CAPSTONE PROJECT REPORT

Telco data set for customer churn

JANUARY 28, 2021

AL-HUSSAIN TECHNICAL UNIVERSITY
Business Park

Table of Contents

Abstract

The machine learning field, which can be briefly defined as enabling computers make successful predictions using past experiences, has exhibited an impressive development recently with the help of the rapid increase in the storage capacity and processing power of computers. Together with many other disciplines, machine learning methods have been widely employed in bioinformatics. The difficulties and cost of biological analyses have led to the development of sophisticated machine learning approaches for this application area. In this chapter, we first review the fundamental concepts of machine learning such as feature assessment, unsupervised versus supervised learning and types of classification. Then, we point out the main issues of designing machine learning experiments and their performance evaluation. Finally, we introduce some supervised learning methods.

Introduction

I'm writing this report to explain and clarify the data set I used for the Data Science capstone project. This project was assigned by Dr. Ibrahim Abu-alhoul as a capstone project for Data Science certification at Al Hussein Technical University. In this project we were tasked to make an end-to-end Machine Learning project for any data from the Internet. The project was assigned at the end of the training on January 28th, 2021. This report will describe my project, explain the process I used to complete my project, analyze the steps I took in completing my project, and provide two suggestions for future projects.

Problem Statement

The goal of this project is to analyze the customer data to predict if the customer will churn the company or stay in the company.

Description the data

The data set is called Telco dataset. It is for the customer Telco telecommunication company. The data previewed whether customers churn the company or not depending on customer data. The data thus reveals the problems faced by customers who churn the company. The data includes: Customer ID, Gender, Senior Citizen, Tenure, Internet Service Type, Online Backup, Monthly Charges, Total Charges, and Churn Columns. I cleaned these columns, made visualization, and checked the relationship between the columns. I chose this dataset because it's a famous problem for most telecommunication companies in the world, and I found a huge number of notebooks trying to solve this problem for many companies around the world. The dataset reveals the customers who churn the company and customers who stayed in the company.

Used Python Libraries

Data was pre-processed using Pandas and NumPy libraries and the learning/validating process was built with Sklearn. Plots were created using seaborn and matplotlib.

```
#import the required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
%matplotlib inline
```

```
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from imblearn.combine import SMOTEENN
```

Figure 1: Libraries I used in the project

Datasets and Input

The input data consisted of one CSV file which is:

Telco-Customer-Churn.CSV.

It contains full data for Telco customers. In Machine Learning level I split the data to two pieces one for train the model and another piece for testing the model.

Data Distribution

The below figure shows how the target class is distributed.

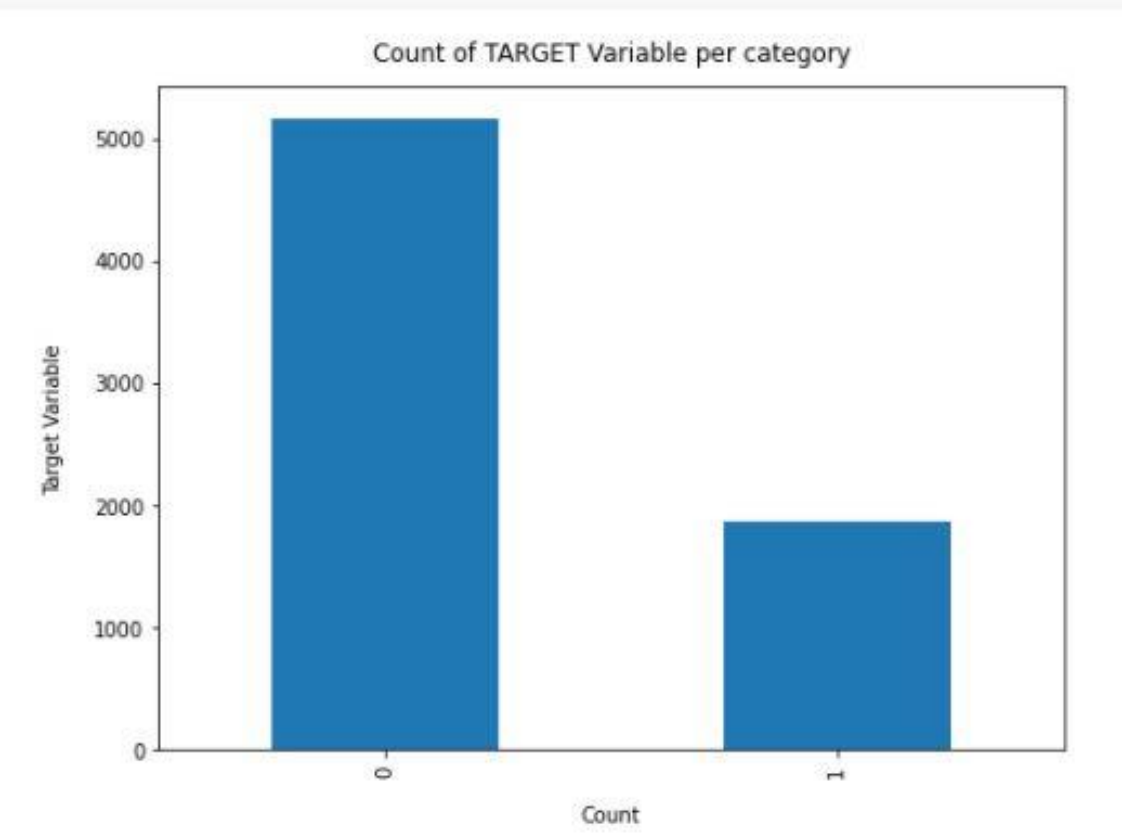


Figure 2: showing the Target variable per category.

