. (2022). *dictionary.apa.org*. Retrieved from https://dictionary.apa.org/global-assessment-of-functioning-scale

American Psychological Association. (2022). *Dictionary.apa.org*. Retrieved from https://dictionary.apa.org/body-mass-index

American Psychological Association. (2022). *Dictionary.apa.org*. Retrieved from https://dictionary.apa.org/blood-pressure

Azur, M. J., Stuart, E. A., Frangakis, C., & Leaf, P. J. (2011). (2011, Mar 20). *Multiple imputation by chained equations: what is it and how does it work?. International*

journal of methods in psychiatric research, 20(1),. Retrieved from https://doi.org/10.1002/mpr.329

Badawy, Abdulla A-B; Guillemin, Gilles . (2019, August 21). *Ncbi.nlm.nih.gov*. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6710706/

Bassuk, Shari S; Rifai, Nader; Ridker, Paul M. (2004, August 29). *Current problems in cardiology*. Retrieved from https://pubmed.ncbi.nlm.nih.gov/15258556/

Breiman, Leo; Cutle, Adele . (2022). *Stat.berkeley.edu*. Retrieved from https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

Brownlee, J. (2020, August 20). *machinelearningmastery.com*. Retrieved from https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

Centers for Disease Control and Prevention. (2020, January 31). *Cdc.gov*. Retrieved from https://www.cdc.gov/cholesterol/

Dae Jong Oh, Ji Won Han, Jong Bin Bae, Hye Sung Kim, Seung Wan Suh, Seonjeong Byun, Tae Hui Kim, Kyung Phil Kwak, Bong Jo Kim, Shin Gyeom Kim, Jeong Lan Kim, Seok Woo Moon, Joon Hyuk Park, Seung-Ho Ryu, Jong Chul Youn, Dong Young Lee, Dong Woo Lee, Seok B. (2020, november). *Erythrocyte Characteristics and the Risk of Depression in Late Life: A Population-Based Prospective Study*. Retrieved from https://www.sciencedirect.com/science/article/pii/S152586102030414X: https://doi.org/10.1016/j.jamda.2020.05.012

Denea, Denise. (2020, June 8). *Mdapp.co*. Retrieved from https://www.mdapp.co/waist-to-height-ratio-whtr-calculator-433/

Derogatis, Leonard R. (1993). *pearsonclinical.com.au*. Retrieved from https://www.pearsonclinical.com.au/products/view/224#:~:text=The%20Global%20Severity%20Index%20(GSI,based%20on%20reducing%20symptom%20severity

Dubey, Akash . (2018, December 15). *Towardsdatascience.com*. Retrieved from https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f

Ehrlich, D., & Humpel, C. (2012, december 22). *Platelets in psychiatric disorders*. Retrieved from World journal of psychiatry: https://doi.org/10.5498/wjp.v2.i6.91

Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirami. (2013). *An Introduction to Statistical Learning with Applications in R.* New York: Springer.

Google Inc. (n.d.). *colab.research.google.com*. Retrieved from https://colab.research.google.com/?hl=es

Healthline. (2020, April 1). *healthline.com*. Retrieved from https://www.healthline.com/health/affective-disorders

Hernández Martínez, A., Valencia Ortiz, A. I., García Cruz, R., Fernández Martínez, E., & Ortega Andrade, N. A. (2022, march 14). *Cognitive-behavioral intervention to modify ruminative responses and plasma IL-6 in college students with depressive symptomatology.* Retrieved from https://rus.ucf.edu.cu/index.php/rus/article/view/2717

Huo,Tianyao; Guo,Yi ; Shenkman, Elizabeth; Muller,Keith. (2018, February 13). *Assessing the reliability of the short form 12 (SF-12) health survey in adults with mental health conditions: a report from the wellness incentive and navigation (WIN) study. Health and quality of life outcomes*. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5811954/

IBM. (2021, August 17). *ibm.com*. Retrieved from https://www.ibm.com/docs/es/spss-modeler/saas?topic=dm-crisp-help-overview

