Profusion Data Academy Technical Test

Machine Learning Model for Customer Churn Prediction in Telecommunication Company

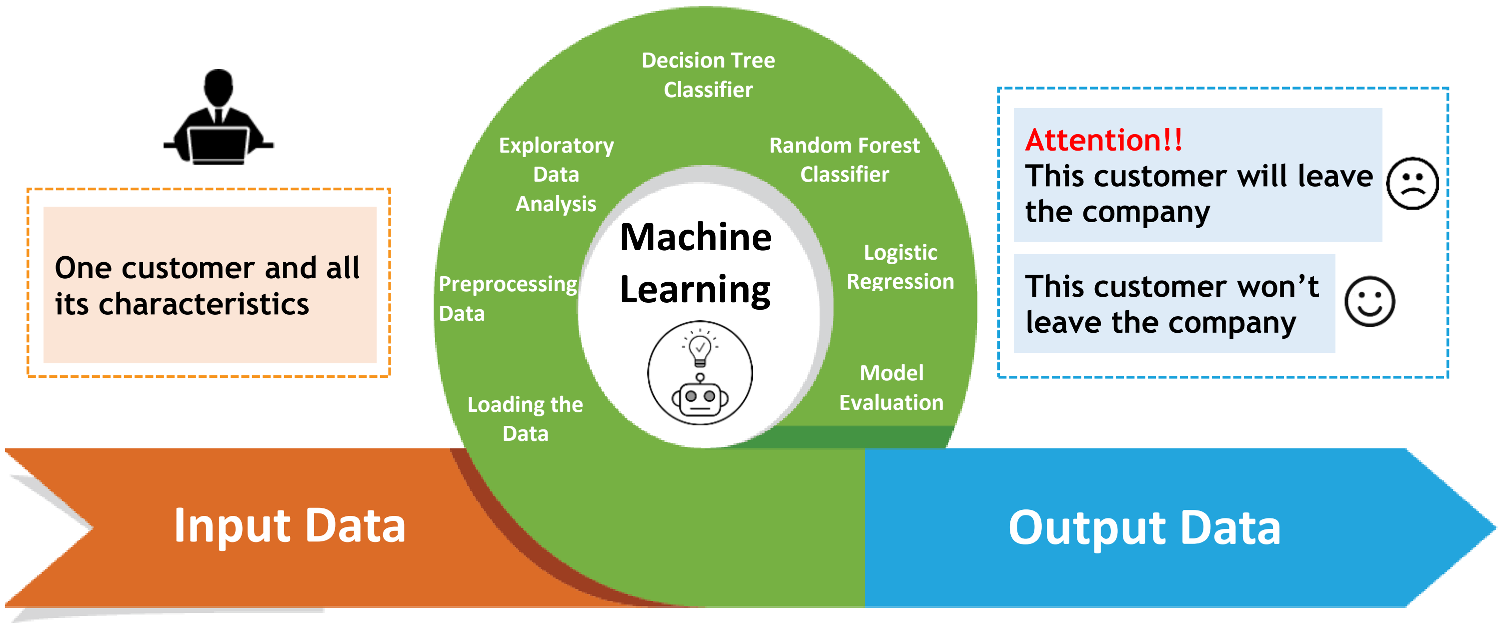
Julio Ernesto Pérez Lara

April 2021

GitHub: Code here

Table of Contents

Business Overview



the logistic regression will predict on average 80.10% of the clients correctly as customers who will cancel their contract based on the historical data

Introduction

During the last 10 years, telecommunication companies have started to suffer from a loss of valuable customers to competitors [1], mostly due to the increase in the number of telecom services providers in Asia, Europe, and the US. [2].

According to this paper [1], Customer churn happens when clients decide to cut their relationships with a company. In other words, “Churn” is a measurement of business that shows buyers who stop doing transactions ($) with an enterprise or a service, also known as customer attrition [3].

As explained by [4], Customer Churn is important not only because it can directly affect the revenues of an organization, but also because it is always more expensive to gain new customers than to keep a current client.

As mentioned by Mariana Curi [5] “The reasons that lead clients to the cancellation decision can be numerous, coming from poor service quality, delay on customer support, prices, new competitors entering the market, and so on. Usually, there is no single reason, but a combination of events can cause a customer displeasure”.

For that reason, Telecommunication companies have started to analyze the data using Machine Learning techniques to understand the reason behind the loss of customers. For example, in paper [6], the authors performed a Decision Tree Classifier model and concluded that customers who live in the suburban area and are engaged in line less than 10 minutes are likely to churn. Similarly, in research [7], the authors built a Naïve Bayes Classifier to predict the future churn for wireless customers. This model obtained a 68% of predictive accuracy.

Therefore, to reduce customer churn in a telecom company, it is necessary to predict which clients are at high risk of churn. By doing this, any organization will be able to create reactive action plans and tackle those factors that cause customer attrition.

Objective

This project is requiring an automated decision for the future: is a customer likely to churn? (yes/no). This represents a binary-classification problem where the output has 2 classes:

- The client will cancel his/her subscription (*y*) given certain features (*x*)
- The client won't cancel his/her subscription (*y*) given certain features (*x*)

In Machine Learning, classification is a supervised learning concept which basically categorizes a set of data into classes [8], and it is perhaps, the most important form of prediction, in which the goal is to forecast whether a record is 0 or 1 [9].

For that reason, a Machine Learning algorithm for classification[1] will be suggested to successfully solve this problem. The main goal is to design a machine learning model capable to predict customer churn based on the "Telecom-Users" dataset available at Kaggle[2]
Mainly Python, Pandas, and Scikit-Learn libraries will be used for this implementation. The complete code can be found on my GitHub and zip file.

Methodology

In this project, a Decision Tree Classifier[3], Random Forest Classifier[4], and Logistic Regression[5] have been built, trained, and evaluated, as Machine Learning algorithms capable of predicting whether the customer will stay or leave the company (*y*), based on the features found (*x*) in the Telecom Users dataset[1]

[1] The most common machine learning algorithms for classification are: (i) Logistic Regression, (ii) K-Nearest (iii) Neighbours, (iv) Support Vector Machines, (v) Kernel SVM, (vi) Naïve Bayes, (vii) Decision Tree Classifier, and (viii) Random Forest Classifier [10]

[2] Telecom-Users datasets can be downloaded from: https://www.kaggle.com/radmirzosimov/telecom-users-dataset

[3] A **Decision Tree Classifier** tries to solve a classification problem, by using tree representation. Each internal node of the tree corresponds to an attribute, and each leaf node corresponds to a class label [8]

[4] A **Random Forest Classifier** creates a "forest" of decision trees, each of which votes on the predicted class of an observation [11]

[5] A **Logistic Regression** "allow us to predict the probability that an observation is of a certain class using a straightforward and well-un understood approach" [11]

The Machine learning procedures applied in this technical test have been organized logically according to the 7 stages of ML proposed by Yufeng in [12]

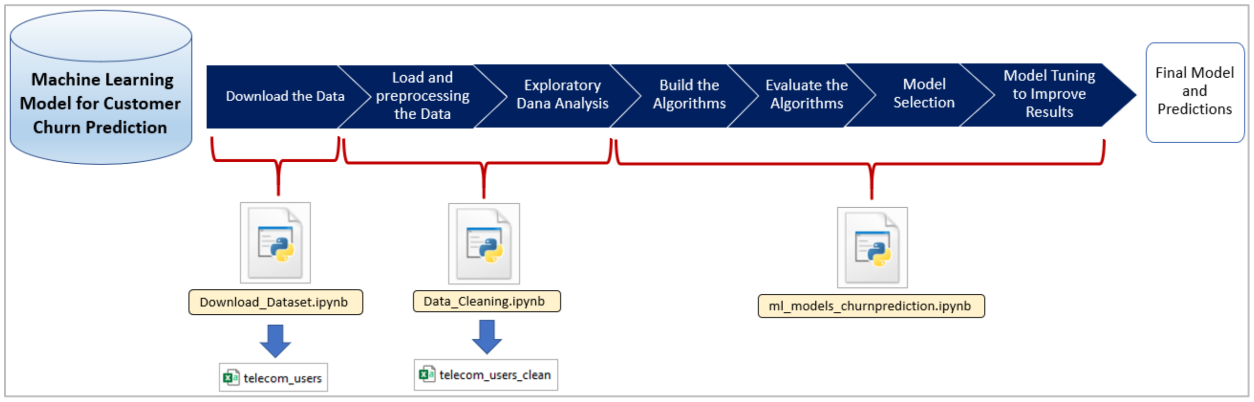
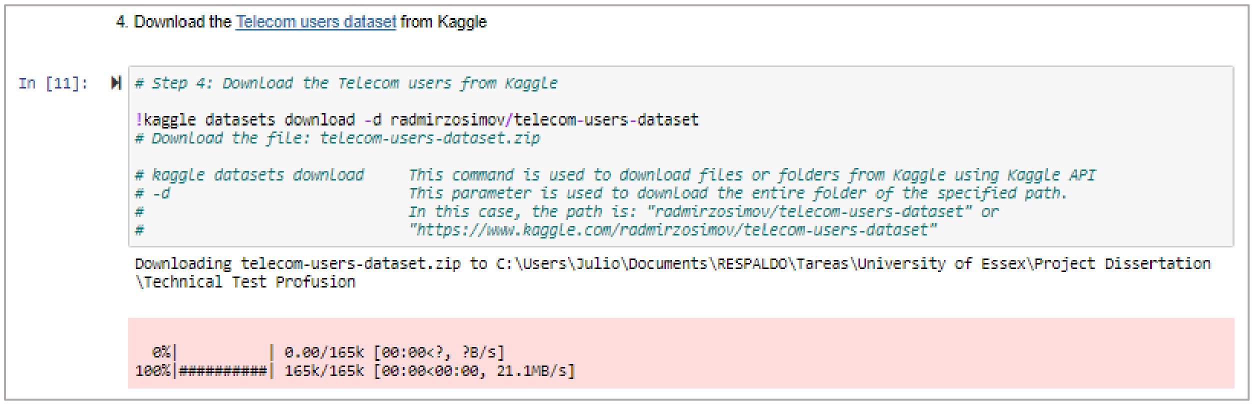

Machine Learning Model for Customer Churn Prediction
Download the Data
Load and preprocessing the Data
Exploratory Dana Analysis
Build the Algorithms
Evaluate the Algorithms
Model Selection
Model Tuning to Improve Results
Final Model and Predictions
Download_Dataset.ipynb
Data_Cleaning.ipynb
ml_models_churnprediction.ipynb
telecom_users
telecom_users_clean

Figure 1: ML Approach to solve the customer churn prediction in the Telecom Users dataset
Source: Own elaboration in Microsoft-Excel. **Notes:**

- The file “telecom_users” will appear in the directory after executing the file “Download_Dataset.ipynb”
- The file “telecom_users_clean” will appear in the directory after executing the file “Data_Cleaning.ipynb”

Dataset

The data used in this project is called Telecom Users dataset and it is available at Kaggle.com. The CSV file was downloaded using the Kaggle API commands, which will help us to avoid performing manual operations. (Figure 2)

