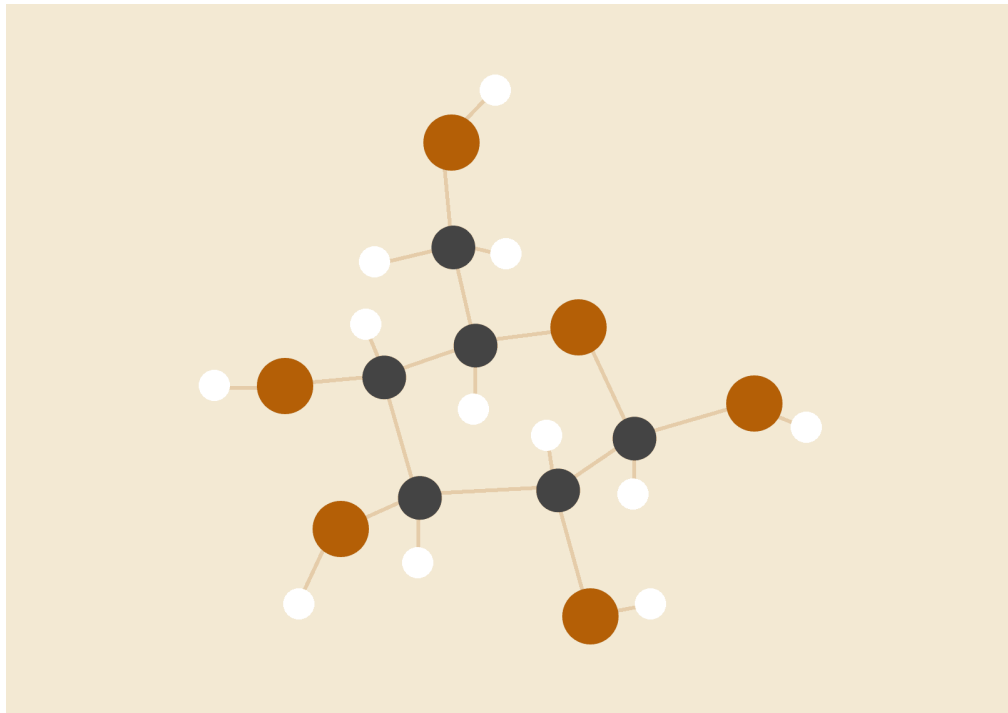


# Customer Churn Machine Learning Project



**Khalil Bagbag**

**Steve Fosso**

**Souhir Arous**

**Hela Ghedass**

**Mohamed Anis Bouaziz**

**Romuald Motcheho**

DATA SCIENCE – CLASSE 4DS5

# Contents

<b>Chapter I: Business Understanding .....</b>	<b>4</b>
<b>Introduction .....</b>	<b>4</b>
<b>1: Project Context .....</b>	<b>4</b>
<b>2: Methodology .....</b>	<b>4</b>
<b>3: Stack .....</b>	<b>4</b>
<b>4: Business objectives related to Customer Churn .....</b>	<b>4</b>
<b>5: Data science goals related to Customer Churn .....</b>	<b>5</b>
<b>Conclusion .....</b>	<b>5</b>
<b>Chapter II: Data understanding.....</b>	<b>6</b>
<b>Introduction .....</b>	<b>6</b>
<b>1: Familiarization with the Dataset .....</b>	<b>6</b>
<b>2: Verifying Data Quality .....</b>	<b>8</b>
<b>Chapter III: Data preparation .....</b>	<b>10</b>
<b>Introduction .....</b>	<b>10</b>
<b>1: Outliers Fixation.....</b>	<b>10</b>
<b>2: Visualization and Correlation Analysis .....</b>	<b>11</b>
<b>3: Data Transformation .....</b>	<b>17</b>
<b>Chapter IV: Modeling and Evaluation .....</b>	<b>20</b>
<b>1-Introduction.....</b>	<b>20</b>
<b>2-Resampling .....</b>	<b>20</b>
<b>3-Models considered .....</b>	<b>20</b>

3-1:Naive Bayes: .....	20
3-2: Decision Tree: .....	22
3-3: k-nearest neighbor: .....	26
3-4: XGBClassifier.....	28
3-5: Logistic Regression.....	29
3-6: Support Vector Machines.....	31
3-7: Random Forest .....	32
<b>4- Model Selection .....</b>	<b>34</b>
4-1: Confusion matrix and classification report .....	34
4-2: ROC curve and AUC score .....	36
4-3: Approved model .....	37
<b>Conclusion .....</b>	<b>39</b>
Chapter V: Deployment .....	41
<b>1-Introduction.....</b>	<b>41</b>
<b>2 -Deployment environment.....</b>	<b>41</b>
<b>3 -Web Application: .....</b>	<b>41</b>

# Chapter I: Business Understanding

## Introduction

We don't know where we are headed. But that's okay, because the essence of a data science project is to start with nothing and end up knowing things even experts cannot deduce on their own. But to be able to do that, our first step should be to isolate problems related to customer churn in order for us to devise a strategy on how to tackle these problems through data science.

### 1: Project Context

Many companies face a lot of issues like Customer faithfulness, Product compliance, Customer preferences and Customer Churn. All of these can be tackled successfully through the application of Data Science and through it, Machine Learning, to exploit the data and give both the customer and the retailer what they require for the growth of their business.

Throughout the project, we will carefully implement the CRISP methodology to come up with the best model for the Customer Churn problem.

### 2: Methodology

Cross Industry Standard Process is a Data Mining (CRISP-DM) process model developed by IBM in the late 1990s and is considered today as among the best solutions for managing Data Science projects (source: Kdnuggets.com).

### 3: Stack

We will be using, in synergy Python along with its machine learning libraries such as scikit learn, pandas and numpy for data manipulation, matplotlib and seaborn for visualization and the Django Framework for Deployment.

### 4: Business objectives related to Customer Churn

Customer churn is a problem that all B2C companies need to monitor. The simple fact is that most organizations have data that can be used to target these individuals and to understand the key drivers of churn. Customer churn refers to the situation when a customer ends their relationship with a company, and it's a costly problem. Customers are the fuel that powers a business. Loss of customers impacts sales. Furthermore, it's much more difficult and costly to gain new customers than it is to retain existing

customers. As a result, organizations need to focus on reducing customer churn. Identifying those customers at risk of leaving, the company can send a special offer to gain back their trust. It could come in the form of a discount coupon or a warm-up message to remind the customer that they are important to the service provider.

### **5: Data science goals related to Customer Churn**

Machine learning is becoming highly sought after to detect Customer Churn. By showing the machine positive cases of churn the model calculates the probability that someone is about to leave. Classifying customers, and predicting whether a customer will churn or not.

### **Conclusion**

The most crucial part in the entire process is to know where we are going. Now that we have our business objectives, we can proceed forward because we have set our targets and our priorities are straight. But we can't push forward if we don't know our field of expertise.

# Chapter II: Data understanding

## Introduction

Where we stand, we have objectives, we know where we're going. We don't know the how, we don't know the when. All we know is our target. First thing to do is to get insight about customer churn to get to know what kind of data we are expecting and how to correctly use it. To do that, we actually need some data first. That's the aim of this step.

### 1: Familiarization with the Dataset

Source: Public dataset in a csv document. The dataset used in this study is based on historical data. It contains 7043 rows with 33 attributes.