IBM Cloud Education. (2020, December 7). *Ibm.com*. Retrieved from https://www.ibm.com/cloud/learn/random-forest

Kato, A., Sakakibara, H., Tsuboi, H. (2014, may 27). *Depressive symptoms of female nursing staff working in stressful environments and their association with serum creatine kinase and lactate dehydrogenase – a preliminary studyDepressive symptoms of female nursing staff working in stressful environments*. Retrieved from BioPsychoSocial Med: https://doi.org/10.1186/1751-0759-8-21

Lee, J. M., Nadimpalli, S. B., Yoon, J. H., Mun, S. Y., Suh, I., & Kim, H. C. (2017, march). *Association between Mean Corpuscular Hemoglobin Concentration and Future Depressive Symptoms in Women.* Retrieved from The Tohoku journal of experimental medicine: https://doi.org/10.1620/tjem.241.209

Li, G., Miao, J., Pan, C., Jing, P., Chen, G., Mei, J., Sun, W., Lan, Y., Zhao, X., Qiu, X., Wang, Y., Zhu, Z., Zhu, S., & Lian, L. (2021, december 9). *Higher Serum Lactic Dehydrogenase is Associated with Post-Stroke Depression at Discharge. Clinical interventions in aging*. Retrieved from Clinical interventions in aging: https://doi.org/10.2147/CIA.S341169

Mayoclinic. (2021, October 29). *mayoclinic.org*. Retrieved from https://www.mayoclinic.org/diseases-conditions/mood-disorders/symptoms-causes/syc-20365057

Mazzanti, Samuele . (2021, February 12). *Towardsdatascience.com*. Retrieved from https://towardsdatascience.com/mrmr-explained-exactly-how-you-wished-someone-explained-to-you-9cf4ed27458b

MedlinePlus. (2021, Febrero 25). *Medlineplus.gov*. Retrieved from https://medlineplus.gov/spanish/triglycerides.html#:~:text=Los%20triglicéridos%20son%20un%20tipo,también%20provienen%20de%20calorías%20adicionales.

Meskó, B., Görög, M. (202, September 24). *A short guide for medical professionals in the era of artificial intelligence.* Retrieved from https://doi.org/10.1038/s41746-020-00333-z

numpy.org. (2022). Retrieved from https://numpy.org/doc/stable/

Orrù, G., Monaro, M., Conversano, C., Gemignani, A., & Sartori, G. (2020, January 10). Retrieved from https://www.frontiersin.org/articles/10.3389/fpsyg.2019.02970/full

Pandas.org. (2022, Junio 23). *Pandas.org*. Retrieved from https://pandas.pydata.org

Paula Rodó. (18 de junio de 2019). *Multicolinealidad. Economipedia.com*.

scikit-learn. (2022, August). Retrieved from https://scikit-learn.org/stable/index.html

seaborn.pydata.org. (2022). Retrieved from https://seaborn.pydata.org

Sierra,Pilar, Cámara,Rosa, Tobella,Helena, Livianos,Lorenzo. (abril de 2014). *¿Cuál es la relevancia real y el manejo de las principales alteraciones tiroideas en los pacientes bipolares?* Obtenido de Revista de Psiquiatría y Salud Mental - Journal of Psychiatry and Mental Health: https://www.elsevier.es/es-revista-revista-psiquiatria-salud-mental--286-articulo-cual-es-relevancia-real-el-S1888989113000840

T. Hastie, R. T. (2009). *The Elements of Statistical Learning. Data Mining, Inference and Prediction. 2nd Ed.* . New York: Springer.