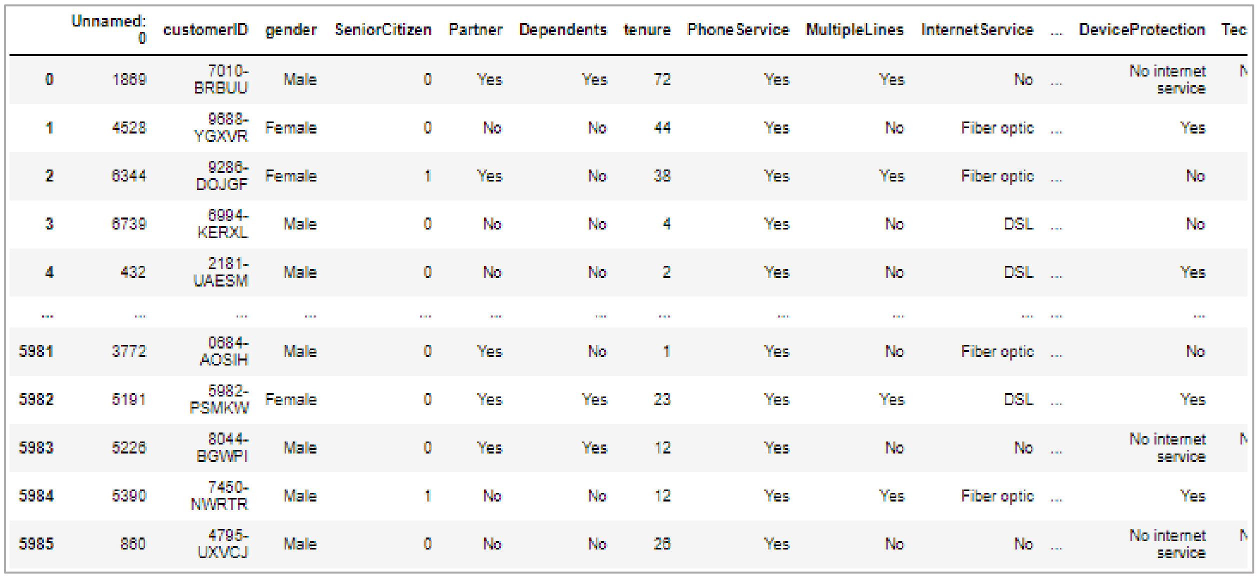


4. Download the Telecom users dataset from Kaggle

```
In [11]:  # Step 4: Download the Telecom users from Kaggle

          !kaggle datasets download -d radmirzosimov/telecom-users-dataset
          # Download the file: telecom-users-dataset.zip

          # kaggle datasets download     This command is used to download files or folders from Kaggle using Kaggle API
          # -d                           This parameter is used to download the entire folder of the specified path.
          #                              In this case, the path is: "radmirzosimov/telecom-users-dataset" or
          #                              "https://www.kaggle.com/radmirzosimov/telecom-users-dataset"

          Downloading telecom-users-dataset.zip to C:\Users\Julio\Documents\RESPALDO\Tareas\University of Essex\Project Dissertation
          \Technical Test Profusion


            0%|          | 0.00/165k [00:00<?, ?B/s]
          100%|##########| 165k/165k [00:00<00:00, 21.1MB/s]
```

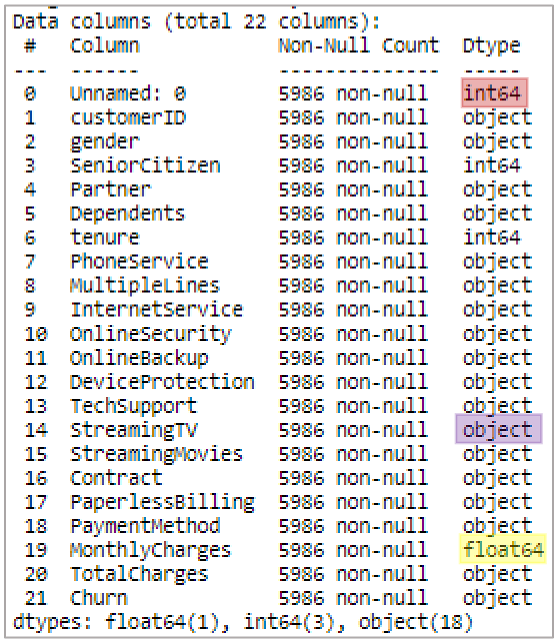
Figure 2: Code used to download the Telecom Users file.
Source: *Own elaboration in Jupyter Notebook.* **Notes:** For further information about this process, please go to the file: *Download_Dataset.ipynb* and read the comments in code

The Telecom Users dataset provides 5986 customer information in 22 columns, which contain both numerical and categorical types of information. (Figure 3 and Figure 4)



	Unnamed: 0	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	...	DeviceProtection	Tec
0	1869	7010-BRBUU	Male	0	Yes	Yes	72	Yes	Yes	No	...	No internet service	N
1	4528	9688-YGXVR	Female	0	No	No	44	Yes	No	Fiber optic	...	Yes	
2	6344	9286-DOJGF	Female	1	Yes	No	38	Yes	Yes	Fiber optic	...	No	
3	6739	6994-KERXL	Male	0	No	No	4	Yes	No	DSL	...	No	
4	432	2181-UAESM	Male	0	No	No	2	Yes	No	DSL	...	Yes	
...	...	...	...	...	...	...	...	...	...	...	...	...	
5981	3772	0684-AOSIH	Male	0	Yes	No	1	Yes	No	Fiber optic	...	No	
5982	5191	5982-PSMKW	Female	0	Yes	Yes	23	Yes	Yes	DSL	...	Yes	
5983	5226	8044-BGWPI	Male	0	Yes	Yes	12	Yes	No	No	...	No internet service	N
5984	5390	7450-NWRTR	Male	1	No	No	12	Yes	Yes	Fiber optic	...	Yes	
5985	860	4795-UXVCJ	Male	0	No	No	26	Yes	No	No	...	No internet service	N

Figure 3: Example of some rows and columns found in Telecom Users dataframe.
Source: Own elaboration in Jupyter Notebook. **Notes:** For further information about this process, please go to the file: Data_Cleaning.ipynb



```
Data columns (total 22 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Unnamed: 0        5986 non-null   int64
 1   customerID        5986 non-null   object
 2   gender            5986 non-null   object
 3   SeniorCitizen     5986 non-null   int64
 4   Partner           5986 non-null   object
 5   Dependents        5986 non-null   object
 6   tenure            5986 non-null   int64
 7   PhoneService      5986 non-null   object
 8   MultipleLines     5986 non-null   object
 9   InternetService   5986 non-null   object
 10  OnlineSecurity    5986 non-null   object
 11  OnlineBackup      5986 non-null   object
 12  DeviceProtection  5986 non-null   object
 13  TechSupport       5986 non-null   object
 14  StreamingTV       5986 non-null   object
 15  StreamingMovies   5986 non-null   object
 16  Contract          5986 non-null   object
 17  PaperlessBilling  5986 non-null   object
 18  PaymentMethod     5986 non-null   object
 19  MonthlyCharges    5986 non-null   float64
 20  TotalCharges      5986 non-null   object
 21  Churn             5986 non-null   object
dtypes: float64(1), int64(3), object(18)
```

Figure 4: Data types found in Telecom Users Dataframe.
Source: Own elaboration in Jupyter Notebook. **Notes:** For further information about this process, please go to the file: Data_Cleaning.ipynb

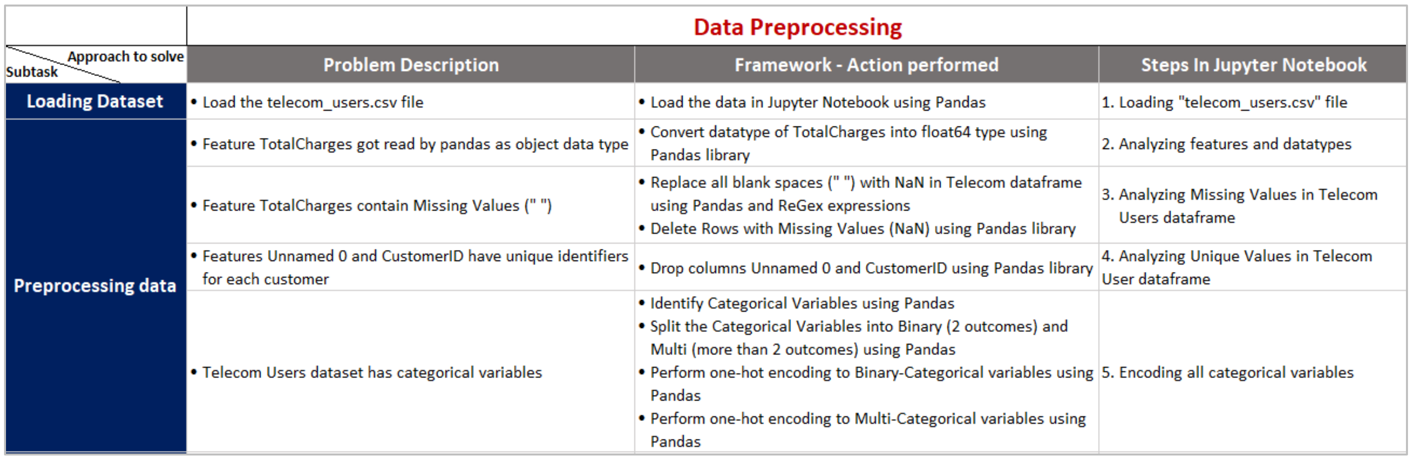
The columns of the dataset are:

A. **Unnammed (0):** This column seems to be a sequence number
B. **customerID:** Unique identifier to each customer
C. **gender:** Client gender (Male / Female)
D. **SeniorCitizen:** Whether the client is retired or not (1, 0)
E. **Partner:** Whether the client is married or not (Yes, No)
F. **Dependents:** Whether the client has children or not (Yes, No)
G. **tenure:** Number of months that a person has been a client of the company
H. **PhoneService:** Whether the telephone service is connected or not (Yes, No)
I. **MultipleLines:** Whether multiple phone lines are connected (Yes, No, No Internet Service)
J. **InternetService:** Client's Internet service provider (DSL, Fiber optic, No)
K. **OnlineSecurity:** Whether the online security service is connected (Yes, No, No Internet Service)
L. **OnlineBackup:** Whether the online backup service is activated (Yes, No, No Internet Service)
M. **DeviceProtection:** Whether the client has equipment insurance (Yes, No, No Internet Service)
N. **TechSupport:** Whether the technical support service is connected (Yes, No, No Internet Service)
O. **StreamingTV:** Whether the streaming tv service is connected (Yes, No, No Internet Service)
P. **StreamingMovies:** Whether the streaming cinema service is activated (Yes, No, No Internet Service)
Q. **Contract:** Type of customer contract (Month-to-month, One year, Two-year)
R. **PaperlessBilling:** Whether the client uses paperless billing (Yes, No)
S. **PaymentMethod:** Payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))
T. **MonthlyCharges:** Current monthly payment
U. **TotalCharges:** The total amount that the client paid for the services for the entire time
V. **Churn:** whether there was a churn (Yes, No)

Load the Data and Preprocessing

According to [11], Data preprocessing in Machine Learning refers to the methods of cleaning, organizing, and making ready the raw data to make it suitable for any machine learning algorithm. In other words, preprocessing is a data mining technique that converts raw data into a comprehensible and readable format [9].

In this model, each subtask of this step is summarized in the Table 1, but for a more detailed explanation, please go to the Jupyter file: “Data_Cleaning.ipynb” and read all the comments and markdowns located in all the subprocesses.



