Mini Project 2

Full Name: Miguel Ángel Benítez Alguacil
Student ID: 1900853

Åbo Akademi University

Machine Learning

Index

1. Introduction (problem statement)

The aim of this project is to implement a tree-based approach to predict behaviour to retain customers of a telecommunications company. This company need to understand who is leaving and why. By analysing the data we could suggest a customer retention strategy.

In the next pages I will explain how I have solved the problem using Python 3 libraries and Jupyter Notebook. Also, I will talk about the following:

- Descriptive analysis of the data.
- What is the data about and how to use it to implement the task.
- Explanation of the packages used
- Steps took to perform the task
- Interpreting the output
- Description of the customer retaining strategy
- How efficient is the model?

2. Descriptive analysis of the data

2.1. What is the data about and how to use it

For this project we are using a dataset with **7043 entries**, **one per customer**, recording various data:

- **Gender**: the gender of the customer (female or male).
- **SeniorCitizen**: whether the customer is a senior or not.
- **Partner**: whether the customer has a partner or not.
- **Dependent**: whether the customer has a commitment to stay in the company or not.
- **Tenure**: how long the customer has been in the company.
- **PhoneService**: whether the customer has phone service or not.
- **MultipleLines**: whether the customer has multiple lines, a single line or don't have phone service at all.
- **InternetService**: whether the customer has internet service and if he has it, which type.
- **OnlineSecurity**: whether the customer has online security or not, or no internet service.
- **OnlineBackup**: whether the customer has online backup or not, or no internet service.
- **DeviceProtection**: whether the customer has device protection or not, or no internet service.
- **TechSupport**: whether the customer has tech support or not, or no internet service.
- **StreamingTV**: whether the customer has streaming TV or not, or no internet service.
- **StreamingMovies**: whether the customer has streaming movies or not, or no internet service.
- **Contract**: the kind of contract the customer has (Month-to-month, One year or Two year)
- **PaperlessBilling**: whether the customer receive the bill on paper or not.
- **PaymentMethod**: the kind of payment method the customer has (Electronic check, Mailed check or Bank transfer)

- **MonthlyCharges**: the charges of this month.
- **TotalCharges**: the total charges of this customer.
- **Churn**: whether the customer has left within the last month.

We are given a data set with all this information, so there are **20 columns**, from which we want to predict the **'Churn'** one. To do so, we will use the other **19**.

2.2. Explanation of the packages used

To complete this task I have used the following **packages/libraries**:

- **NumPy:** NumPy is the fundamental package for scientific computing with Python. I used it to count the 0 and 1 within the prediction's array.
- **Pandas:** Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool. I used it to read the data from the Telco-Customer-Churn2.csv file.
- **Sklearn (Scikit-Learn):** It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN. I used to create the Decision Trees, the train and test subsets, the classification report and to create the image with the tree.
- **IPython (Interactive Python):** A command shell for interactive computing in multiple programming languages that offers introspection, rich media, shell syntax, tab completion, and history. I used it to display the images.

```
In [1]: import numpy as np
        %matplotlib inline

        import pandas as pd

        import sklearn
        from sklearn.tree import DecisionTreeClassifier
        from sklearn import tree, metrics, preprocessing
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import classification_report
        from sklearn.tree import export_graphviz

        from IPython.display import Image
```

3. Steps to perform the task.

The first thing we need to do is to **load the data**. To do so we will use pandas, reading from the file using '**;**' as separation.

```
In [2]: # Access the Telco-Customer-Churn2.csv
location = '/home/mike/Documentos/Erasmus/Machine Learning/MiniProject 2/Telco-Customer-Churn2.csv'
info = pd.read_csv(location, sep=';')
info
```

This is the table:

Out[2]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProt
0	Female	0	Yes	No	1	No	No-phone-service	DSL	No	Yes	
1	Male	0	No	No	34	Yes	No	DSL	Yes	No	
2	Male	0	No	No	2	Yes	No	DSL	Yes	Yes	
3	Male	0	No	No	45	No	No-phone-service	DSL	Yes	No	
4	Female	0	No	No	2	Yes	No	Fiber-optic	No	No	
...	...	...	...	...	...	...	...	...	...	...	
7038	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	No	
7039	Female	0	Yes	Yes	72	Yes	Yes	Fiber-optic	No	Yes	
7040	Female	0	Yes	Yes	11	No	No-phone-service	DSL	Yes	No	
7041	Male	1	Yes	No	4	Yes	Yes	Fiber-optic	No	No	
7042	Male	0	No	No	66	Yes	No	Fiber-optic	Yes	No	