Using

`.shape` `.columns` `.dtypes` `.value_counts()`

Attributes	Type	Description	Values
CustomerID	object	Unique number to represent customer	Unique
Count	int64	Status of having a count	Constant {1}
Country	object	Customer Country	Constant {United States} Constant {California}
State	object	Customer State	Different
City	object	Customer City	Different
Zip Code	int64	Customer Zip Code	Different
Lat Long	object	Customer location	Different
Latitude	float64	Location according to latitude	Different
Longitude	float64	Location according to longitude	{Female   Male}
Gender	object	Customer gender	{Yes   No}
Senior Citizen	object	Customer status based on age	{Yes   No}
Partner	object	Customer status based on partner	{Yes   No}

Dependents	object	Customer status based on dependency	Different
Tenure Months	int64	Customer period on using services from Telco company	{Yes   No}
Phone Service	object	Status of having phone service	{Yes   No}
Multiple Lines	object	Status of having multiple lines	{Fibre Optique   DSL
Internet Service	object	Status of having internet service	No}
			{Yes   No}
Online Security	object	Status of having online security service	{Yes   No}
Online Backup	object	Status of having online backup service	{Yes   No}
Device	object	Status of having device protection	{Yes   No}
Protection	object	Status of having technical support	{Yes   No}
Tech Support	object	Status of having streaming tv service	{Yes   No}
Streaming TV	object	Status of having streaming movies	{Month-to-Month
Streaming	object	Status of having contract	2years   1 year}
Movies			{Yes   No}
Contract	object	Status of billing	{Electronic check
	object	Method of payment by customer	Mailed check   Bank
Paperless Billing			transfer   Credit card}
Payment			Different
Method	float64	Customer monthly charges	Different
	object	Customer total charges	{Yes   No}
	object	Status of churn	{1,0}
Monthly	int64	Status of churn	Different
Charges	int64	Customer state of churn	Different
Total Charges	int64	Customer lifetime value	Different
Churn Label	object	Customer churn reason	
Churn Value			
Churn Score			
CLTV			
Churn Reason			

There are 24 attributes categorized as objects. Object type attributes consist of categorical data which put each customer into a certain group. Other types of data in the dataset are int64 and float64 which means the data of the attribute can be calculated.

Churn Rate = 26.54% ; Remaining = 73.46 %

```
(data['Churn Value'].sum()/7043)*100
```

26.536987079369588

Lat Long represents the concatenation of the attributes

Latitude and Longitude.

The attribute Churn Value is similar to Churn Label, it returns the customer status of churn but as a binary number.

```
maskV = data['Churn Value'] == '1'
maskL = data['Churn Label'] == "No"
data[maskV & maskL]
```

0 rows × 33 columns

```
maskV = data['Churn Value'] == '0'
maskL = data['Churn Label'] == "Yes"
data[maskV & maskL]
```

0 rows × 33 columns

## 2: Verifying Data Quality

```
print(data.describe(include='all'))
```

	CustomerID	City	Zip Code	Latitude	Longitude	Gender	Senior	Citizen	Partner	Dependents	Tenure	Months
count	7043	7043	7043.000000	7043.000000	7043.000000	7043		7043	7043	7043	7043.000000	
unique	7043	1129	NaN	NaN	NaN	7043		7043	7043	7043	7043.000000	
top	2862-PFNIK	Los Angeles	NaN	NaN	NaN	2		2	2	2	NaN	
freq	1	305	NaN	NaN	NaN	Male		No	No	No	NaN	
mean	NaN	NaN	93521.964646	36.282441	-119.798880	3555		5901	3641	5416	NaN	
std	NaN	NaN	1865.794555	2.455723	2.157889	NaN		NaN	NaN	NaN	32.371149	
min	NaN	NaN	90001.000000	32.555828	-124.301372	NaN		NaN	NaN	NaN	24.559481	
25%	NaN	NaN	92102.000000	34.030915	-121.815412	NaN		NaN	NaN	NaN	0.000000	
50%	NaN	NaN	93552.000000	36.391777	-119.730885	NaN		NaN	NaN	NaN	9.000000	
75%	NaN	NaN	95351.000000	38.224869	-118.043237	NaN		NaN	NaN	NaN	29.000000	
max	NaN	NaN	96161.000000	41.962127	-114.192901	NaN		NaN	NaN	NaN	55.000000	

	Streaming Movies	Contract	Paperless Billing	Payment Method	Monthly Charges	Total Charges	Churn Value	Churn Score	CLTV	Churn	Reason
	7043	7043	7043	7043	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	
	3	3	2	4	NaN	NaN	NaN	NaN	NaN	NaN	7043
	No	Month-to-month	Yes	Electronic check	NaN	NaN	NaN	NaN	NaN	NaN	21
	2785	3875	4171	2365	NaN	NaN	NaN	NaN	NaN	NaN	False
	NaN	NaN	NaN	NaN	64.761692	2283.300441	0.265370	58.699418	4400.295755	NaN	5174
	NaN	NaN	NaN	NaN	30.090047	2265.000258	0.441561	21.525131	1183.057152	NaN	NaN
	NaN	NaN	NaN	NaN	18.250000	18.800000	0.000000	5.000000	2003.000000	NaN	NaN
	NaN	NaN	NaN	NaN	35.500000	402.225000	0.000000	40.000000	3469.000000	NaN	NaN
	NaN	NaN	NaN	NaN	70.350000	1400.550000	0.000000	61.000000	4527.000000	NaN	NaN
	NaN	NaN	NaN	NaN	89.850000	3786.600000	1.000000	75.000000	5380.500000	NaN	NaN
	NaN	NaN	NaN	NaN	118.750000	8684.800000	1.000000	100.000000	6500.000000	NaN	NaN

All formats and sizes are valid except for the attributes Total Charges and Churn Reason.

Total charges is of type object instead of float and 11 data values are missing.



```

lesId = list()
for cle,value in df['Total Charges'].iteritems():
    if type(value) not in (int,float):
        lesId.append(cle)
x = df[['Total Charges','Monthly Charges','Contract','Tenure Months','Churn Value']]
x.loc[lesId,:]=

```

	Total Charges	Monthly Charges	Contract	Tenure Months	Churn Value
2234		52.55	Two year	0	0
2438		20.25	Two year	0	0
2568		80.85	Two year	0	0
2667		25.75	Two year	0	0
2856		56.05	Two year	0	0
4331		19.85	Two year	0	0
4687		25.35	Two year	0	0
5104		20.00	Two year	0	0
5719		19.70	One year	0	0
6772		73.35	Two year	0	0
6840		61.90	Two year	0	0

73.46% of the values in  
are missing.

column Churn Reason

```
df.isna().mean()
```

Zip Code	0.00000
Lat Long	0.00000
Latitude	0.00000
Longitude	0.00000
Gender	0.00000
Senior Citizen	0.00000
Partner	0.00000
Dependents	0.00000
Tenure Months	0.00000
Phone Service	0.00000
Multiple Lines	0.00000
Internet Service	0.00000
Online Security	0.00000
Online Backup	0.00000

Device Protection	0.00000
Tech Support	0.00000
Streaming TV	0.00000
Streaming Movies	0.00000
Contract	0.00000
Paperless Billing	0.00000
Payment Method	0.00000
Monthly Charges	0.00000
Total Charges	0.00000
Churn Label	0.00000
Churn Value	0.00000
Churn Score	0.00000
CLTV	0.00000
Churn Reason	0.73463
dtype: float64	

# Chapter III: Data preparation

## Introduction

Data preparation is an important process dealing with a number of key issues related to the organizational data environment such as data quality, availability, loading etc.