	Data Preprocessing		
Approach to solve / Subtask	Problem Description	Framework - Action performed	Steps In Jupyter Notebook
Loading Dataset	• Load the telecom_users.csv file	• Load the data in Jupyter Notebook using Pandas	1. Loading "telecom_users.csv" file
Preprocessing data	• Feature TotalCharges got read by pandas as object data type	• Convert datatype of TotalCharges into float64 type using Pandas library	2. Analyzing features and datatypes
	• Feature TotalCharges contain Missing Values (" ")	• Replace all blank spaces (" ") with NaN in Telecom dataframe using Pandas and ReGex expressions • Delete Rows with Missing Values (NaN) using Pandas library	3. Analyzing Missing Values in Telecom Users dataframe
	• Features Unnamed 0 and CustomerID have unique identifiers for each customer	• Drop columns Unnamed 0 and CustomerID using Pandas library	4. Analyzing Unique Values in Telecom User dataframe
	• Telecom Users dataset has categorical variables	• Identify Categorical Variables using Pandas • Split the Categorical Variables into Binary (2 outcomes) and Multi (more than 2 outcomes) using Pandas • Perform one-hot encoding to Binary-Categorical variables using Pandas • Perform one-hot encoding to Multi-Categorical variables using Pandas	5. Encoding all categorical variables

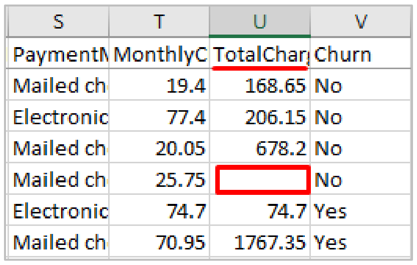
Table 1: Overview of the Subtasks performed in Preprocessing phase.
Source: Own elaboration in Microsoft-Excel. **Notes:**

- The feature **TotalCharges** got read by pandas as object data type. This is incorrect, since this variable should store only numbers with decimals (charges). (Figure 4)
  Usually, this happens because the feature TotalCharges contains some spaces **(" ")** as values, causing that Pandas recognize the whole column as object type (string) and not as float64 type (numbers with decimals) (Figure 5)
  In fact, having this feature as object type could occasionate problems during the exploratory analysis and need to be handled.

- The **TotalCharge** has **NaN** values, which means that the value is not available there (missing).
  To be more precise, there are **10 missing values** in TotalCharge. (Figure 6)
  In other words, about **0.17%** of the TotalCharge values are missing (10/5986)
  As a result, the records (rows) with NaN values will be deleted from the dataframe since they do not represent more than 3% of the whole data

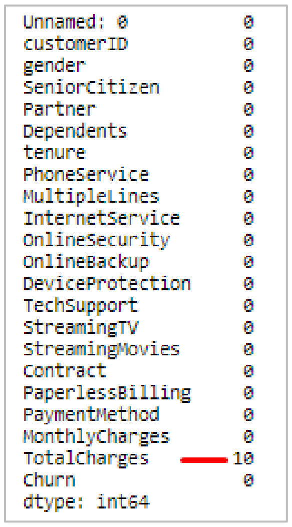
- As observed in figure (Figure 7) the column **Unnamed 0** and **CustomerID** have unique identifiers for each customer, confirming that each row represents a sequence number or a single customer. This features won't contribute to the Machine Learning models; therefore, they should be deleted.

- Figure 8 shows the one-hot encoding process, which according to [11], It is the most common and correct way to deal with non-ordinal categorical data, because it consists of creating an additional column for each group of the categorical feature and mark each observations belonging (value=1) or not (value=0) to that group".



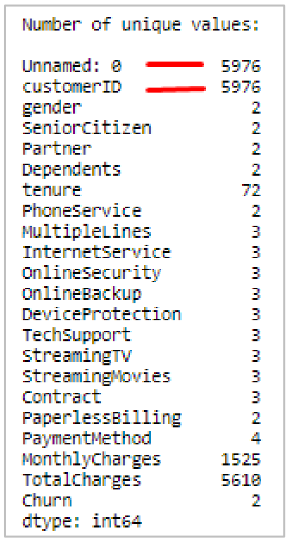
S	T	U	V
PaymentM	MonthlyC	TotalChar	Churn
Mailed ch	19.4	168.65	No
Electronic	77.4	206.15	No
Mailed ch	20.05	678.2	No
Mailed ch	25.75		No
Electronic	74.7	74.7	Yes
Mailed ch	70.95	1767.35	Yes

Figure 5: Example of spaces (Missing values) in feature **TotalCharges**
Source: Own elaboration based on the Telecom Users dataset download from Kaggle.com



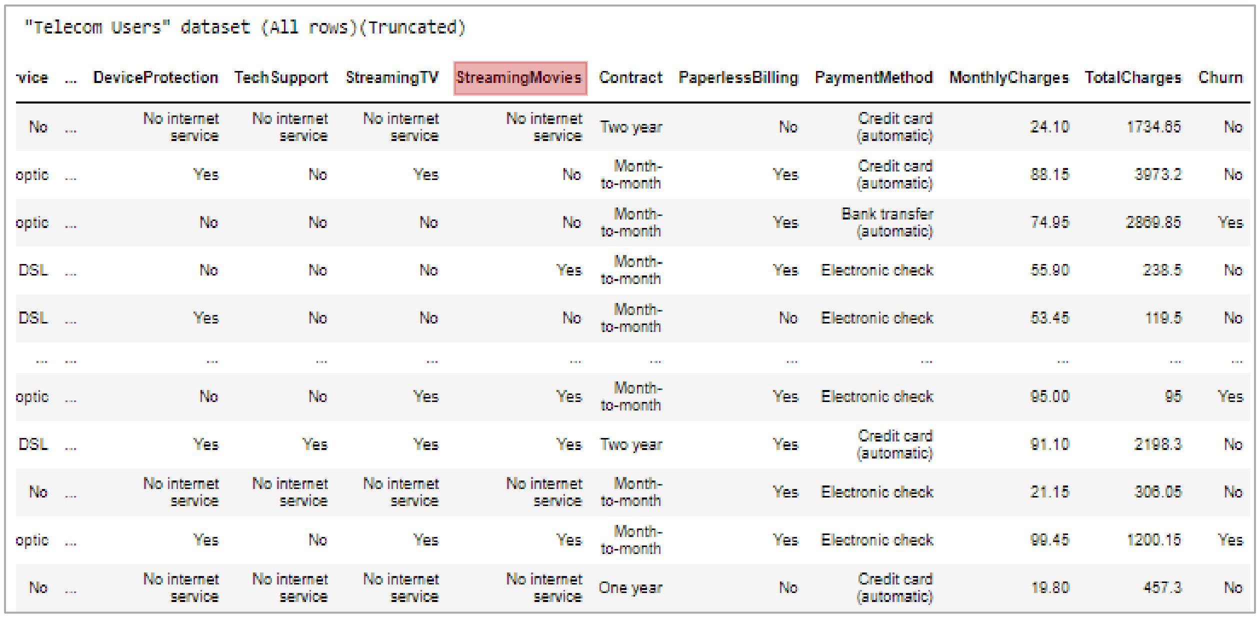
```
Unnamed: 0          0
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges       10
Churn               0
dtype: int64
```

Figure 6: Missing Values (NaN) found in Telecom Users Dataframe.
Source: Own elaboration in Jupyter Notebook. **Notes:** For further information about this process, please go to the file: Data_Cleaning.ipynb

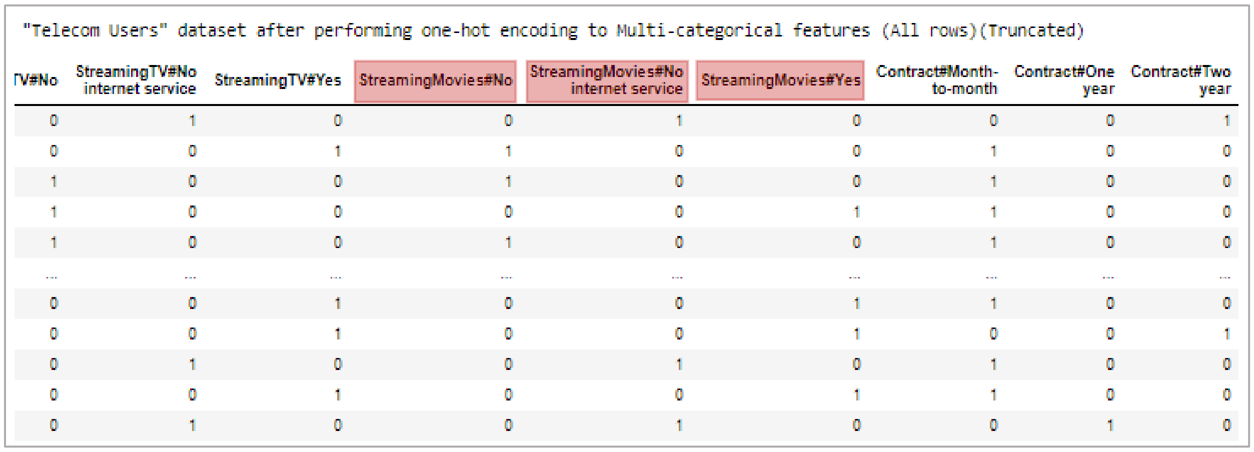


```
Number of unique values:

Unnamed: 0          5976
customerID          5976
gender                 2
SeniorCitizen          2
Partner                2
Dependents             2
tenure                72
PhoneService           2
MultipleLines          3
InternetService        3
OnlineSecurity         3
OnlineBackup           3
DeviceProtection       3
TechSupport            3
StreamingTV            3
StreamingMovies        3
Contract               3
PaperlessBilling       2
PaymentMethod          4
MonthlyCharges      1525
TotalCharges        5610
Churn                  2
dtype: int64
```

Figure 7: Total number of unique values found in Telecom Users Dataframe.
Source: Own elaboration in Jupyter Notebook. **Notes:** For further information about this process, please go to the file: Data_Cleaning.ipynb


"Telecom Users" dataset (All rows)(Truncated)

vice	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
No	...	No internet service	No internet service	No internet service	No internet service	Two year	No	Credit card (automatic)	24.10	1734.65	No
optic	...	Yes	No	Yes	No	Month-to-month	Yes	Credit card (automatic)	88.15	3973.2	No
optic	...	No	No	No	No	Month-to-month	Yes	Bank transfer (automatic)	74.95	2869.85	Yes
DSL	...	No	No	No	Yes	Month-to-month	Yes	Electronic check	55.90	238.5	No
DSL	...	Yes	No	No	No	Month-to-month	No	Electronic check	53.45	119.5	No
...	...	...	...	...	...	...	...	...	...	...	...
optic	...	No	No	Yes	Yes	Month-to-month	Yes	Electronic check	95.00	95	Yes
DSL	...	Yes	Yes	Yes	Yes	Two year	Yes	Credit card (automatic)	91.10	2198.3	No
No	...	No internet service	No internet service	No internet service	No internet service	Month-to-month	Yes	Electronic check	21.15	306.05	No
optic	...	Yes	No	Yes	Yes	Month-to-month	Yes	Electronic check	99.45	1200.15	Yes
No	...	No internet service	No internet service	No internet service	No internet service	One year	No	Credit card (automatic)	19.80	457.3	No


"Telecom Users" dataset after performing one-hot encoding to Multi-categorical features (All rows)(Truncated)

TV#No	StreamingTV#No internet service	StreamingTV#Yes	StreamingMovies#No	StreamingMovies#No internet service	StreamingMovies#Yes	Contract#Month-to-month	Contract#One year	Contract#Two year
0	1	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0	0
1	0	0	1	0	0	1	0	0
1	0	0	0	0	1	1	0	0
1	0	0	1	0	0	1	0	0
...	...	...	...	...	...	...	...	...
0	0	1	0	0	1	1	0	0
0	0	1	0	0	1	0	0	1
0	1	0	0	1	0	1	0	0
0	0	1	0	0	1	1	0	0
0	1	0	0	1	0	0	1	0

Figure 8: Comparative between original dataset vs dataset after performing One-hot encoding
Source: Own elaboration in Jupyter Notebook. **Notes:**

- For further information about this process, please go to the file: Data_Cleaning.ipynb
- The new dataset (clean) provides 5976 customer information in 41 columns, which contain only numerical information

Exploratory Data Analysis (EDA)

According to [11], EDA refers to the important process of performing initial investigations on data to discover patterns or spot anomalies. As explained in [9], EDA is primarily used to gain a maximum understanding of the dataset and its underlying structure.