7043 rows × 20 columns

As we can see, the **data** is **not prepared** for Python to work with it. We need to change the **String values into numbers** in the following columns: **'gender'**, **'Partner'**, **'Dependents'**, **'PhoneService'**, **'MultipleLines'**, **'InternetService'**, **'OnlineSecurity'**, **'OnlineBackup'**, **'DeviceProtection'**, **'TechSupport'**, **'StreamingTV'**, **'StreamingMovies'**, **'Contract'**, **'PaperlessBilling'**, **'PaymentMethod'** and **'Churn'**

Out[3]:

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	Te
0	0	0	1	0	1	0	1	0	0	2	0	
1	1	0	0	0	34	1	0	0	2	0	2	
2	1	0	0	0	2	1	0	0	2	2	0	
3	1	0	0	0	45	0	1	0	2	0	2	
4	0	0	0	0	2	1	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	1	0	1	1	24	1	2	0	2	0	2	
7039	0	0	1	1	72	1	2	1	0	2	2	
7040	0	0	1	1	11	0	1	0	2	0	0	
7041	1	1	1	0	4	1	2	1	0	0	0	
7042	1	0	0	0	66	1	0	1	2	0	2	

7043 rows × 20 columns

Once we have done this, we can start working with the data.
First we need to **check** whether there are **enough elements** to do the prediction and if there are null values in our data. Using the function "**info**" we can easily know that.

```
In [4]: # Check the values
        info.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   gender            7043 non-null   int8
 1   SeniorCitizen     7043 non-null   int64
 2   Partner           7043 non-null   int8
 3   Dependents        7043 non-null   int8
 4   tenure            7043 non-null   int64
 5   PhoneService      7043 non-null   int8
 6   MultipleLines     7043 non-null   int8
 7   InternetService   7043 non-null   int8
 8   OnlineSecurity    7043 non-null   int8
 9   OnlineBackup      7043 non-null   int8
 10  DeviceProtection  7043 non-null   int8
 11  TechSupport       7043 non-null   int8
 12  StreamingTV       7043 non-null   int8
 13  StreamingMovies   7043 non-null   int8
 14  Contract          7043 non-null   int8
 15  PaperlessBilling  7043 non-null   int8
 16  PaymentMethod     7043 non-null   int8
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   object
 19  Churn             7043 non-null   int8
dtypes: float64(1), int64(2), int8(16), object(1)
memory usage: 330.3+ KB
```

As there are **7043 entries** (more than enough to do a prediction) and no null data, **it is possible to do the prediction** but first we need to **change the type** of the **'TotalCharges'** column to **float64** because we can not work with the Object type.

We need to split the data in **4** different **subsets**: **X_train**, **X_test**, **y_train** and **y_test**. In the X values there will be all the columns except for 'Churn', which will be in the Y.. The **train subsets** are going to be used first to **fit the Decision Tree Classifier Model** and to obtain a **classification report** and see if the data will be good enough for doing the prediction.
The **test subsets** are going to be the ones used for obtaining the **final results**.

After we have the 4 subsets we have to **create a Decision Tree Classifier model** to predict the Churn. I called mine "**DecTree**", also we need to **fit** the model using the training subsets. After this, we can obtain the **classification report**

3.1. Interpreting the output.

This is the classification report:

	precision	recall	f1-score	support
0	0.83	0.81	0.82	1310
1	0.49	0.53	0.51	451
accuracy			0.74	1761
macro avg	0.66	0.67	0.66	1761
weighted avg	0.75	0.74	0.74	1761

If we focus on the "**accuracy**" value we see that is good enough, so we can continue.