Processing the data

The first step in my project was loading the data on my computer. The second step was looking at the data and starting to understand it. Next, I removed or filled the nulls in the data. By that point, I had clean data and I could use it how I wanted. In the next step, I started making visualization on the data and seeing the relationships between the columns using seaborn and matplotlib libraries. Then, I made a train test split using Sk-learn library to prepare the data for building model level. The last step was to build two models called Decision Tree and Random Forest models. Finally, I compared the accuracy of each model to see which model gives me the highest accuracy.

Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

The aim of the following preprocessing is to predict for churn variable. The steps will execute as follows:

1. Cleaning

   - Convert ‘Total Charges’ column to numeric.
   - Remove the nulls.
   - Divide tenure column to bins.
   - Remove columns not required for processing.

2. Data exploration

   - Plot distribution of individual predictors by churn.
   - Convert the target variable 'Churn' in a binary numeric variable.
   - Convert all the categorical variables into dummy variables.
   - Relationship between Monthly Charges and Total Charges.
   - Churn by Monthly Charges and Total Charges.
   - Build a correlation of all predictors with 'Churn'.

1_Cleaning:

1_ Convert 'Total Charges' column to numeric.

```
#convert totalcharges column to numaric because it have numaric values
df.TotalCharges = pd.to_numeric(df.TotalCharges, errors='coerce')
df.isnull().sum()
```

Figure 3: The figure shows how I convert the Total Charges column to numeric.

2_ Remove the nulls.

```
#Removing missing values
newdf.dropna(how = 'any', inplace = True)
```

Figure 4: The figure shows how I remove the nulls from data.

3_ Divide tenure column to bins.

```
[ ]  # Get the max of tenure
     print(newdf['tenure'].max()) #72

     72
```

```
[ ]  # Group the tenure in bins of 12 months
     labels = ["{0} - {1}".format(i, i + 11) for i in range(1, 72, 12)]

     newdf['tenure_group'] = pd.cut(newdf.tenure, range(1, 80, 12), right=False, labels=labels)
```

```
[ ]  newdf['tenure_group'].value_counts()

     1 - 12     2175
     61 - 72    1407
     13 - 24    1024
     49 - 60     832
     25 - 36     832
     37 - 48     762
     Name: tenure_group, dtype: int64
```

Figure 5: The figure shows how I split the Tenure column to bins.

4_ Remove columns not required for processing.