In this model, EDA was performed to answer the following 4 questions:

1. How long is the customer lifespan until subscription cancellation?

As shown in the bar chart (Figure 9):

- The majority of customers canceled their subscriptions in the first month. In fact, 20% of subscribers left in the first month.
- Most of the clients left in the first 3 months (31.9%).
- Senior management should develop a strategy no longer than 3 months to try to tackle the factors that cause customer churn.

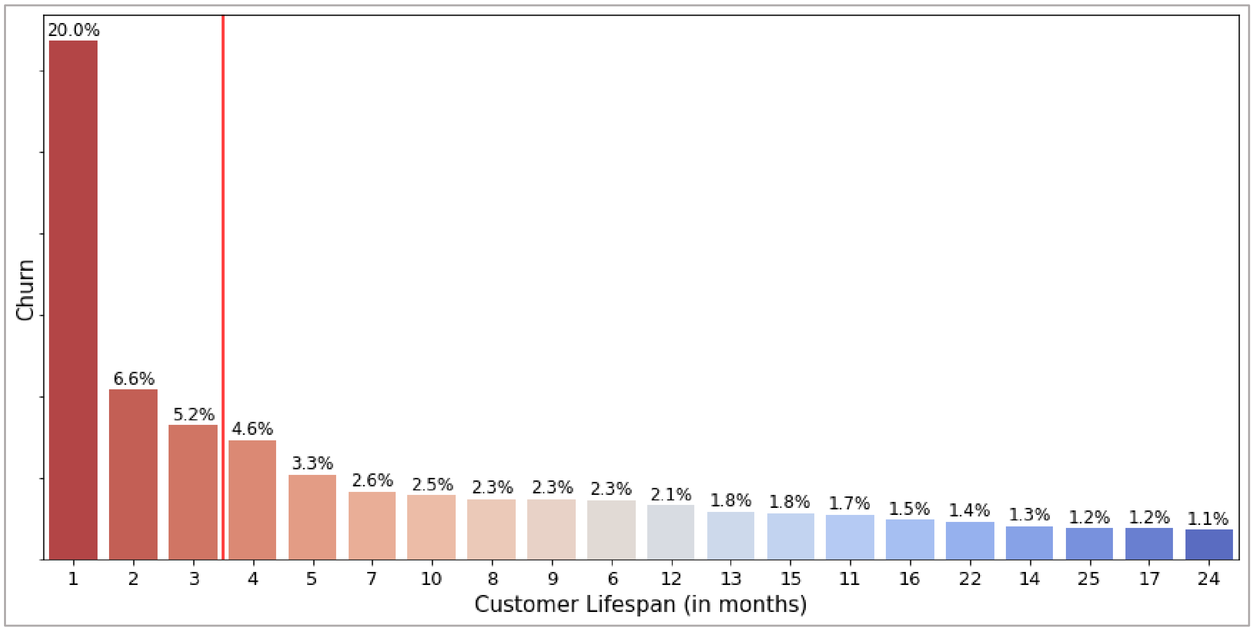


Figure 9: Bar chart that illustrates the customer lifespan until subscription cancellation.
Source: Own elaboration in Jupyter Notebook using matplotlib and seaborn libraries. **Notes:**

- For further information about this process, please go to the file: Data_Cleaning.ipynb
- This chart displays only the first 20 months with more subscription cancelations

2. How is the profile of customers that cancel their subscription?

The first thing to do is to classify the features in terms of personal attributes, services, or contract aspects:

A) In terms of personal attributes:

According to the dataset provided, these are the personal attributes:

1. Gender: Client gender (Male / Female)

2. SeniorCitizen: Whether the client is retired or not (Yes, No)

3. Partner: Whether the client is married or not (Yes, No)

4. **Dependents**: Whether the client has children or not (Yes, No)

The following countplots summarizes which personal characteristics contribute mostly to the cancelation decision:

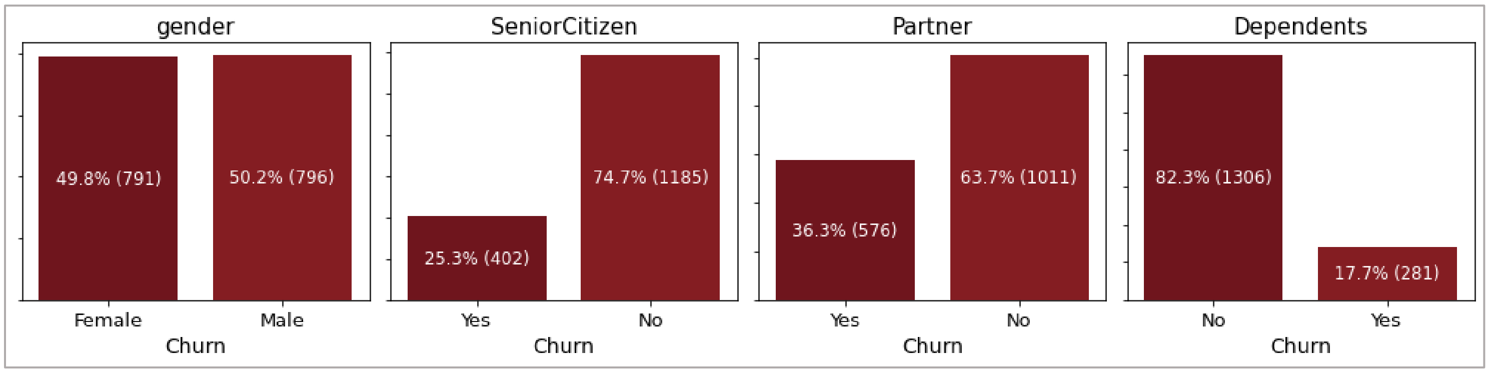


Figure 10: Countplots "Churn" vs Personal attributes
Source: Own elaboration in Jupyter Notebook using matplotlib and seaborn libraries. **Notes:**

- For further information about this process, please go to the file: Data_Cleaning.ipynb

As shown in Figure 10:

- Clients **without dependents** are **4 times more** likely to cancel their subscription
- Customers that are **NOT Senior Citizens** are **3 times** more likely to churn.
- Single clients are almost **2 times** more likely to cancel their subscription

B) In terms of services attributes

According to the dataset provided, these are the attributes of the service:

1. PhoneService: Whether the telephone service is connected or not (Yes, No)

2. MultipleLines: Whether multiple phone lines are connected (Yes, No, No Internet Service)

3. InternetService: Client's Internet service provider (DSL, Fiber optic, No)

4. **OnlineSecurity**: Whether the online security service is connected (Yes, No, No Internet Service)

5. OnlineBackup: Whether the online backup service is activated (Yes, No, No Internet Service)

6. DeviceProtection: Whether the client has equipment insurance (Yes, No, No Internet Service)

7. TechSupport: Whether the technical support service is connected (Yes, No, No Internet Service)

8. StreamingTV: Whether the streaming tv service is connected (Yes, No, No Internet Service)

9. StreamingMovies: Whether the streaming cinema service is activated (Yes, No, No Internet Service)

The countplots in Figure 11 summarizes which service characteristics contribute mostly to the cancelation decision:

- It gives an understanding regarding which kind of services the customers are more likely to cancel.
- **90% of the customers** that have **Phone Service enabled** canceled their subscriptions
- Customers that have **fiber-optic internet service** are **2 times** more likely to churn than those who have DSL.
- Customers that do **not have online security, device protection, online backup, and tech support services enabled** are more likely to cancel their subscription

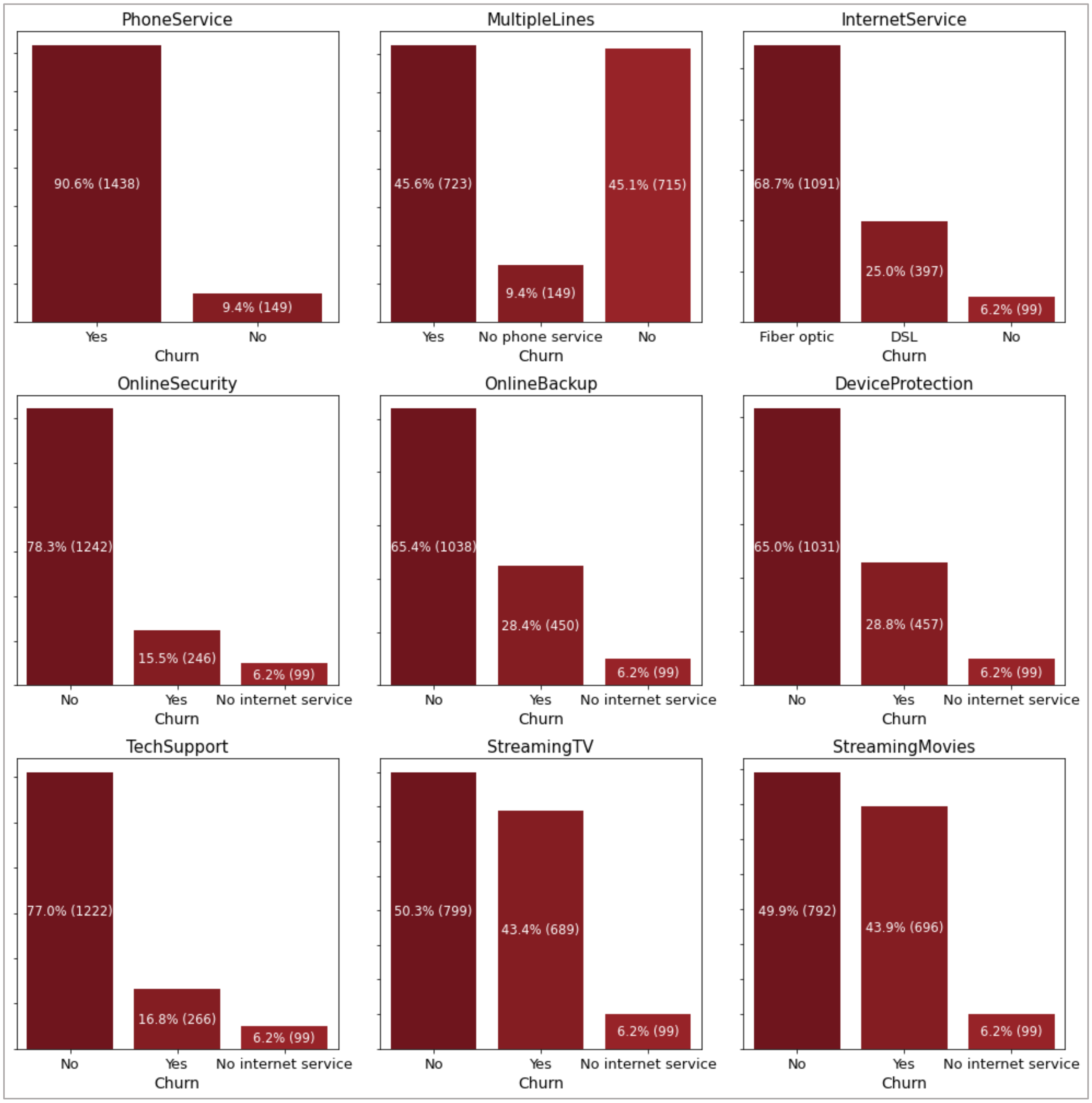


Figure 11: Countplots "Churn" vs Services attributes
Source: Own elaboration in Jupyter Notebook using matplotlib and seaborn libraries. **Notes:**