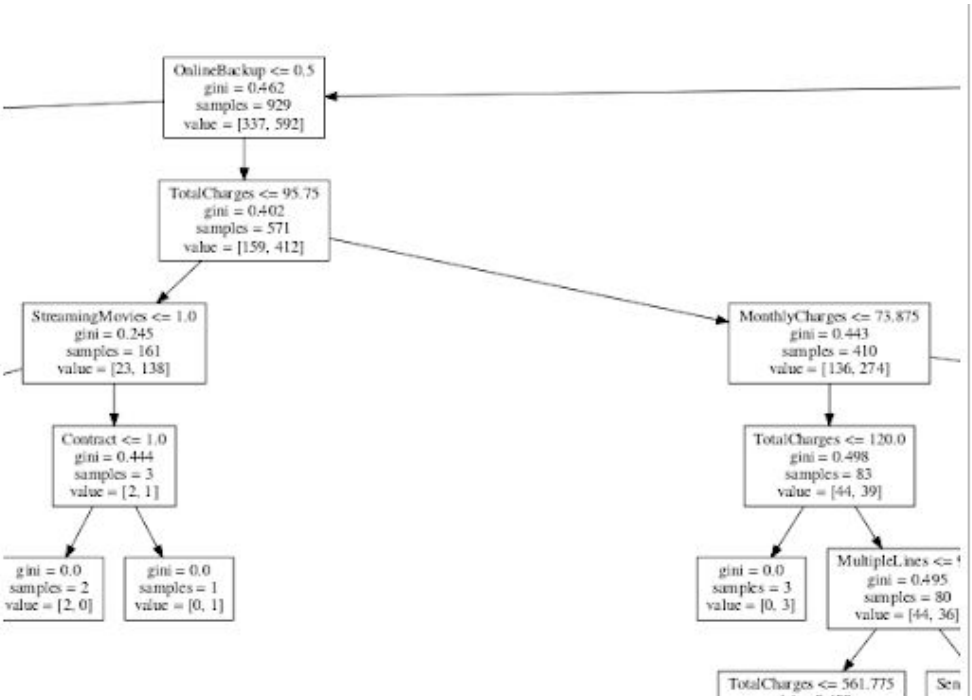
We can know that the **precision** when it predicted that the customer was **not leaving** it was correct the **83%** of the times, and that it predicted it **1310 times** and only 223 (17%) was wrong.
Also, we can see that the accuracy of the “leaving” prediction is not as good, because only the **49%** of the times the prediction was right about that. **451 times** was predicted that the customer was **leaving**, so in the end 230 times (51%) the prediction was wrong!

In the end the prediction was right the 74% of the times, so we can say that it is good enough.

Now it is time to use the **test subsets**. We are going to **predict** the model, obtain the model **score** and the resulting **probabilities**. The **test score is 0.768** which is quite good, so it is going to be a fair prediction.

3.2. Description of the retaining strategy.

The node that presents more proportion of customers leaving is the **OnlineBackup** , that has **929 samples** and **592** of them **left** when they had contracted the online backup or did not have internet service (the condition is that **OnlineBackup<=0.5** so that mean that **the customer do NOT have the service**, because the values in this column were: **0→No; 1 → No Internet Service; 2→ Yes**).
When this **condition is false** (the customer had the service or did not have internet service), in the node below, there are **571** samples of which **412 left** the company because the **TotalCharges** are **less or equal 95.75€**.



This may mean that **customers are not happy with the Online Backup** service or have no internet service, but still **pay too much** as Total Charges reflects. Some of the **measures** to avoid this situation could be **lower prices** or try to **improve the quality of this service**, as it does not meet customer expectations.

3.3. How efficient is the model?

According to the **score**, the model is **quite efficient** because its value is **0.737649063032368** and the **closer** it gets **to 1**, the **better**.

4. Conclusion.

In my opinion, the use of this type of diagram is **really interesting** because it allow you to **see** the **conditions under** which what we want to anticipate **can happen**.

It is a **tool** that if used well can be **very powerful** when we have more **complex data sets**, because with the Logistic Regression it would be more complicated to perform than using a tree.

Also, when the **data has lots of variables** as this one, **searching information** in the tree can be a little tricky and **difficult** because you really have to understand how it works.

As for the project, I would follow the **strategy** that I describe before: to try to **improve** the **Online Backup service** or **lower** the **prices**.