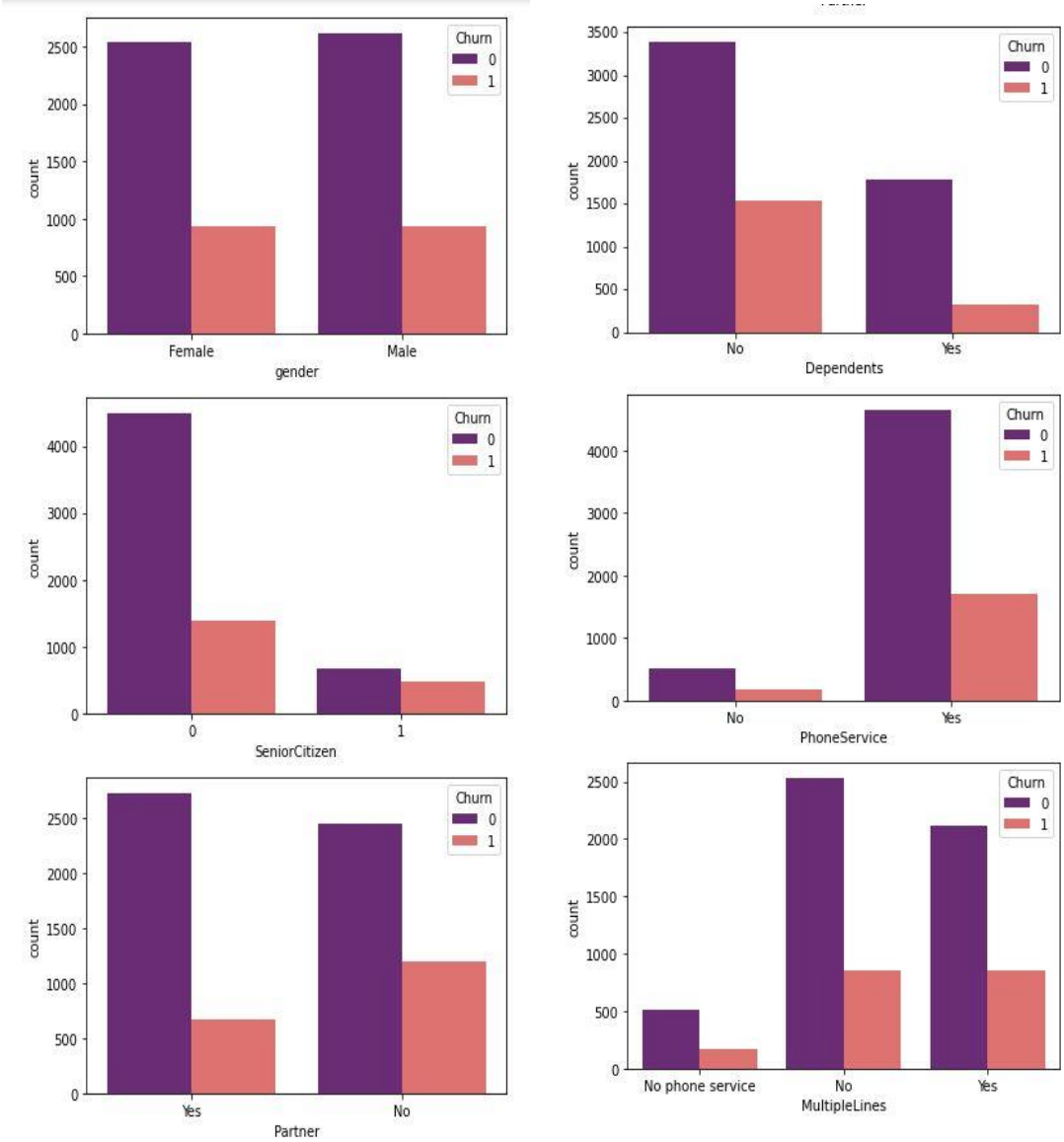
```
#drop column customerID and tenure
newdf.drop(columns= ['customerID','tenure'], axis=1, inplace=True)
newdf.head()
```

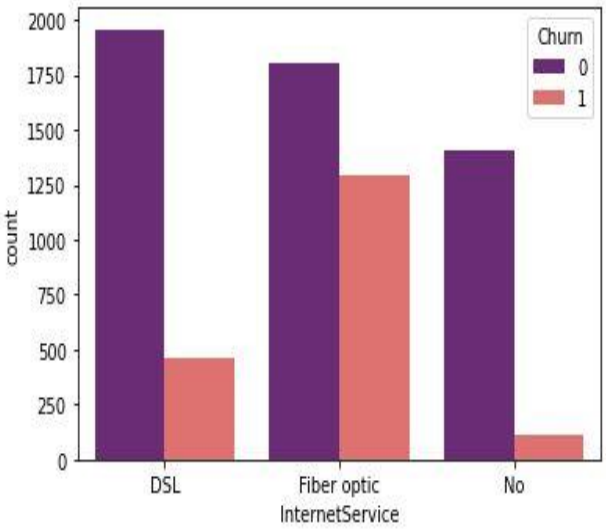
Figure 6: The figure shows how I drop columns not required anymore.

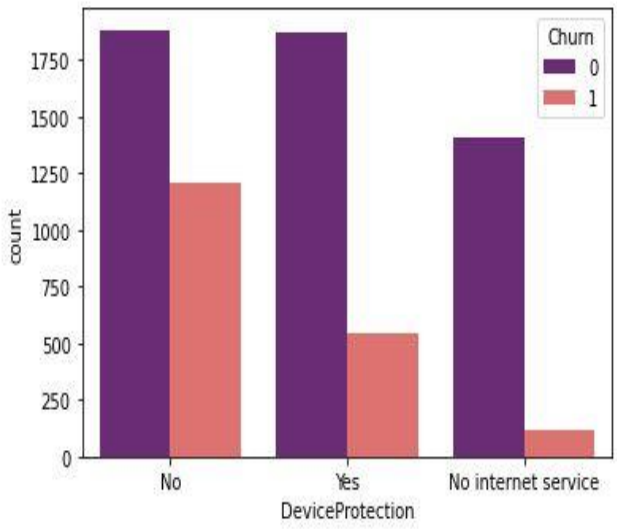
2_Data Exploration:

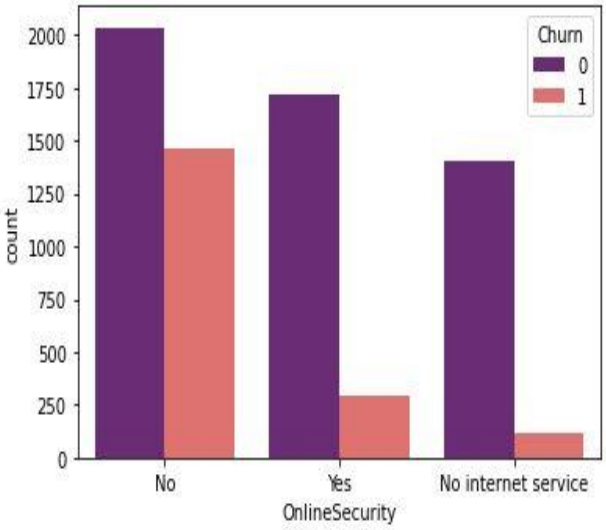
1_ Plot distribution of individual predictors by churn.