- For further information about this process, please go to the file: Data_Cleaning.ipynb

C) In terms of contract aspects:

According to the dataset provided, these are the contract attributes:

1. Contract: Type of customer contract (Month-to-month, One year, Two-year)

2. PaperlessBilling: Whether the client uses paperless billing (Yes, No)

3. PaymentMethod: Payment method (Electronic check, Mailed check, Bank transfer (automatic), Credit card (automatic))

The following countplots summarizes which contract characteristics contribute mostly to the cancelation decision:

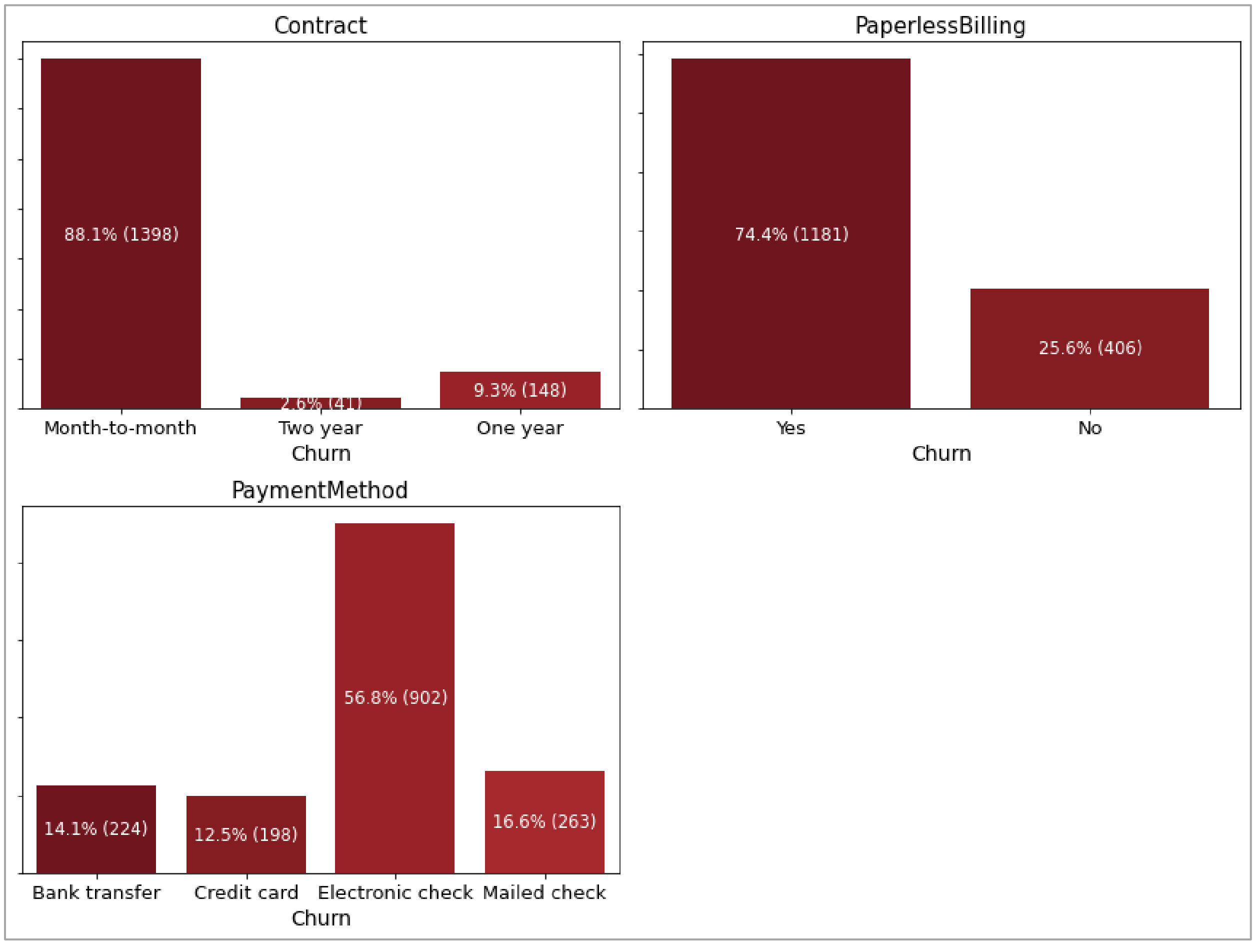


Figure 12: Countplots “Churn” vs Contract aspects
Source: Own elaboration in Jupyter Notebook using matplotlib and seaborn libraries. **Notes:**

- For further information about this process, please go to the file: Data_Cleaning.ipynb

As shown in Figure 12:

- The majority of **clients** that cancel their subscription have **month-to-month contract type** and **paperless billing enabled**.
- Customers that have payment method as **electronic check** are **3 times** more likely to churn.

3. What is the correlation between the target variable "Churn" with the other features?

As shown in the correlation chart (Figure 13):

1. **Month-to-month contracts, absence of online security,** and **absence of tech support seem** to be **positively** correlated with the churn.
2. **Tenure**, **two-years contract,** and **absence of internet service** seem to be **negatively** correlated with churn.

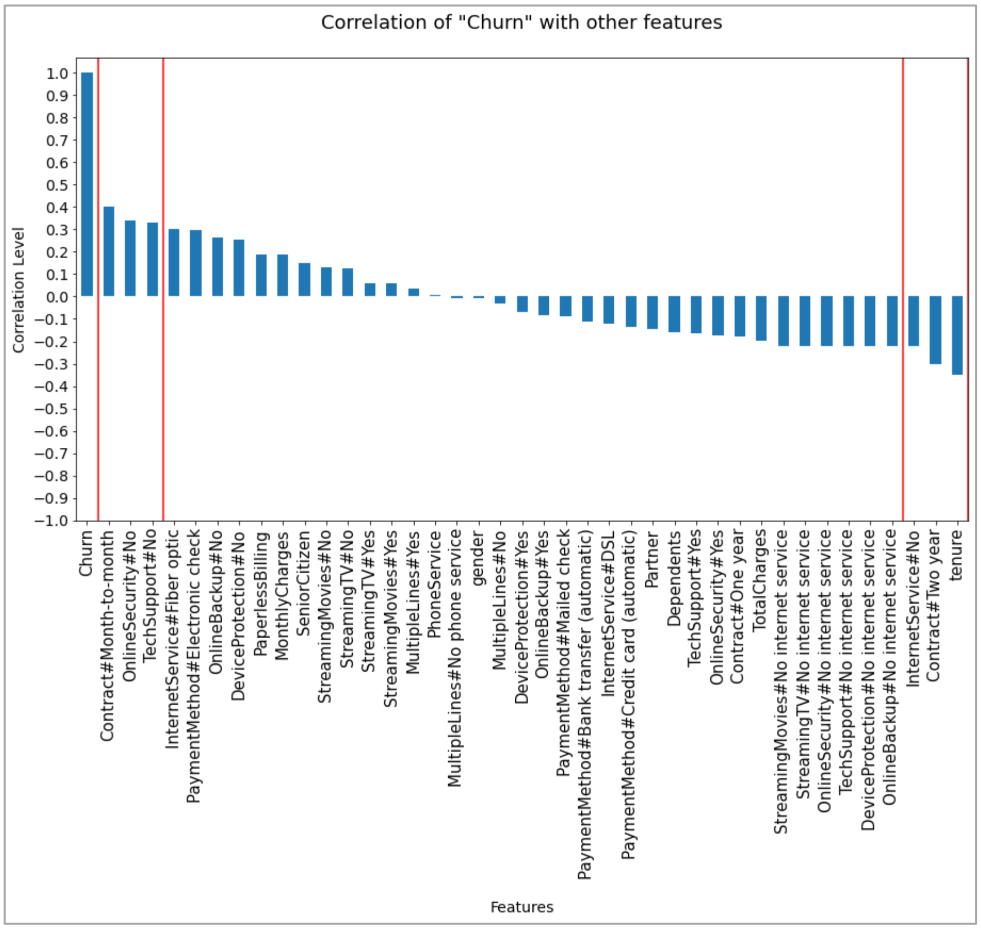


Figure 13: Correlation chart between "Churn" with Other features
Source: Own elaboration in Jupyter Notebook using matplotlib and pandas libraries. **Notes:**

- For further information about this process, please go to the file: Data_Cleaning.ipynb

4. Is the dataset balanced?

As shown in the countplot (Figure 14):

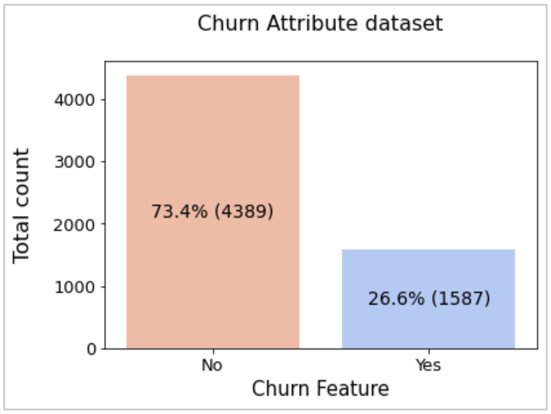
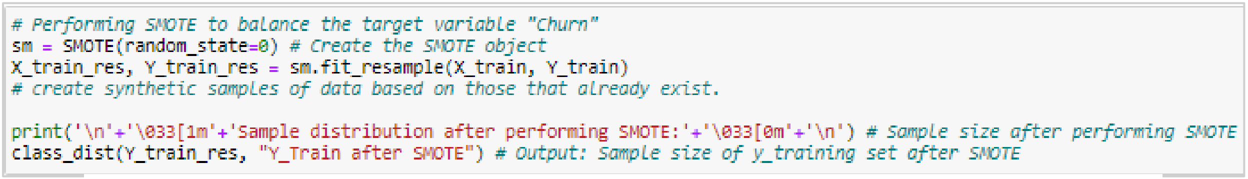


Figure 14: Countplot of classes in "Churn" feature
Source: Own elaboration in Jupyter Notebook using matplotlib and pandas libraries. **Notes:**

- For further information about this process, please go to the file: 03_ml_models_churnprediction

1. The class "**No**" is **almost 3 times bigger** than class "**Yes**", which demonstrates that the dataset is highly imbalanced.
   Ideally, the dataset should be balanced (50%/50%) to avoid models overfitting.