### 1: Outliers Fixation

For a better and more optimal result we have to make strategic decisions and select the most useful variables for the development of a model. Thus, there are columns we decided to delete.

Since The attributes 'Count', 'Country' and 'State' are constants, we decided to remove them. Same decision for the column 'City', we concluded that it is not necessary since it informs us about the customer's location while we can get it from the attributes 'Zip Code', 'Latitude' and 'Longitude'.

```
df.drop(['Count', 'Country', 'State', 'City'], inplace = True, axis = 1)
```

In the same way, we removed the columns 'CustomerID', 'Churn Reason', 'Churn Label' and 'Lat Long'.

The CustomerID is unique for each customer, has no meaningful impact on our classification model.

73 % of Churn Reason values are missing, the others are all different and don't help us to conclude anything.

Churn Label is similar to Churn Value, there is no point in keeping both so we decided to preserve churn value because of its numerical nature.

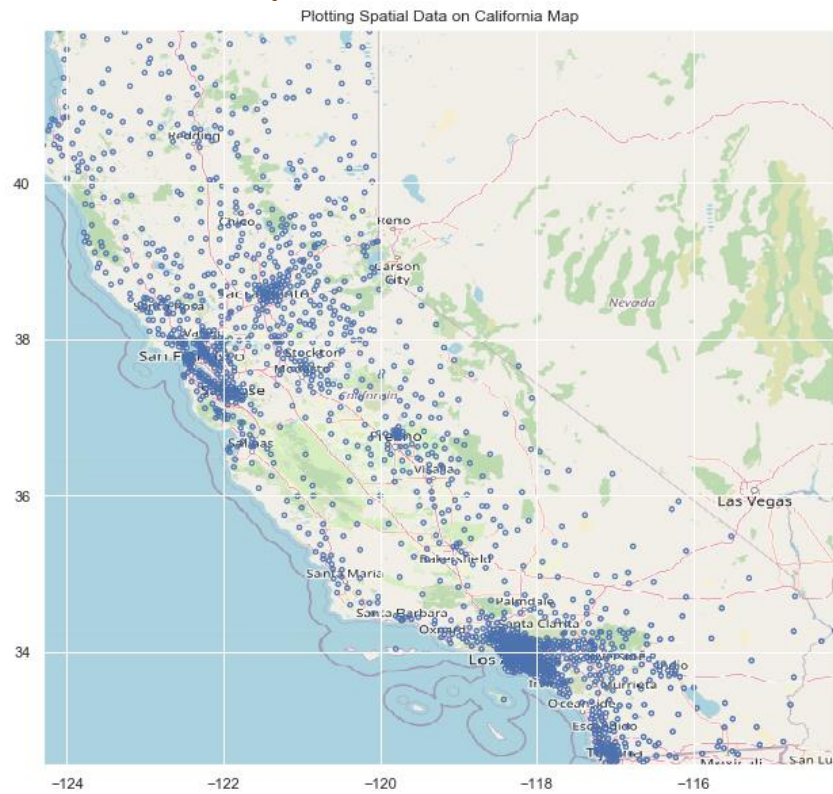
Lat Long is the concatenation of latitude and longitude, it has no added information.

It is necessary to clean up any unwanted data before the next prediction model. Data cleaning lays the groundwork for efficient, accurate and effective data analysis. Without cleaning data beforehand, the analysis process won't be clear or as accurate because the information in the dataset will be unorganized and scattered.

We removed the 11 missing values from the column 'Total Charges' because they are irrelevant compared to the total data. Besides, we converted its type to float.

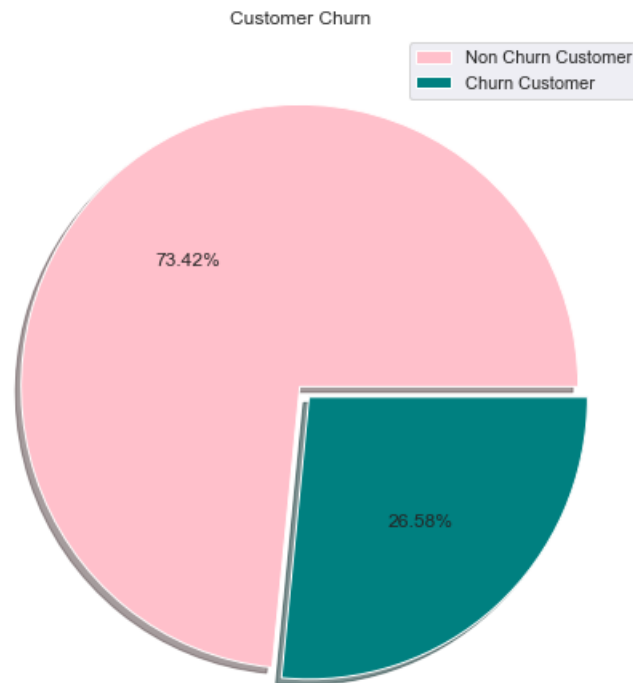
```
df.drop(lesId, axis = 0, inplace = True)
df['Total Charges'] = df['Total Charges'].astype('float')
```

## 2: Visualization and Correlation Analysis

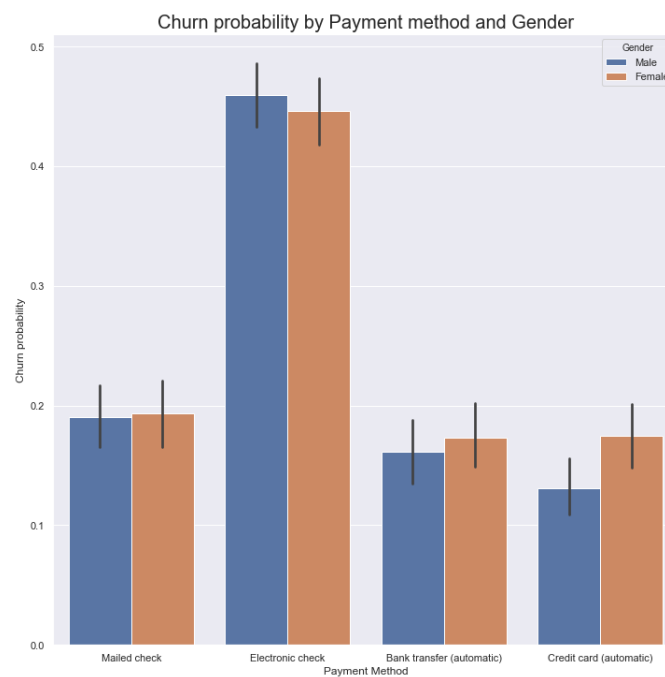


Customers who have left are distributed randomly except in a few cities (but that's simply because there are more customers in those cities).