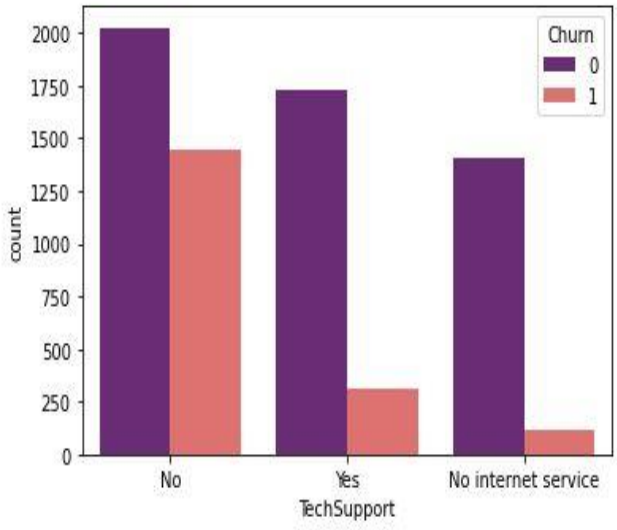
```
for i, predictor in enumerate(newdf.drop(columns=['Churn', 'TotalCharges', 'MonthlyCharges'])):
    plt.figure(i)
    sns.countplot(data=newdf, x=predictor, hue='Churn',palette='magma')
```

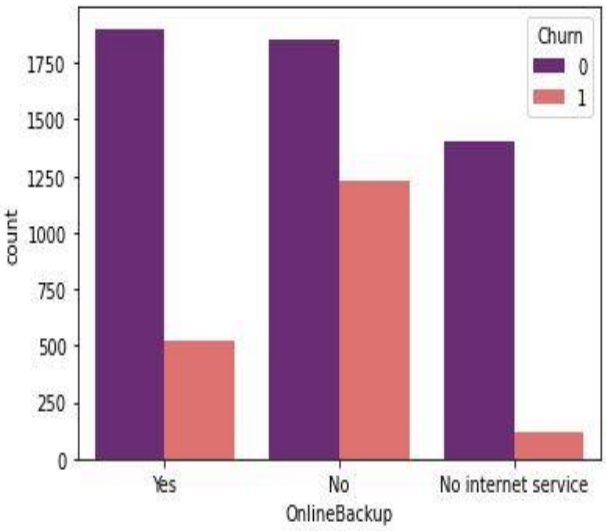


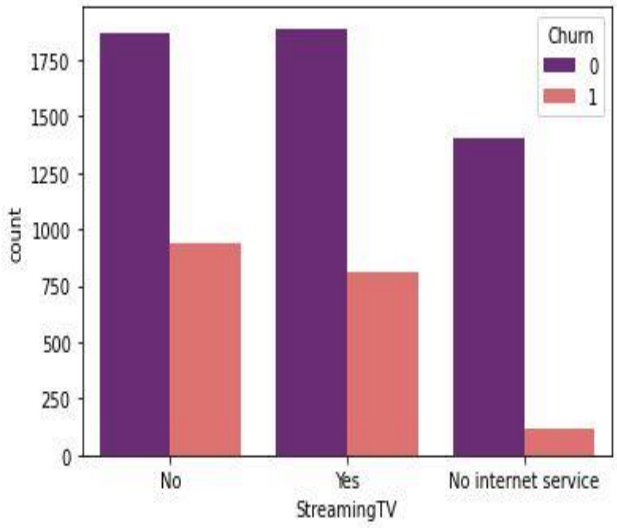

Churn
0
1
count
2000
1750
1500
1250
1000
750
500
250
0
DSL
Fiber optic
No
InternetService


Churn
0
1
count
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
DeviceProtection


Churn
0
1
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
OnlineSecurity


Churn
0
1
count
2000
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
TechSupport


Churn
0
1
count
1750
1500
1250
1000
750
500
250
0
Yes
No
No internet service
OnlineBackup


Churn
0
1
count
1750
1500
1250
1000
750
500
250
0
No
Yes
No internet service
StreamingTV