For this reason, the SMOTE technique should be implemented to balance the target variable "Churn" (Figure 15). In other words, the SMOTE method will randomly pick a point from the minority class and computed the k-nearest neighbors for this point. Finally, the synthetic will be added between the chosen point and its neighbors.



```
# Performing SMOTE to balance the target variable "Churn"
sm = SMOTE(random_state=0) # Create the SMOTE object
X_train_res, Y_train_res = sm.fit_resample(X_train, Y_train)
# create synthetic samples of data based on those that already exist.

print('\n'+'\033[1m'+'Sample distribution after performing SMOTE:'+'\033[0m'+'\n') # Sample size after performing SMOTE
class_dist(Y_train_res, "Y_Train after SMOTE") # Output: Sample size of y_training set after SMOTE
```

Figure 15: Code used to performed SMOTE in the dataset
Source: Own elaboration in Jupyter Notebook using imblearn.over_sampling libraries. **Notes:**

- For further information about this process, please go to the file: 03_ml_models_churnprediction

Once the SMOTE technique has been applied to the dataset, it could be observed that the attribute is now balanced:

```
6146 total examples in Y_Train after SMOTE
class 0: 3073 examples, 50.00%
class 1: 3073 examples, 50.00%
```

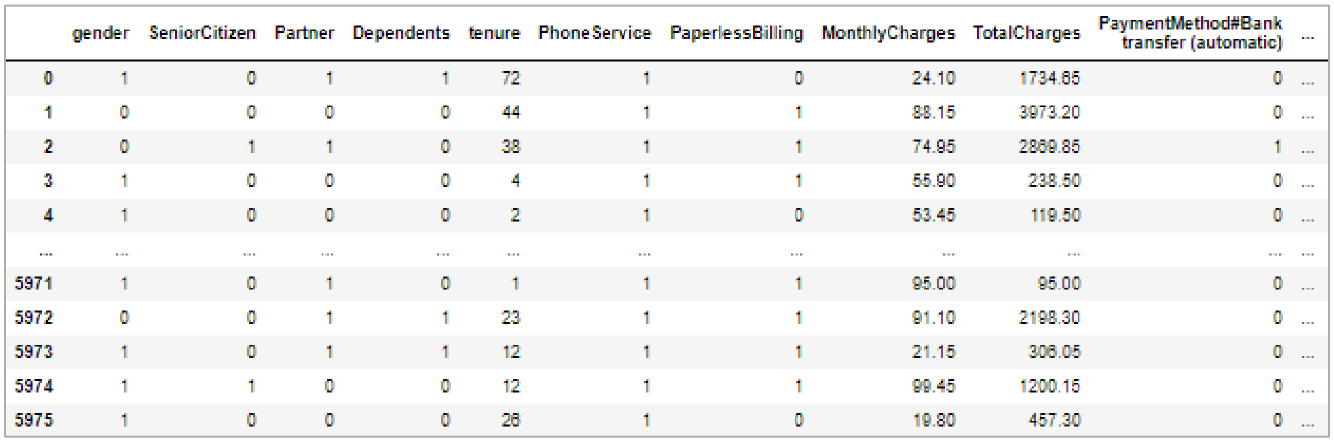
As seen earlier, oversampling was applied only to the training data. In other words, none of the information in the test data is being used to create synthetic observation, therefore, no information will bleed from test data into the model training.

Build the Machine Learning Models

After preprocessing the data and balance the target variable, the next step is the Machine Learning model implementation, which is summarized in 4 main steps:

1. Splitting the dataset into feature variables (independent variables) and target variable (dependent variable)
2. Splitting the data into training and testing sets.
3. Building 3 different Machine Learning algorithms: (i) Decision Tree Classifier, (ii) Random Forest Classifier, and (iii) Logistic Regression.
4. Applying GridSearchCv function for hyperparameter tuning to the best model.

For the first part, I split the original dataframe into 2 subsets: (i) feature variables dataframe (x), containing all attributes except column "Churn", and (ii) target variable dataframe that holds only the values that we wish to predict: column "Churn". (Figure 16 and Figure 17)

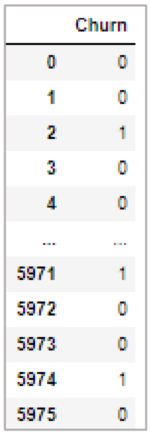


	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	PaperlessBilling	MonthlyCharges	TotalCharges	PaymentMethod#Bank transfer (automatic)	...
0	1	0	1	1	72	1	0	24.10	1734.65	0	...
1	0	0	0	0	44	1	1	88.15	3973.20	0	...
2	0	1	1	0	38	1	1	74.95	2869.85	1	...
3	1	0	0	0	4	1	1	55.90	238.50	0	...
4	1	0	0	0	2	1	0	53.45	119.50	0	...
...	...	...	...	...	...	...	...	...	...	...	...
5971	1	0	1	0	1	1	1	95.00	95.00	0	...
5972	0	0	1	1	23	1	1	91.10	2198.30	0	...
5973	1	0	1	1	12	1	1	21.15	306.05	0	...
5974	1	1	0	0	12	1	1	99.45	1200.15	0	...
5975	1	0	0	0	26	1	0	19.80	457.30	0	...

Figure 16: Feature variables dataframe (X)
Source: Own elaboration in Jupyter Notebook using pandas libraries. **Notes:**

- For further information about this process, please go to the file: ml_models_churnprediction.ipynb

For the second part, 70% of the dataset has been used for training the ML algorithms and 30% for testing and evaluating the models. This has been done following the explanation described by [13]

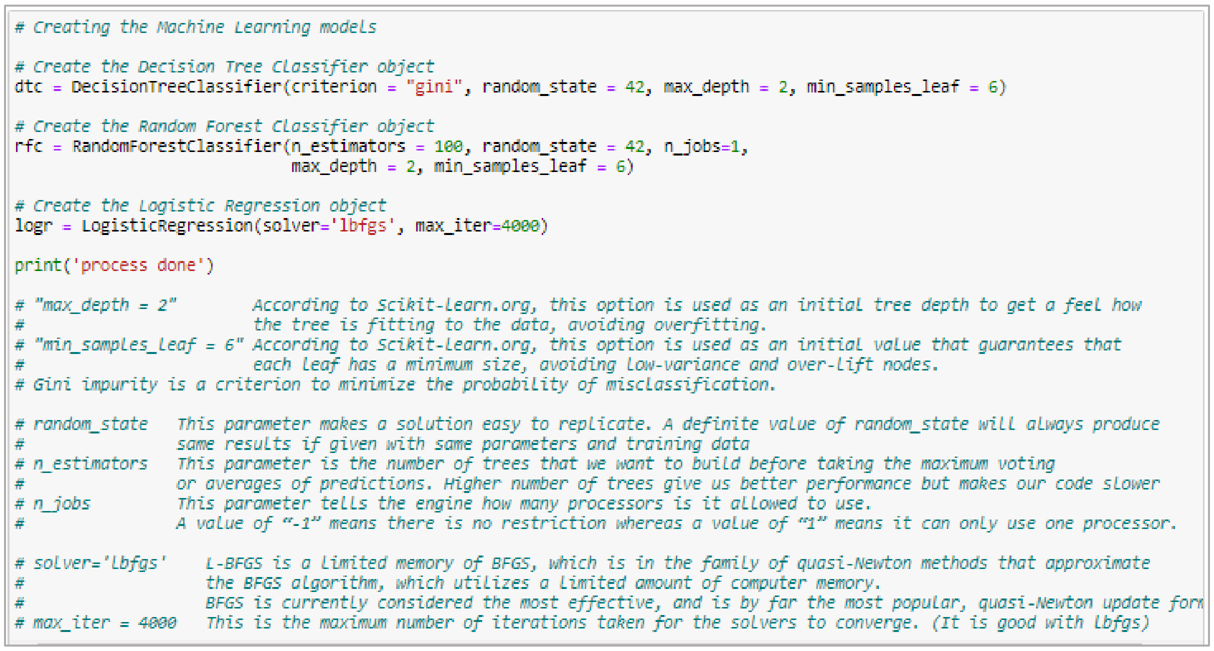


	Churn
0	0
1	0
2	1
3	0
4	0
...	...
5971	1
5972	0
5973	0
5974	1
5975	0

Figure 17: Target variable dataframe (Y)
Source: Own elaboration in Jupyter Notebook using pandas libraries. **Notes:**

- For further information about this process, please go to the file: ml_models_churnprediction.ipynb

For the third part, since the company wants to make a binary-categorical prediction, 3 different supervised algorithms, such as Decision Tree Classifier, Random Forest Classifier, and Logistic Regression have been proposed to forecast whether the customer will churn or not (y). These algorithms have been built using scikit-learn python library [6] (Figure 18)



```
# Creating the Machine Learning models

# Create the Decision Tree Classifier object
dtc = DecisionTreeClassifier(criterion = "gini", random_state = 42, max_depth = 2, min_samples_leaf = 6)

# Create the Random Forest Classifier object
rfc = RandomForestClassifier(n_estimators = 100, random_state = 42, n_jobs=1,
                             max_depth = 2, min_samples_leaf = 6)

# Create the Logistic Regression object
logr = LogisticRegression(solver='lbfgs', max_iter=4000)

print('process done')

# "max_depth = 2"        According to Scikit-Learn.org, this option is used as an initial tree depth to get a feel how
#                        the tree is fitting to the data, avoiding overfitting.
# "min_samples_leaf = 6" According to Scikit-Learn.org, this option is used as an initial value that guarantees that
#                        each leaf has a minimum size, avoiding low-variance and over-lift nodes.
# Gini impurity is a criterion to minimize the probability of misclassification.

# random_state   This parameter makes a solution easy to replicate. A definite value of random_state will always produce
#                same results if given with same parameters and training data
# n_estimators   This parameter is the number of trees that we want to build before taking the maximum voting
#                or averages of predictions. Higher number of trees give us better performance but makes our code slower
# n_jobs         This parameter tells the engine how many processors is it allowed to use.
#                A value of “-1” means there is no restriction whereas a value of “1” means it can only use one processor.

# solver='lbfgs'    L-BFGS is a limited memory of BFGS, which is in the family of quasi-Newton methods that approximate
#                   the BFGS algorithm, which utilizes a limited amount of computer memory.
#                   BFGS is currently considered the most effective, and is by far the most popular, quasi-Newton update form
# max_iter = 4000   This is the maximum number of iterations taken for the solvers to converge. (It is good with lbfgs)
```

Figure 18: Code used to build the 3 ML classifiers (Decision Tree, Random Forest, and Logistic Regression)
Source: Own elaboration in Python. **Notes:**

- For further information about this process, please go to the file: ml_models_churnprediction.ipynb

[6] For further information about the python library, please go to the documentation find in: https://scikit-learn.org/stable/

Decision Tree Classifier (DTC)

As concluded in [11], “decision tree classifier is a very popular supervised method for classification problems. In this algorithm, the data is continuously split according to a certain parameter (Figure 19). In general, as defined in [11], a tree-based learning model consists of:

1. **Nodes:** Test for the value of a certain attribute
2. **Edges/Branches:** Correspond to the outcome of a test and connect to the next node or leaf
3. **Leaf node:** Terminal nodes that predict the outcome (represent class labels)