Tanaka, Toshio ; Narazaki, Masashi; Kishimoto, Tadamitsu. (2014, October 6). *Cold Spring Harbor perspectives in biology*. Retrieved from

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4176007/: https://doi.org/10.1101/cshperspect.a016295

The Python Wiki. (2018, September 16). Retrieved from www.wiki.python.org/moin/FrontPage

who. (2022, March 2). *World Health Organization.* Retrieved from who.int: https://www.who.int/news/item/02-03-2022-covid-19-pandemic-triggers-25-increase-in-prevalence-of-anxiety-and-depression-worldwide

Wikipedia.org. (2022). Retrieved from https://en.wikipedia.org/wiki/Hamilton_Rating_Scale_for_Depression

Wikipedia.org. (2022). Retrieved from https://en.wikipedia.org/wiki/Waist–hip_ratio

Wikipedia.org. (2022). Retrieved from https://en.wikipedia.org/wiki/Kynurenine

Wong, M. L., Dong, C., Esposito, K., Thakur, S., Liu, W., Elashoff, R. M., & Licinio, J. (2008, july 16). *Elevated stress-hemoconcentration in major depression is normalized by antidepressant treatment: secondary analysis from a randomized, double-blind clinical trial and relevance to cardiovascular disease risk.* Retrieved from https://doi.org/10.1371/journal.pone.0002350: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2391294/

World Health Organization. (2012, March 1). *Who.int*. Retrieved from https://www.who.int/tools/whoqol

World Health Organization. (2019). *https://icd.who.int*. Retrieved from https://icd.who.int/browse10/2019/en#/F32

ANEXO 1: CÓDIGO FUENTE DE MODELO REGRESIÓN LOGÍSTICA

```
pipe = Pipeline(steps=[('scaler',StandardScaler()), # Preproceso

                           ('LogReg',LogisticRegression(random_state=5))])#
Modelo

nFolds = 10

param = {}

LogReg_fit = GridSearchCV(estimator=pipe, #

                       param_grid=param, #

                       n_jobs=-1, #

                       scoring='accuracy', #

                       cv=nFolds) #

LogReg_fit.fit(X_train, y_train) #

scores = cross_val_score(LogReg_fit, X_train, y_train, cv = 10) # Cross validation
scores

for i in range(0,len(scores)):

    print("Resample %i: Accuracy - %0.2f" % (i+1, scores[i]) )

print("Mean accuracy - %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

ax = sns.boxplot(y = scores).set_title('Boxplot for summary metrics of test
samples')

plt.xlabel('Accuracy')

plt.show()
```

ANEXO 2: CÓDIGO FUENTE DE MODELO RANDOM FOREST

```
param = {'RF__n_estimators': range(10, 200, 10)} #

pipe = Pipeline(steps=[('scaler', StandardScaler()),

                       ('RF', RandomForestClassifier(criterion='gini', #


max_features=len(X_train.columns),  #

                                                    min_samples_split=5,  #

                                                    min_samples_leaf=5, #

                                                    max_depth=5, #

                                                    random_state=150))]) #

nFolds = 10

rf_fit = GridSearchCV(estimator=pipe, #

                      param_grid=param, #

                      n_jobs=-1, #

                      scoring='accuracy', #

                      cv=nFolds) #

rf_fit.fit(X_train, y_train) #

nFolds = 10

cv_errors = np.empty([nFolds,len(rf_fit.cv_results_['split0_test_score'])])

for split in range(nFolds):

    cv_errors[split,:] = rf_fit.cv_results_['split' + str(split) + '_test_score']

meanAcc = cv_errors.mean(0)

stdAcc = cv_errors.std(0)

scores_rf = meanAcc #
```

```
plt.figure(figsize=(12, 6))

plt.plot(rf_fit.cv_results_['param_RF__n_estimators'], meanAcc, marker='o',
markersize=10)

plt.errorbar(rf_fit.cv_results_['param_RF__n_estimators'], meanAcc, yerr=stdAcc,
linestyle="None", ecolor='lightblue')

plt.plot(rf_fit.best_params_['RF__n_estimators'], rf_fit.best_score_,marker='o',
markersize=15, color='red')

plt.title('Accuracy Rate number of trees Value. Best model with ' +
str(rf_fit.best_params_['RF__n_estimators']) + ' trees')

plt.xlabel('Number of trees')

plt.ylabel('Accuracy')

plt.show()
```