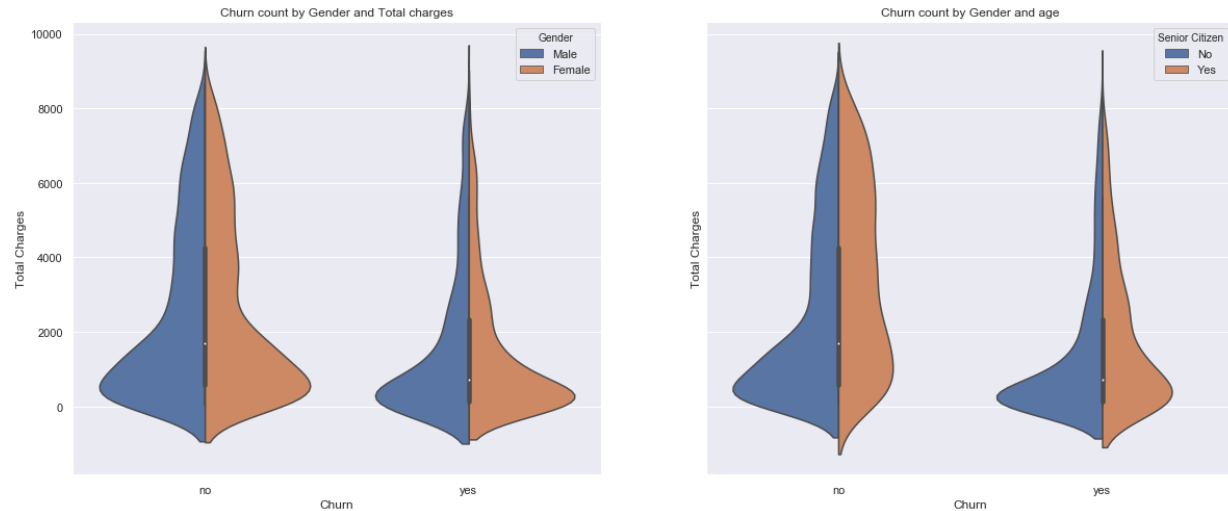
churned represent



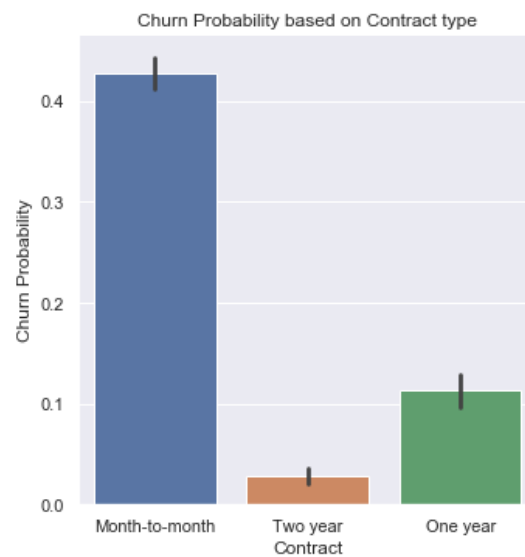
Customers who  
26.58% of the data



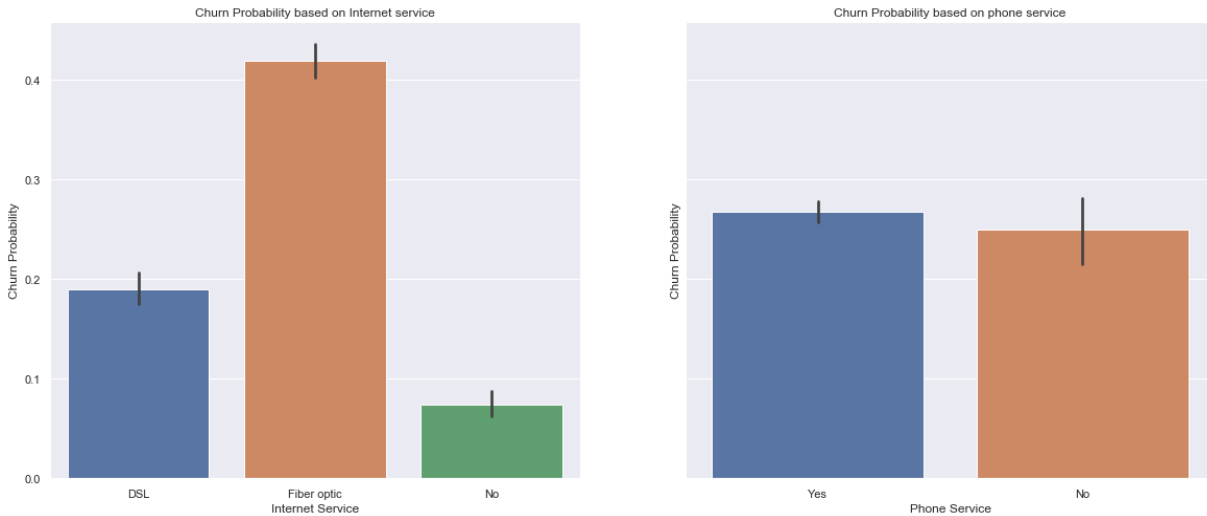
We can see that, no matter their gender, customers who use Electronic check as a payment method have a higher probability of churning.



customers with lower total charges tend to churn more gender has no significant on churn probability  
 younger citizens tend to churn more than senior citizens

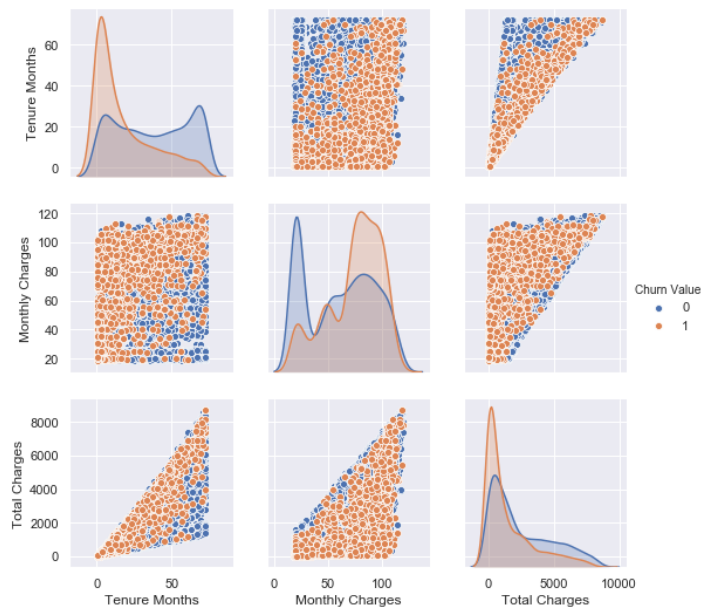


We can see that customers with a month to month contract have a higher chance of churning and those with a long-term contract don't churn à lot