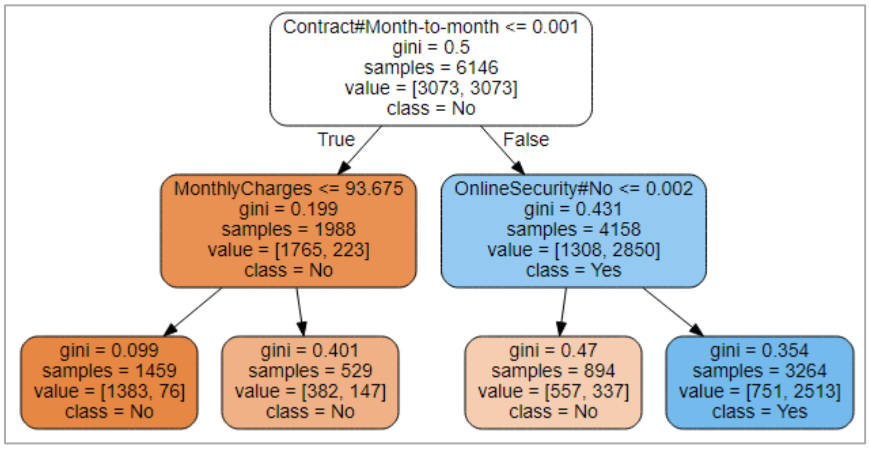


Figure 19: Decision Tree Graph for the Telecom User dataset
Source: Own elaboration in Python using graphviz and sklearn libraries. **Notes:**

- For further information about the code and parameters used, please go to the file: ml_models_churnprediction.ipynb

Random Forest Classifier (RFC)

In [11], the authors defined a random forest classifier as a supervised learning algorithm that consists of many decision trees. The key principle comprises the construction of “many” simple decision trees in the training stage and the majority vote across them in the classification stage. In other words, the RFC combines the output of multiple (randomly created) decision trees to generate the final output. (Figure 20)

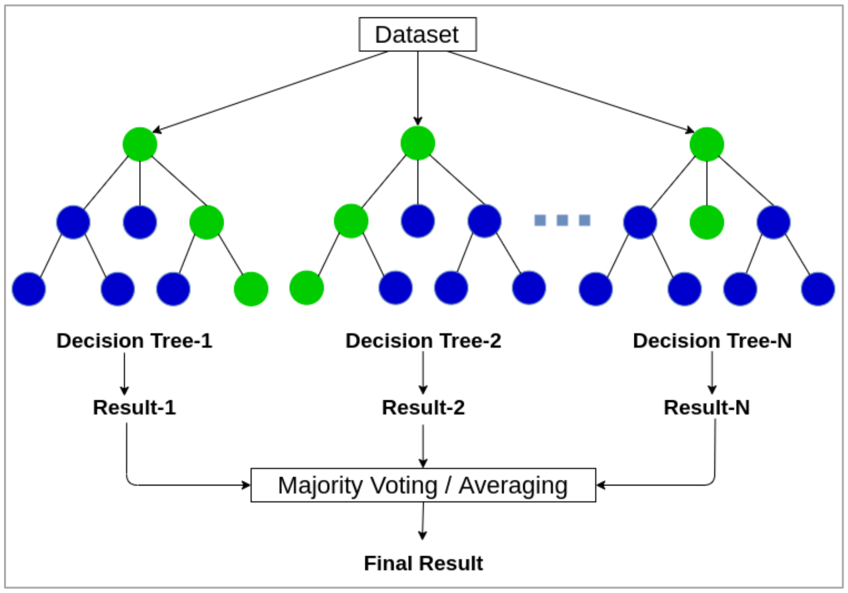


Figure 20: Illustration of the Random Forest Classifier process
Source: Reprinted from [16]

Logistic Regression Classifier (LOGR)

In machine learning, logistic regression is a widely recognized classification method that is used to predict the probability of a binary-categorical dependent variable [14]. In other words, as [15] states, a logistic regression model will identify relationships between our target feature (will churn/won't churn), and our remaining features to apply probabilistic calculations for determining which class the customer should belong to.

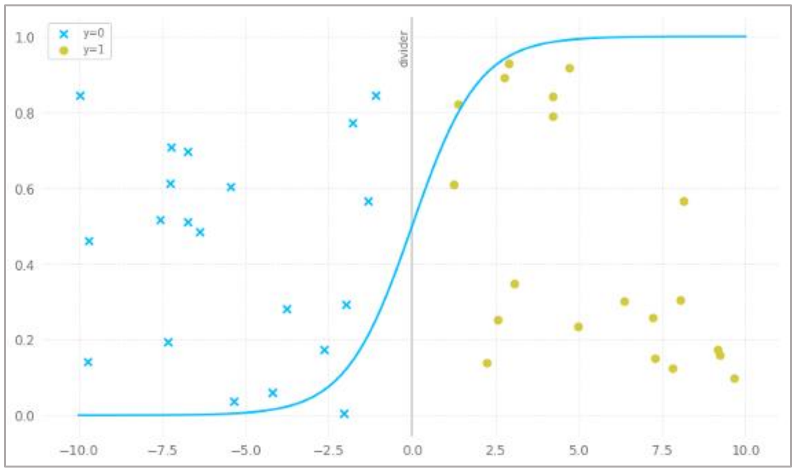


Figure 21: Illustration of the Sigmoid function in some random data (logistic regression for binary classification).
Source: Reprinted from [17]

Evaluate the Machine Learning Models

After training and testing the 3 ML models, the essential model evaluation techniques for predictive binary classification[7] should be applied to find the effectiveness of the algorithm built.

In this case, the recall metric should be the score with more priority. This is because Recall tells us what proportion of clients that actually left the company was predicted by the algorithm before. As defined in [11], Recall is the number of true positives divided by the number of true positives plus the number of false negatives:

$$\text{Recall} = \frac{TP}{TP+FN}$$

Where:

- TP is the number of true positives. It means: "observations that are part of the positive class (has left the company) and that we predicted correctly" [11]
- FN is the number of false negatives (Type II error). It means "observations predicted to be part of the negative class that are actually part of the positive class" [11]. For example, predicting that customer won't cancel their subscription when indeed he is going to cancel.

Consequently, Table 2 shows the results of the performance measures of 3 ML algorithms capable of predicting whether the customer will churn or not.

Model	Recall (s)	Recall (x̄)	Worst Recall Possible	Accuracy (s)	Acuracy (x̄)	Worst Accuracy Possible	Precision (s)	Precision (x̄)	Worst Precision Possible
Decision Tree Classifier	0.08	81.80%	73.80%	0.04	78.70%	74.70%	0.02	77.00%	75.00%
Random Forest Classifier	0.07	83.80%	76.80%	0.03	79.30%	76.30%	0.02	79.60%	77.60%
Logistic Regression	0.03	80.10%	77.10%	0.02	76.80%	74.80%	0.02	75.20%	73.20%

Table 2: Evaluation metrics for Classification models
Source: Own elaboration in Microsoft-Excel to illustrate different performance measures obtained by 3 different machine learning classification algorithms

As observed in Table 2, the real model recall of the Decision Tree Classifier is 81.80% +/- 8%, whereas the rest of the models are 83.80% +/- 7% for Random Forest Classifier and 80.10% +/- 3% for Logistic Regression.

[7] As described in [11], the essential metrics and methods used for assessing the performance of predictive binary classification models, includes: Average Classification Accuracy, Confusion Matrix, Precision, Recall, Specificity and ROC Curve.

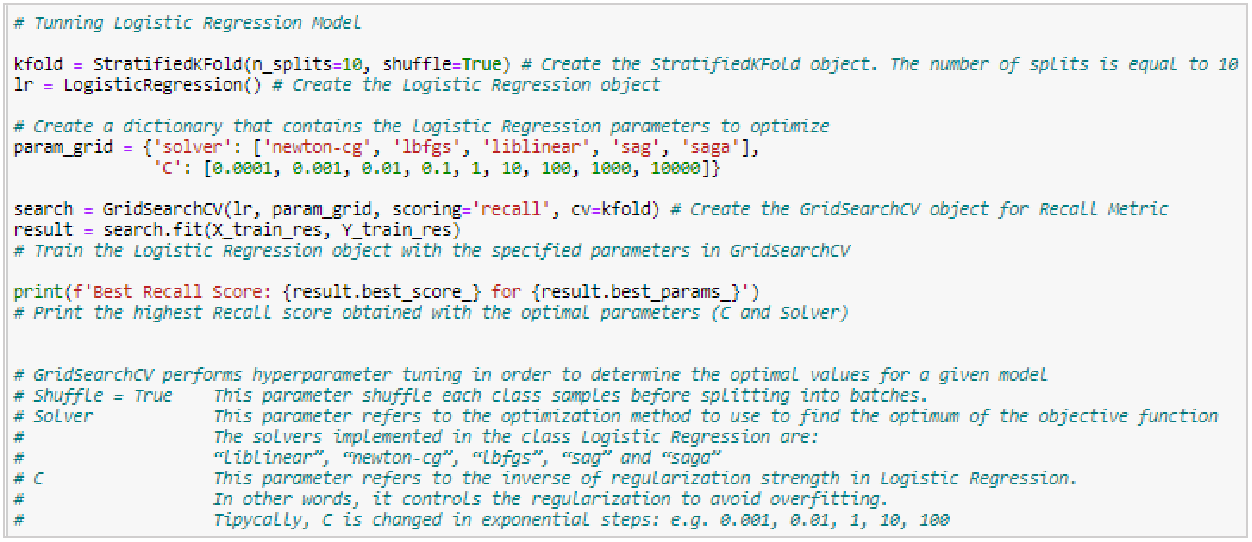
For example, It can be also said that the **logistic regression** has predicted on average **80.10%** of the clients correctly as customers who will cancel their contract based on the historical data.

In other words, if this model analyzes **100** random customers from the whole database, it will be misclassifying only **20 clients** as customers who won't cancel their subscription when indeed they are going to cancel.
This is not too bad if we consider that anything over 50% means the model is better than random. However, we should consider applying different ML algorithms to improve the recall of the model.

Similarly, the above interpretation also applies to the results obtained with the other 2 ML models: (Decision Tree Classifier and Random Forest Classifier).

Model Selection

As shown in Table 2, the Logistic Regression model performed better in terms of recall metric and less variation. In fact, the worst average recall value that can be obtained by applying this model is 0.77 whereas the rest of the models are 0.738 for the Decision Tree Classifier and 0.768 for the Random Forest Classifier.
Under these circumstances, it is better to tune the hyperparameters of Logistic Regression and check if it can deliver even better results. (Figure 22)



```
# Tunning Logistic Regression Model

kfold = StratifiedKFold(n_splits=10, shuffle=True) # Create the StratifiedKFold object. The number of splits is equal to 10
lr = LogisticRegression() # Create the Logistic Regression object

# Create a dictionary that contains the Logistic Regression parameters to optimize
param_grid = {'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
              'C': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]}

search = GridSearchCV(lr, param_grid, scoring='recall', cv=kfold) # Create the GridSearchCV object for Recall Metric
result = search.fit(X_train_res, Y_train_res)
# Train the Logistic Regression object with the specified parameters in GridSearchCV

print(f'Best Recall Score: {result.best_score_} for {result.best_params_}')
# Print the highest Recall score obtained with the optimal parameters (C and Solver)


# GridSearchCV performs hyperparameter tuning in order to determine the optimal values for a given model
# Shuffle = True    This parameter shuffle each class samples before splitting into batches.
# Solver            This parameter refers to the optimization method to use to find the optimum of the objective function
#                   The solvers implemented in the class Logistic Regression are:
#                   "liblinear", "newton-cg", "lbfgs", "sag" and "saga"
# C                 This parameter refers to the inverse of regularization strength in Logistic Regression.
#                   In other words, it controls the regularization to avoid overfitting.
#                   Tipycally, C is changed in exponential steps: e.g. 0.001, 0.01, 1, 10, 100
```