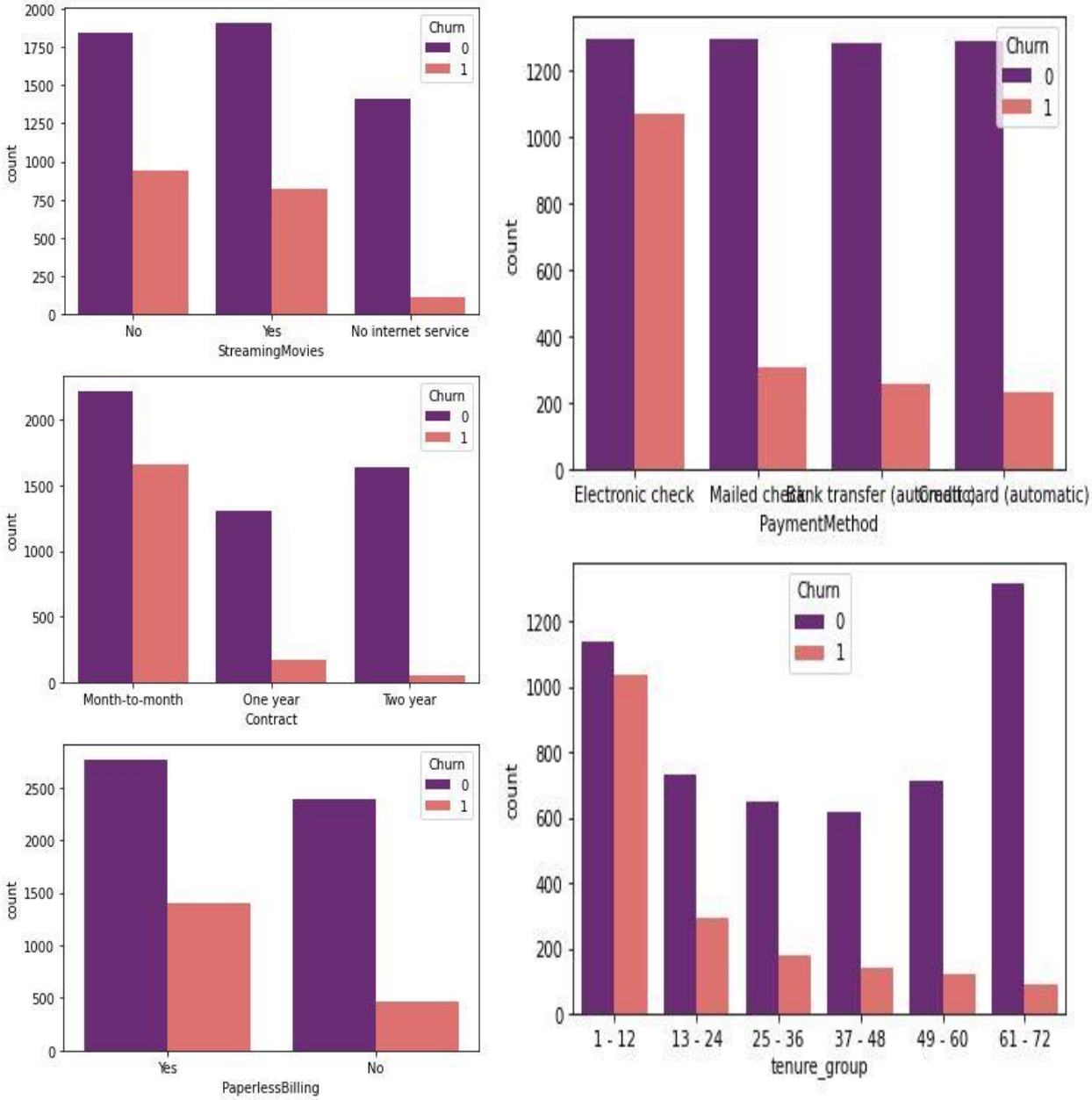


Figure 7: The figure shows how I visualize the data together.

2_ Convert the target variable 'Churn' in a binary numeric variable.

```
[ ] newdf['Churn'] = np.where(newdf.Churn == 'Yes',1,0)
```

```
[ ] newdf.head()
```

tipleLines	InternetService	OnlineSecurity	OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	TotalCharges	Churn
No phone service	DSL	No	Yes	No	No	No	No	Month-to-month	Yes	Electronic check	29.85	29.85	0
No	DSL	Yes	No	Yes	No	No	No	One year	No	Mailed check	56.95	1889.50	0
No	DSL	Yes	Yes	No	No	No	No	Month-to-month	Yes	Mailed check	53.85	108.15	1
No phone service	DSL	Yes	No	Yes	Yes	No	No	One year	No	Bank transfer (automatic)	42.30	1840.75	0
No	Fiber optic	No	No	No	No	No	No	Month-to-month	Yes	Electronic check	70.70	151.65	1

Figure 8: The figure shows how I convert 'churn' column to binary numeric.

3_ Convert all the categorical variables into dummy variables.

```
[ ] dummies_df = pd.get_dummies(newdf)
    dummies_df.head()
```

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneService_No	PhoneService_Yes	MultipleLines_No	MultipleLines_No phone service	M
0	0	29.85	29.85	0	1	0	0	1	1	0	1	0	0	1	
1	0	56.95	1889.50	0	0	1	1	0	1	0	0	1	1	0	
2	0	53.85	108.15	1	0	1	1	0	1	0	0	1	1	0	
3	0	42.30	1840.75	0	0	1	1	0	1	0	1	0	0	1	
4	0	70.70	151.65	1	1	0	1	0	1	0	0	1	1	0	

Figure 9: The figure shows how I convert all categorical variables into dummy variables.