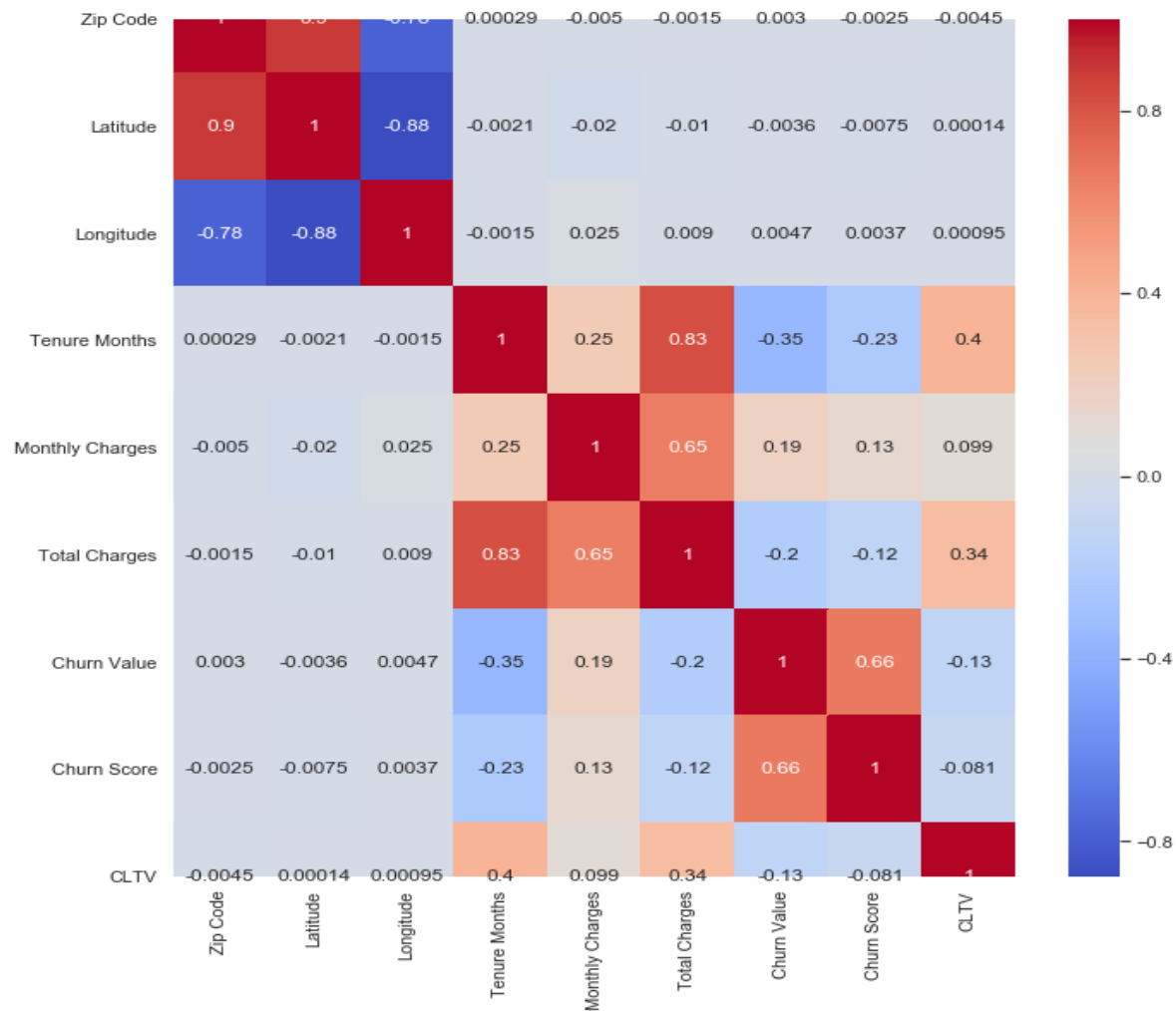


Customers who paid for a fiber optic service surprisingly tend to churn more, and those with no internet service at all churn the least.

Having a phone service has no important impact on churning

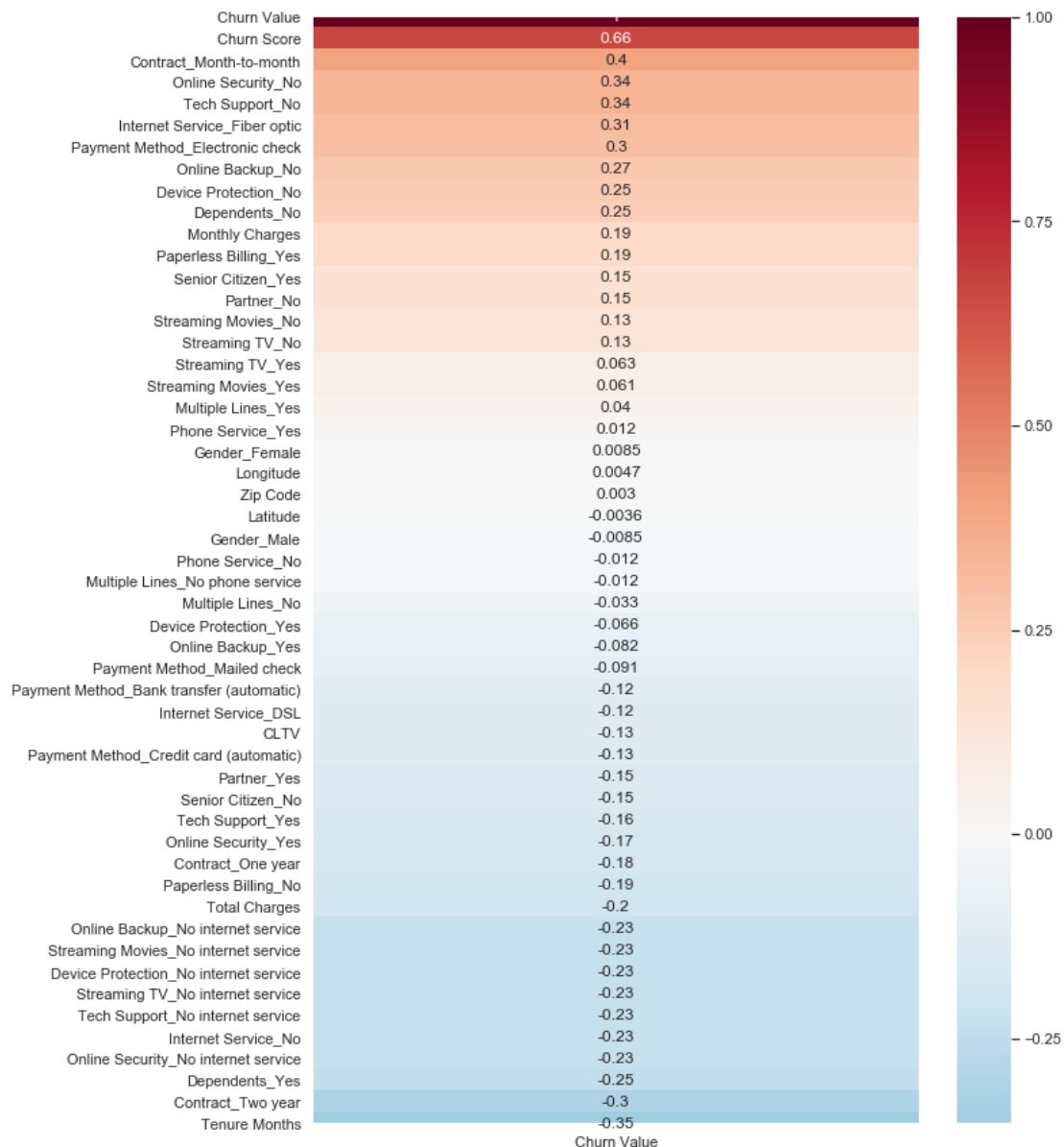


Pair plots allow us to see both distribution of single variables (Tenure, Monthly Charges, Total Charges) and relationships between two variables.



Finally, we wanted to explore the correlation between different charges (correlation matrix) and the relationship between the different categorical data and our target variable (correlation heatmap)

At this stage we decided to remove Latitude and Longitude in favor of keeping Zip code because they are clearly highly correlated.



We can see there's a strong correlation between the churn score and churning and that's because that score is a statistical machine learning prediction value that estimates the state of churn for a given user at any given time, evaluating user demographic info, browsing behavior, and historical purchase data among other signals, so it has a strong influence on our desired output. we also decided to remove CLTV for the same reason.

Let's try to summarize some of the key findings from this part:

- The dataset does not have any missing or erroneous data values.