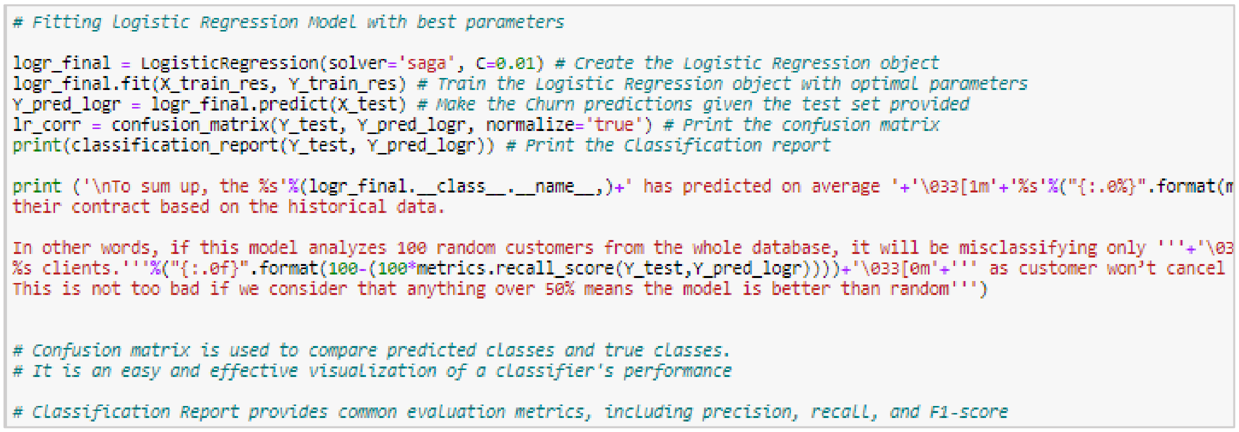
Figure 22: Code used to tune the parameters of the Logistic Regression model.
Source: Own elaboration in Python. **Notes:**

- For further information about this process, please go to the file: ml_models_churnprediction.ipynb
- Only the critical parameters of Logistic Regression has been tuned: “Solver” and “C”

The results obtained after performed the code displayed in Figure 22 indicates that the optimal parameters for the logistic regression algorithm are:

C = 0.01 and Solver = “Saga”

Afterward, another logistic regression model has been created using scikit-learn python library [8], but this time, it has considered the optimal parameters found in the previous section (Figure 23).



```
# Fitting Logistic Regression Model with best parameters

logr_final = LogisticRegression(solver='saga', C=0.01) # Create the Logistic Regression object
logr_final.fit(X_train_res, Y_train_res) # Train the Logistic Regression object with optimal parameters
Y_pred_logr = logr_final.predict(X_test) # Make the Churn predictions given the test set provided
lr_corr = confusion_matrix(Y_test, Y_pred_logr, normalize='true') # Print the confusion matrix
print(classification_report(Y_test, Y_pred_logr)) # Print the Classification report

print ('\nTo sum up, the %s'%(logr_final.__class__.__name__,)+' has predicted on average '+'\033[1m'+'%s'%("{:.0%}".format(m
their contract based on the historical data.

In other words, if this model analyzes 100 random customers from the whole database, it will be misclassifying only '''+'\03
%s clients.'''%("{:.0f}".format(100-(100*metrics.recall_score(Y_test,Y_pred_logr))))+'\033[0m'+''' as customer won't cancel
This is not too bad if we consider that anything over 50% means the model is better than random''')


# Confusion matrix is used to compare predicted classes and true classes.
# It is an easy and effective visualization of a classifier's performance

# Classification Report provides common evaluation metrics, including precision, recall, and F1-score
```

Figure 23: Code used to build the Logistic Regression Model with the optimal hyperparameters.
Source: Own elaboration in Python using scikit-learn library. **Notes:**

- For further information about this process, please go to the file: ml_models_churnprediction.ipynb

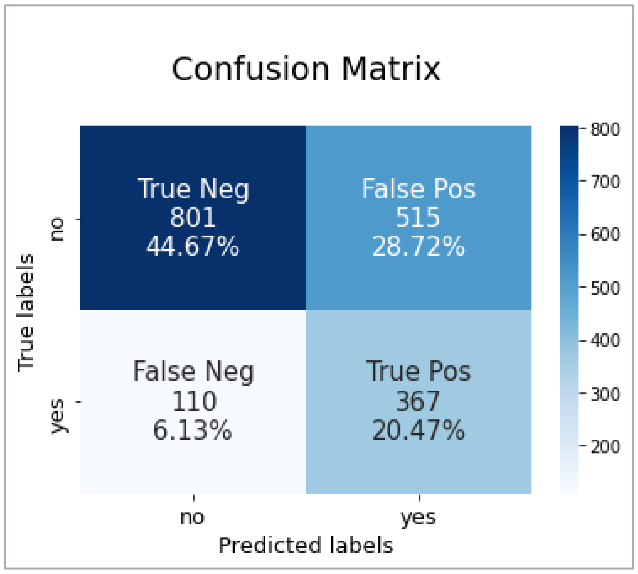
As observed in Figure 24, there is no improvement in the Recall metric after tuning the model. The recall score falls between the Confidence Interval (77%, 83%).

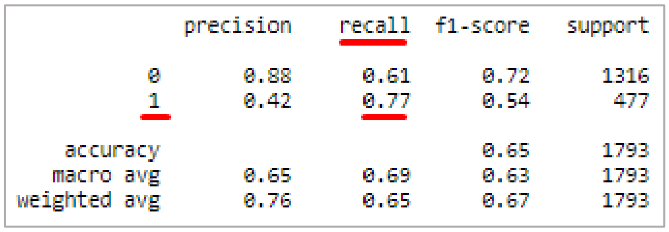
To sum up, the Logistic Regression has predicted on average 77% of the clients correctly as customers who will cancel their contract based on the historical data.

[8] For further information about the python library, please go to the documentation find in: https://scikit-learn.org/stable/

In other words, if this model analyzes 100 random customers from the whole database, it will be misclassifying only 23 clients. as customer won't cancel their subscription when indeed they are going to cancel.

This is not too bad if we consider that anything over 50% means the model is better than random.





```
              precision    recall  f1-score   support

           0       0.88      0.61      0.72      1316
           1       0.42      0.77      0.54       477

    accuracy                           0.65      1793
   macro avg       0.65      0.69      0.63      1793
weighted avg       0.76      0.65      0.67      1793
```

Figure 24: Confusion Matrix and Classification Report for the Logistic Regression with the optimal hyperparameters. **Source:** Own elaboration in Jupyter Notebook using scikit-learn library. **Notes:**

- For further information about this process, please go to the file: ml_models_churnprediction.ipynb

Conclusion

This report presented a Machine Learning model for classification to predict if the Telecom customers will cancel their contract or not (y) given certain features (x). Then 3 supervised learning algorithms (Decision Tree Classifier, Random Forest Classifier, and Logistic Regression) were built and used as predictors. Finally, the comparative experiments were carried out and provided the following results:

- An Exploratory Data Analysis (EDA) allowed us to gain a maximum understanding of the profile of customers who leave the company. This examination affirmed that having a monthly contract, or not having tech support could seriously impact customer churn.
- A new approach based on SMOTE technique to balance the target variable (churn).
- The effectiveness of the 3 ML techniques, in which the Logistic Regression performed better in terms of recall metric and less variation. In fact, the worst average recall value that can be obtained by applying this model is 0.77 whereas the rest of the models are 0.738 for the Decision Tree Classifier and 0.768 for the Random Forest Classifier.

Nevertheless, there are some limitations in the proposed techniques and there are always expected work to be done in the future, such as the following recommendations:

- Perform a different technique to deal with imbalanced data such as the undersampling approach using Tomek Links or penalized learning that increases the cost of classification mistakes on the minority class.
- Built other supervised learning algorithms for classification such as Support Vector Machine (SVM) or Xgboost,
- Perform different techniques in preprocessing such as data standardization, especially if SVM, PCA, or KNN will be implemented as ML models. This will prevent features with wider ranges from dominating the distance metric.
- Tune other Machine Learning models, not only Logistic Regression, and check if *the* algorithm can deliver even better results.
- Tune other parameters with GridSearchCV in any of the 3 ML models.

References

[1] B. Huang, M. T. Kechadi, and B. Buckley, “Customer churn prediction in telecommunications” *Expert Systems with Applications*, vol. 39, no. 1, pp. 1414–1425, 2012

[2] J. Sampson, J.-L. Hu, H.-H. Hsu, C. Hsiao, and H.-Y. Tsao, “Is mobile jumping more efficient? evidence from major Asia-pacific telecommunications firms,” *Asia Pacific Management Review*, vol. 24, no. 2, pp. 190–199, 2019.

[3] Y. Xie, X. Li, E. Ngai, and W. Ying, “Customer churn prediction using improved balanced random forests, ”Expert Systems with Applications, vol. 36, no. 3, pp. 5445–5449, 2009

[4] A. Keramati, H. Ghaneei, and S. M. Mirmo Hammadi, “Developing a prediction model for customer churn from electronic banking services using data mining,” Financial Innovation, vol. 2, no. 1, pp. 1–13, 2016

[5] M. Curi, “Customer Churn in Telecom Segment,” *Towards Data Science.* March 2021 [Online]. Accessed: 2021-03-31. Available: https://towardsdatascience.com/customer-churn-in-telecom-segment-5e49356f39e5

[6] K. Binti Oseman, N. A. Haris, and F. bin Abu Bakar, “Data mining in churn analysis model for the telecommunication industry,” Journal of Statistical Modeling and Analytics Vol, vol. 1, no. 19-27, 2010.

[7] S. V. Nath and R. S. Behara, “Customer churn analysis in the wireless industry: A data mining approach,” *in Proceedings-annual meeting of the decision sciences institute*, vol. 561, 2003, pp. 505–510.

[8] E. Alpaydin, “Introduction to Machine Learning,”. *The MIT Press, Cambridge Massachusetts* Vol 3, 2014.

[9] Bruce, A. Bruce, and P. Gedeck, Practical Statistics for Data Scientists:50+ Essential Concepts Using R and Python. *O’Reilly Media*, 2020

[10] A. Sidath, “Machine Learning Classifiers. What is Classification?” *Towards Data Science.* March 2021 [Online]. Accessed: 2021-03-31. Available: https://towardsdatascience.com/machine-learningclassifiers-a5cc4e1b0623

[11] C. Albon, Machine learning with python cookbook: Practical solutions from preprocessing to deep learning.” *O’Reilly Media*, Inc.”, 2018.

[12] G. Yufeng, “The 7 Steps of Machine Learning.” *Towards Data Science* April 2021 [Online]. Accessed: 2021-04-01. https://towardsdatascience.com/the-7-steps-of-machine-learning-2877d7e5548e

[13] P. Dangeti, Statistics for Machine Learning. *Packt Publishing Ltd*, 2017

[14] J. Pesantez-Narvaez, M. Guillen, and M. Alcañiz. “Predicting motor insurance claims using telematics data – xgboost versus logistic regression,” *Risk,* vol. 7, no. 2, p. 70, 2019.

[15] L. Susan, “Building a Logistic Regression in Python, step by step” *Towards Data Science.* [Online]. Accessed: 2021-04-07. https://towardsdatascience.com/building-a-logisticregression-in-python-step-by-step-becd4d56c9c8

[16] A. Sharma, “Decision Tree vs Random Forest – which algorithm should you use?” *Analytics Vidhya* [Online]. Accessed: 2021-04-07.
https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

[17] D. Hornung, “Binary Classification with Logistic Regression,”. [Online]. Accessed: 2021-04-07
https://towardsdatascience.com/binary-classification-with-logistic-regression
31b5a25693c4