4_ Relationship between Monthly Charges and Total Charges.

```
sns.jointplot(data=dummies_df, x='MonthlyCharges', y='TotalCharges',kind='hex',color="red")
```

```
<seaborn.axisgrid.JointGrid at 0x7f30567770f0>
```

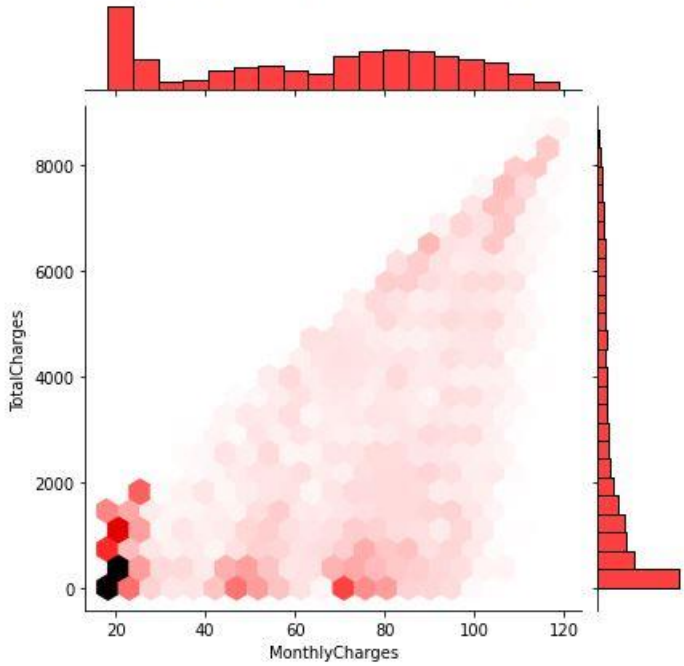


Figure 10: Note here that the Total charges increase as monthly charges increase.

5_ Churn by Monthly Charges and Total Charges.

```
Mth = sns.kdeplot(dummies_df.MonthlyCharges[(dummies_df["Churn"] == 0) ],
                  color="blue", shade = True)
Mth = sns.kdeplot(dummies_df.MonthlyCharges[(dummies_df["Churn"] == 1) ],
                  ax =Mth,color="black", shade= True)
Mth.legend(["No Churn","Churn"],loc='upper right')
Mth.set_ylabel('Density')
Mth.set_xlabel('Monthly Charges')
Mth.set_title('Monthly charges by churn')
```

```
Text(0.5, 1.0, 'Monthly charges by churn')
```

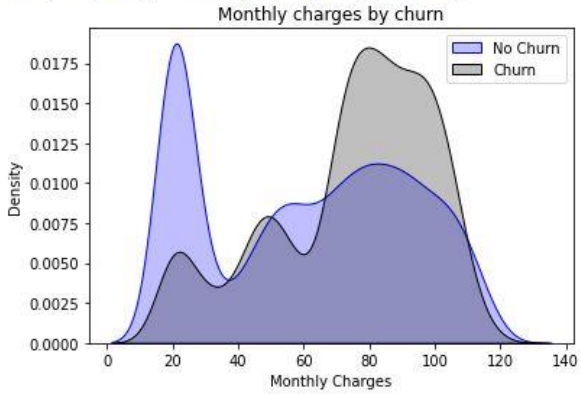


Figure 11: **The figure shows what is the correlation between churn by monthly charges and total**

charges.

6_ Build a correlation of all predictors with 'Churn'.

```
[ ] plt.figure(figsize=(20,8))
    dummies_df.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fdcc590d4a8>
```