- Strongest positive correlation with the target features is Monthly Charges and Age whilst negative correlation is with Partner, Dependents and Tenure.
- Most of the customers in the dataset are younger people.
- There are a lot of new customers in the organization (less than 10 months old) followed by a loyal customer base that's above 70 months old.
- Customers with a month-to-month connection have a very high probability to churn that too if they have subscribed to pay via electronic checks.

### 3: Data Transformation

This crucial step concerns transforming the raw data that was collected into a form that can be used in predictive modeling. The first part is encoding our data, machine learning models often take in numerical data as input so our goal is to transform our categorical data to numerical ones.

One approach is pandas' built in function **get\_dummies()** which takes in raw categorical data and turns it into dummy/indicator variables. here it is applied to our dataset:

```
In [4]: df=pd.get_dummies(df)
y=df['Churn Value']
X=df.drop(['Churn Value'],axis=1)
df
```

Out[4]:

Female	...	Streaming Movies_Yes	Contract_Month- to-month	Contract_One year	Contract_Two year	Paperless Billing_No	Paperless Billing_Yes	Payment Method_Bank transfer (automatic)	Payment Method_Credit card (automatic)	Payment Method_Electronic check	Payment Method_Mailed check
0	...	0	1	0	0	0	1	0	0	0	1
1	...	0	1	0	0	0	1	0	0	1	0
1	...	1	1	0	0	0	1	0	0	1	0
1	...	1	1	0	0	0	1	0	0	1	0
0	...	1	1	0	0	0	1	1	0	0	0
1	...	0	1	0	0	1	0	0	1	0	0
0	...	1	1	0	0	0	1	0	0	1	0
0	...	0	1	0	0	1	0	0	0	0	1
0	...	1	1	0	0	0	1	0	0	1	0

To convert our data into model-understandable numerical data, we can also use the Label Encoder class. So all we have to do, to label encode a column, is import the LabelEncoder class from the sklearn library, fit and transform the column, and then replace the existing data with the new encoded one.

```
X = data.iloc[:,:].values
from sklearn.preprocessing import LabelEncoder
labelencoder_X = LabelEncoder()
X[:,0] = labelencoder_X.fit_transform(X[:,0])
X[:,0]
```

```
array([2564, 6511, 6551, ..., 1525, 3367, 2226], dtype=object)
```

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
new_df = data.apply(lambda x: le.fit_transform(x.astype(str)), axis=0, result_type='expand')
new_df
```

	Zip Code	Latitude	Longitude	Gender	Senior Citizen	Partner	Dependents	Tenure Months	Phone Service	Multiple Lines	...	Streaming TV	Streaming Movies	Contract	Paperless Billing	Payment Method	Monthly Charges	Total Charges	Churn Value	Churn Score	CLTV
0	2	327	502	1	0	0	0	12	1	0	...	0	0	0	1	3	740	157	1	70	867
1	4	405	518	0	0	0	1	12	1	0	...	0	0	0	1	2	1033	925	1	49	486
2	5	393	512	0	0	0	1	71	1	2	...	2	2	0	1	2	1578	6104	1	70	2587
3	8	410	528	0	0	1	1	21	1	2	...	2	2	0	1	2	90	2646	1	68	2260
4	13	385	500	1	0	0	1	44	1	2	...	2	2	0	1	0	68	4265	1	73	2556
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	429	587	55	0	0	0	0	70	1	0	...	1	1	2	1	0	356	770	0	26	2528
7039	430	636	244	1	0	1	1	17	1	2	...	2	2	1	1	3	1295	1597	0	41	101
7040	431	625	18	0	0	1	1	70	1	2	...	2	2	1	1	1	59	5698	0	54	2756
7041	432	512	85	0	0	1	1	3	0	1	...	0	0	0	1	2	441	2994	0	41	550
7042	434	601	166	1	0	0	0	63	1	0	...	2	2	2	1	0	107	5407	0	19	2345

7043 rows × 25 columns

However, depending on the data, label encoding introduces a new problem. For example, we have encoded a set of 'Payment Method' into numerical data. This is actually categorical data and there is no relation, of any kind, between the rows.

The problem here is, since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order,  $0 < 1 < 2$ . But this isn't the case at all. To overcome this problem, we use One Hot Encoder.

What one hot encoding does is, it takes a column which has categorical data, which has been label encoded, and then splits the column into multiple columns. The numbers are replaced by 1s and 0s, depending on which column has what value.

```

from numpy import array
from numpy import argmax
from sklearn.preprocessing import OneHotEncoder
values = array(new_df['Payment Method'])
onehot_encoder = OneHotEncoder(sparse=False)
values = values.reshape(len(values), 1)
onehot_encoded = onehot_encoder.fit_transform(values)
print(onehot_encoded)

```

```

[[0. 0. 0. 1.]
 [0. 0. 1. 0.]
 [0. 0. 1. 0.]
 ...
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [1. 0. 0. 0.]]

```

The

second part of our transformation process is

data

normalization, the purpose

of this practice is to change the values in our data into a common scale because for some classification models, features with different ranges could seriously affect its output and accuracy. For our dataset, we aimed for a minmaxscaler under scikit's library which is an estimator that translates each feature such that it is in the given range on the set, e.g [0..1].

```

In [14]: from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X=scaler.fit_transform(X)
X

```

```

Out[14]: array([[3.24675325e-04, 1.49719140e-01, 5.96389800e-01, ...,
0.00000000e+00, 0.00000000e+00, 1.00000000e+00],
[6.49350649e-04, 1.59834702e-01, 5.92963268e-01, ...,
0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
[8.11688312e-04, 1.58636782e-01, 5.94295517e-01, ...,
0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
...,
[3.73863636e-01, 2.13054465e-01, 8.57123496e-01, ...,
1.00000000e+00, 0.00000000e+00, 0.00000000e+00],
[3.74025974e-01, 1.71371546e-01, 7.35723731e-01, ...,
0.00000000e+00, 1.00000000e+00, 0.00000000e+00],
[3.74512987e-01, 1.98707058e-01, 7.04049999e-01, ...,
0.00000000e+00, 0.00000000e+00, 0.00000000e+00]])

```

# Chapter IV: Modeling and Evaluation

## 1-Introduction

In this step, we are going to use the insight we have earned in the data description steps and crystallize them into models that will provide the backbone to the service we have promised to provide.

The task of applying additional intelligence to the data and bring about meaningful prediction models is the purpose of this process.

## 2-Resampling

As a first step we start by splitting our dataset into training and test sets. To attend our object, we used the Python package scikitlearn, that has pre-defined functions for most common machine learning and data analysis algorithms.

## 3-Models considered

At this point we can construct our model. The classification algorithms we chose to implement are:the XGboost,Decision trees,Randomforest, Logistic regression, Support Vector Machines, Naive Bayes and KNN.

Our goal is to get a high level of accuracy in predicting churn, as well as insight into what factors influence it and specially to identify more numbers of churn than no churn user.

Given the ease of setting up a basic model, a common approach is to initialize and train a variety of different models and pick the most performant one as a starting point.

### 3-1:Naive Bayes:

#### 3-1-1: Model Chosen:

First step of the Naive Bayes model was to choose the best model between GaussianNB, BernoulliNB, MultinomialNB that's why we used a loop using in it the cross\_val\_score method to estimate the best model to use in the Naive method prediction. It's explained in the following figure:

```

from sklearn.naive_bayes import MultinomialNB, GaussianNB, BernoulliNB
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

nb = {'gaussian': GaussianNB(),
      'bernoulli': BernoulliNB(),
      'multinomial': MultinomialNB()}
scores = {}
for key, model in nb.items():
    s = cross_val_score(model, X_train, y_train, cv=5, scoring='accuracy')
    scores[key] = np.mean(s)
scores