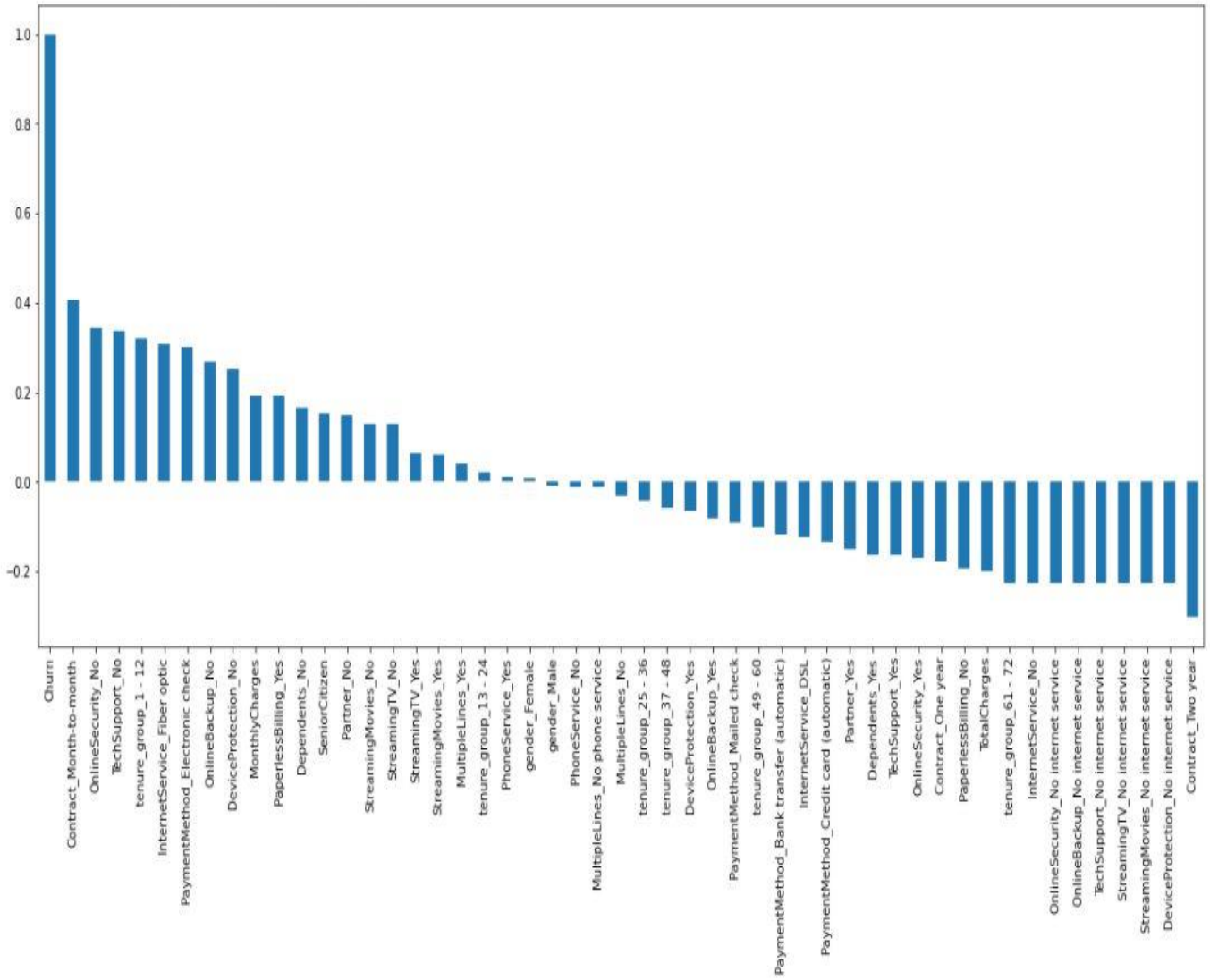


Figure 12: The figure shows the correlations between all predictions with churn.

Analysis

From the beginning, I needed to load the data onto my computer because we can't make anything without loading the data. In the next step, I needed to understand the dataset. This is a very important step for any Data Science project because you must understand the dataset before working with it. The third step is removing the nulls because it will affect the accuracy at the end of the project. The next step was to start making visualization on the data because I needed to know which columns have a strong relationship together to check the redundancy and which columns are not important in the data and don't help me to find more accuracy in the models. The next step is making train test split on the data because without this step I can't build the model and make predictions. The final step is building the models. I made the Decision Tree and Random Forest models because they are two of the best classification models in machine learning and making predictions.

Heat Map

```
plt.figure(figsize=(12,12))
sns.heatmap(dummies_df.corr(), cmap="coolwarm")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f305610ef28>
```

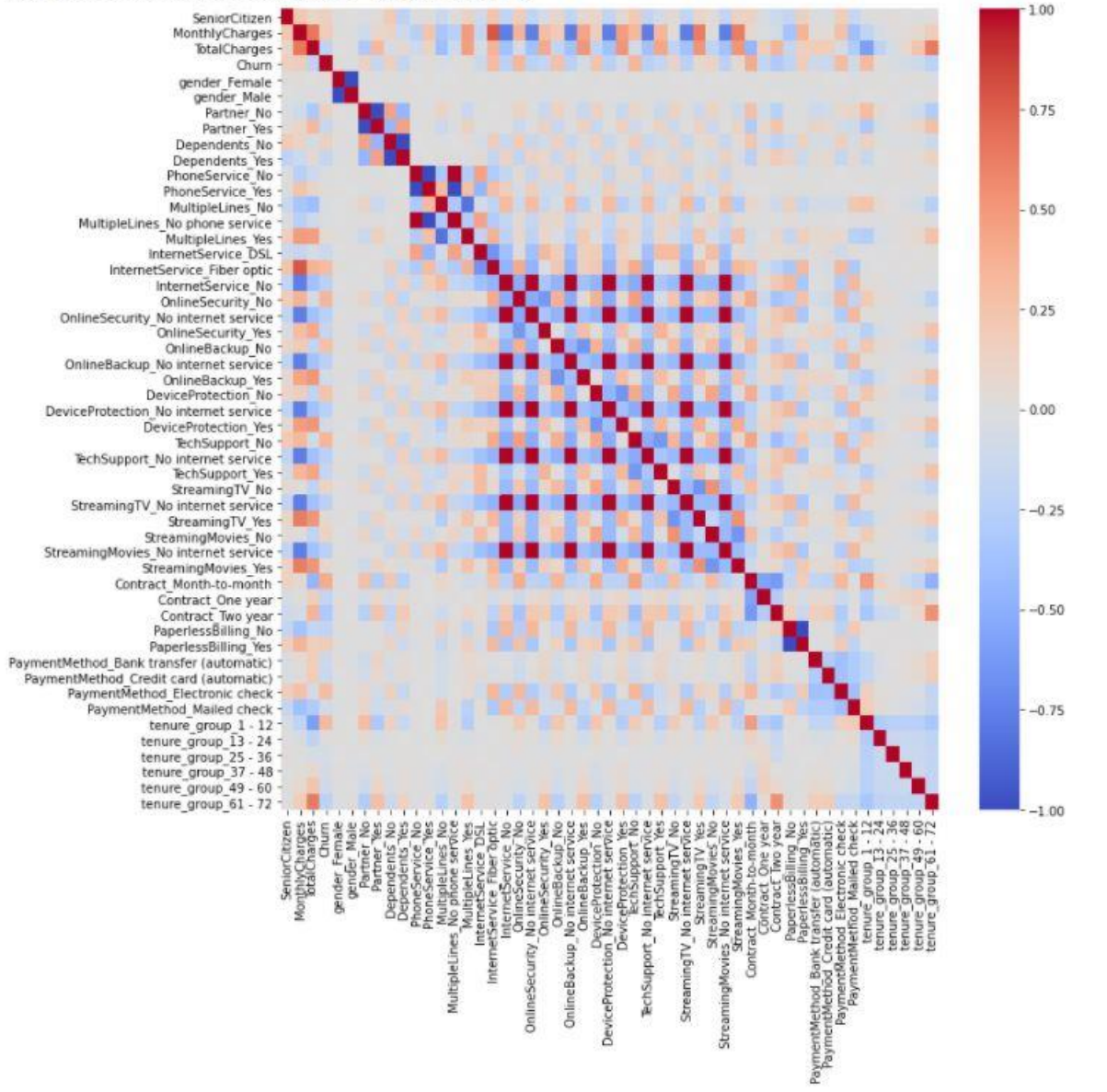


Figure 13: The figure shows the heat map for hole dataset.

Conclusion

In conclusion, I wrote this report to describe the data set and how I used it. In this project I faced many issues in time, accuracy, and data visualization. One of the issues I faced is that the time was not enough for me and for many other trainees to make the project in a perfect way. Another problem I faced in accuracy is that there was one column in the data that I needed to transfer from object type to integer type to remove nulls in it. So, to increase the accuracy, I tried to make it integer. Another problem I faced is the model accuracy; in the beginning, model accuracy was not what I needed from this model, so I tried many models to see what is the best model to give me more accuracy so I could adopt it. Finally, for future projects I suggest giving the trainees more time to make the capstone project and report. The last suggestion is to extend the period of the training and decrease pressure on trainee.

Results

Experiment 1:

1_Decision Tree (Before Up Sampling).

```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

	precision	recall	f1-score	support
0	0.83	0.89	0.86	1044
1	0.61	0.48	0.54	363
accuracy			0.79	1407
macro avg	0.72	0.69	0.70	1407
weighted avg	0.78	0.79	0.78	1407

Figure 14: The figure shows the accuracy for decision tree before Up Sampling.

2_ Decision Tree (After Up Sampling).

```
model_dt_smote.fit(xr_train,yr_train)
yr_predict = model_dt_smote.predict(xr_test)
model_score_r = model_dt_smote.score(xr_test, yr_test)
print(model_score_r)
print(metrics.classification_report(yr_test, yr_predict))
```

```
0.9340753424657534
              precision    recall  f1-score   support

           0       0.94      0.91      0.93       541
           1       0.93      0.95      0.94       627

    accuracy                           0.93      1168
   macro avg       0.94      0.93      0.93      1168
weighted avg       0.93      0.93      0.93      1168
```

Figure 15: The figure shows the accuracy for decision tree after Up Sampling.

Experiment 2:

1_Random Forest (Before Up Sampling)

```
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.83      0.91      0.87      1044
           1       0.64      0.47      0.54       363

    accuracy                           0.79      1407
   macro avg       0.74      0.69      0.70      1407
weighted avg       0.78      0.79      0.78      1407
```

Figure 16: The figure shows the accuracy for Random Forest before Up Sampling.

2_Random Forest (After Up Sampling)

```
print(model_score_r1)
print(metrics.classification_report(yr_test1, yr_predict1))
```

```
0.9525862068965517
              precision    recall  f1-score   support

           0       0.96      0.93      0.95       529
           1       0.95      0.97      0.96       631

    accuracy                           0.95      1160
   macro avg       0.95      0.95      0.95      1160
weighted avg       0.95      0.95      0.95      1160
```

Figure 17: The figure shows the accuracy for Random Forest after Up Sampling.

Future Work

In the classification, I don't achieve enough accuracy. So, In the future I will apply more classification models and I will achieve the best accuracy I can.

References

1_ www.google.com

2_ www.kaggle.com

3_ www.youtube.com

4_ www.github.com