{'gaussian': 0.7701333333333333,
 'bernoulli': 0.7635555555555555,
 'multinomial': 0.664}

```

### 3-1-2: Training model:

As a second step we fitted our model with the X\_train and y\_train data after scaling and normalization them. After that we obtained a 0.74 as a score model and also as accuracy\_score, as we can figure out here:

```

#Les prédictions
y_pred = modele.predict(X_test)

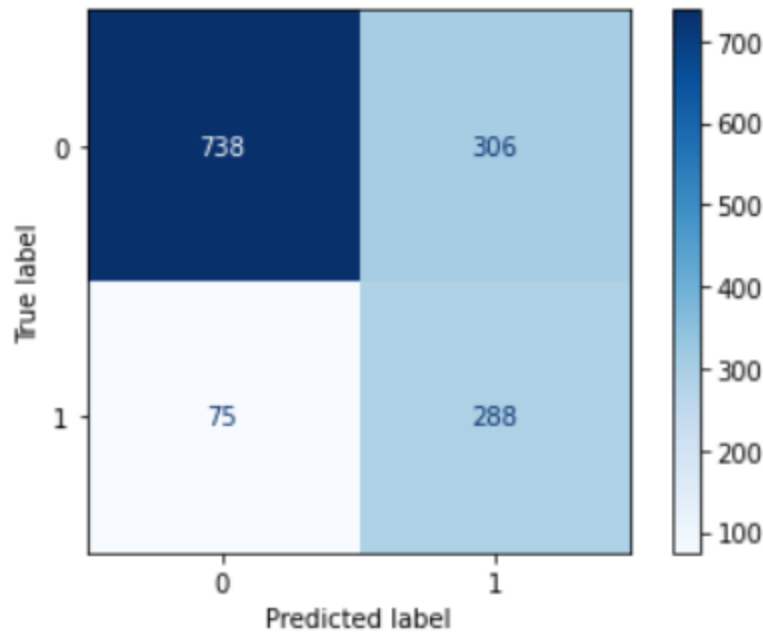
#évaluation de notre modèle
print ("précision: ", accuracy_score(y_test, y_pred))
print ("score: ", modele.score(X_test, y_test))

précision:  0.7420042643923241
score:  0.7420042643923241

```

### 3-1-3: confusion Matrix and classification report:

After that we obtained the confusion matrix with 739 of true negative values, and 288 true positives, 75 true negatives and 306 values of false negatives.



For more information we obtained using the classification report a f1-score of 0.82 for the non churn customer 0.59 for the churn customer.

```
# Classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.91	0.71	0.79	1044
1	0.48	0.79	0.60	363
accuracy			0.73	1407
macro avg	0.70	0.75	0.70	1407
weighted avg	0.80	0.73	0.75	1407

We can conclude that the naive bayes is not a strong model due to the fact that it assumes that all features are independent which is not our case.

### 3-2: Decision Tree:

Decision trees are among the most popular machine learning algorithms given their intelligibility and simplicity.

It's used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

For predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

First thing first, we created a dictionary called 'param\_grid' in order to configure the hyperparameters:

criterion: The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

max\_depth: The maximum depth of the tree.

max\_leaf\_nodes: Max number of leaf nodes - Best nodes are defined as relative reduction in impurity.

Then we created a grid search instance that will test all the combinations of the values of the hyperparameters and give us the best combination which turned out to be [criterion: gini, max\_depth: 6, max\_leaf\_nodes: 17]

```
from sklearn.tree import DecisionTreeClassifier
param_grid = {'criterion': ['gini', 'entropy'],
              'max_depth': np.arange(1,10),
              'max_leaf_nodes': [None, 3,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]
              }

from sklearn.model_selection import GridSearchCV

grid = GridSearchCV(DecisionTreeClassifier(random_state=3), param_grid=param_grid, cv=10 )
grid.fit(X_train, y_train )
grid.best_params_
```

```
{'criterion': 'gini', 'max_depth': 6, 'max_leaf_nodes': 23}
```

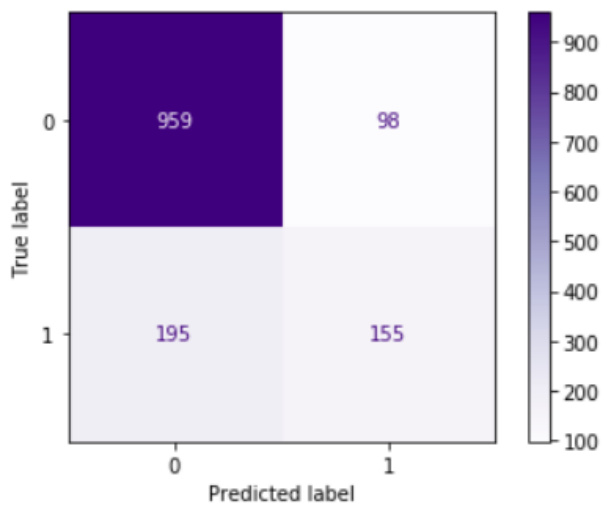
We created an instance of the decision tree classification algorithm called DT, we trained it and got the scores below.

```
DT = DecisionTreeClassifier(random_state=3, criterion='gini', max_depth=6, max_leaf_nodes = 23 )
DT.fit(X_train, y_train)
y_pred = DT.predict(X_test)
```

```
print('Le train score est :', DT.score(X_train, y_train))
print('Le test score est :', DT.score(X_test, y_test))
```

```
Le train score est : 0.8056888888888889
Le test score est : 0.7917555081734187
```

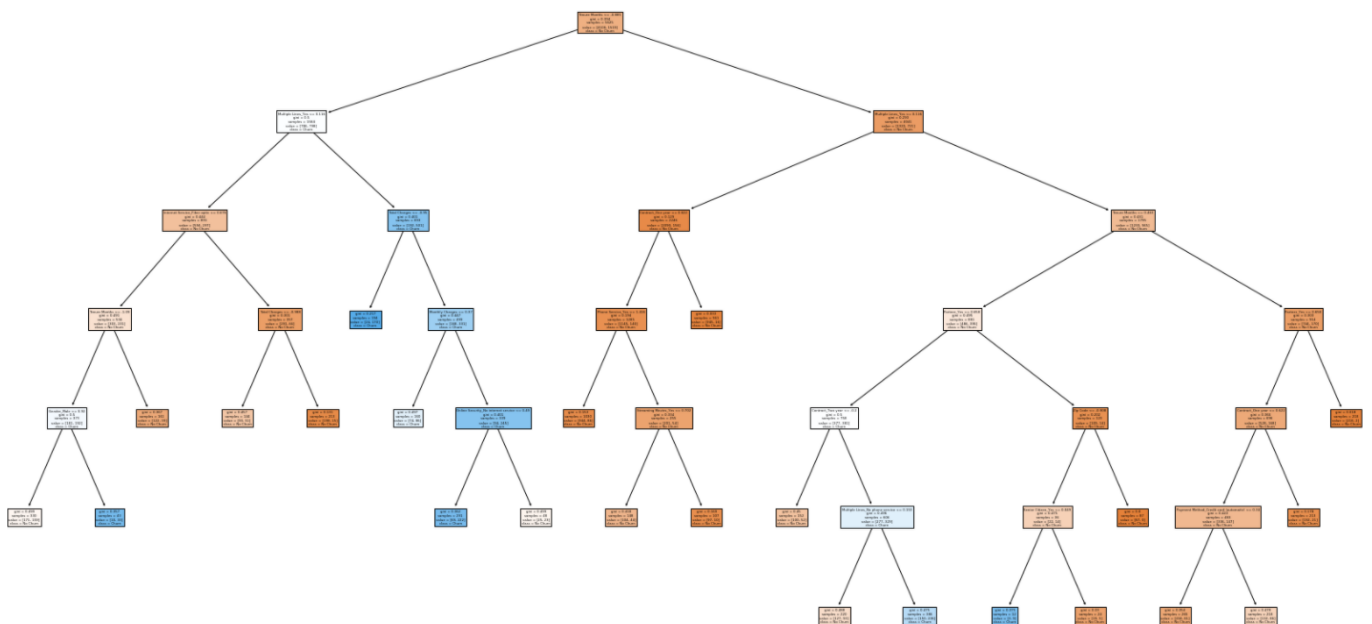
The confusing matrix and the classification report:



	precision	recall	f1-score	support
0	0.831023	0.907285	0.867481	1057
1	0.612648	0.442857	0.514096	350
accuracy	0.791756			1407
macro avg	0.721835	0.675071	0.690788	1407
weighted avg	0.776701	0.791756	0.779574	1407

Model	Accuracy	Precision	Recall	F1 Score	F2 Score
Decision Tree	0.791756	0.612648	0.442857	0.514096	0.468845

The decision tree:



The text report:



```

|--- Tenure Months <= -0.89
|   |--- Internet Service_Fiber optic <= 0.12
|   |   |--- Internet Service_No <= 0.68
|   |   |   |--- Tenure Months <= -1.09
|   |   |   |   |--- Senior Citizen_Yes <= 0.92
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- Senior Citizen_Yes > 0.92
|   |   |   |   |   |--- class: 1
|   |   |   |--- Tenure Months > -1.09
|   |   |   |   |--- class: 0
|   |   |--- Internet Service_No > 0.68
|   |   |--- Churn Value <= -0.99
|   |   |   |--- class: 0
|   |   |--- Churn Value > -0.99
|   |   |   |--- class: 0
|   |--- Internet Service_Fiber optic > 0.12
|   |   |--- Churn Value <= -0.95
|   |   |   |--- class: 1
|   |   |--- Churn Value > -0.95
|   |   |   |--- Monthly Charges <= 0.37
|   |   |   |   |--- class: 1
|   |   |   |--- Monthly Charges > 0.37
|   |   |   |   |--- Online Security_Yes <= 0.49
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- Online Security_Yes > 0.49
|   |   |   |   |   |--- class: 0
|--- Tenure Months > -0.89
|   |--- Internet Service_Fiber optic <= 0.12
|   |   |--- Contract_Two year <= 0.62
|   |   |   |--- Multiple Lines_No phone service <= 1.42
|   |   |   |   |--- class: 0
|   |   |   |--- Multiple Lines_No phone service > 1.42
|   |   |   |   |--- Contract_One year <= 0.70
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- Contract_One year > 0.70
|   |   |   |   |   |--- class: 0
|   |   |   |--- Contract_Two year > 0.62
|   |   |   |   |--- class: 0
|   |--- Internet Service_Fiber optic > 0.12
|   |   |--- Tenure Months <= 0.46
|   |   |   |--- Dependents_Yes <= 0.66
|   |   |   |   |--- Paperless Billing_Yes <= -0.20
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- Paperless Billing_Yes > -0.20
|   |   |   |   |   |--- Multiple Lines_Yes <= 0.15
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Multiple Lines_Yes > 0.15
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- Dependents_Yes > 0.66
|   |   |   |   |--- Zip Code <= -0.91
|   |   |   |   |   |--- Partner_Yes <= 0.05
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- Partner_Yes > 0.05
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- Zip Code > -0.91
|   |   |   |   |   |--- class: 0
|   |   |--- Tenure Months > 0.46
|   |   |   |--- Dependents_Yes <= 0.66
|   |   |   |   |--- Contract_Two year <= 0.62
|   |   |   |   |   |--- Payment Method_Electronic check <= 0.34
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- Payment Method_Electronic check > 0.34
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- Contract_Two year > 0.62
|   |   |   |   |   |--- class: 0
|   |   |   |--- Dependents_Yes > 0.66
|   |   |   |   |--- class: 0

```

The feature importance:

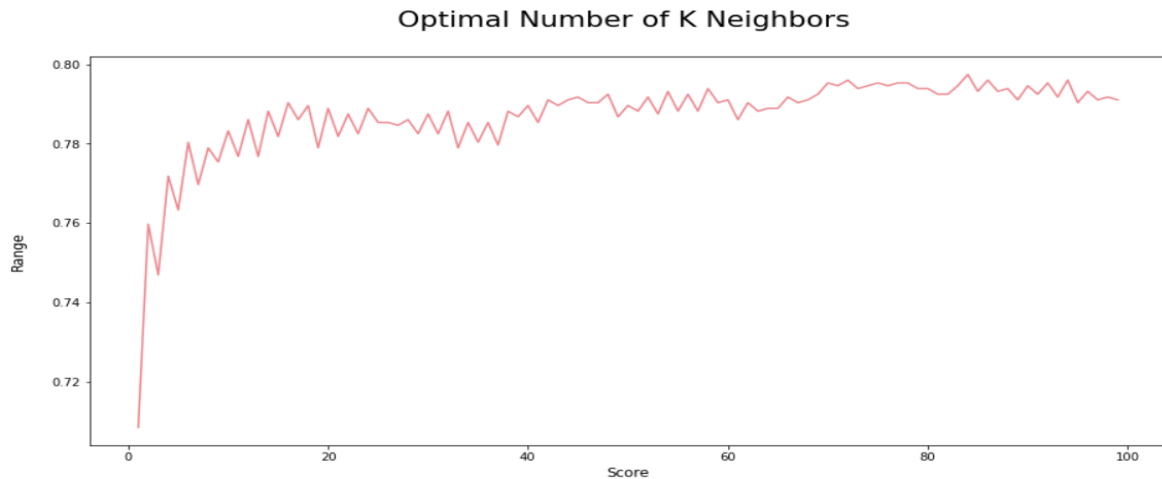
Zip Code 0.010226018098078167	Online Backup 0.013246150661001273
Gender 0.42406582491844524	Device Protection 0.31654846751807864
Senior Citizen 0.009901590617449208	Tech Support 0.03486252312400369
Partner 0.035191532876145334	Streaming TV 0.0
Dependents 0.0	Streaming Movies 0.008807508385603206
Tenure Months 0.008241396814422198	Contract 0.0
Phone Service 0.006232941537788845	Paperless Billing 0.0
Multiple Lines 0.06625279879964321	Payment Method 0.0
Internet Service 0.0	Monthly Charges 0.0
Online Security 0.008930182500802426	Total Charges 0.0

### 3-3: k-nearest neighbor:

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm. It is used to solve **both** classification and regression problems. Its aim is to classify target points (unknown class) according to their distances from points constituting a learning sample.

At first we will plot a graph to find the best value of K that would give us maximum accuracy.

```
from sklearn.neighbors import KNeighborsClassifier
score_array = []
for each in range(1,100):
    knn_loop = KNeighborsClassifier(n_neighbors = each)
    knn_loop.fit(X_train,y_train)
    score_array.append(knn_loop.score(X_test,y_test))
fig = plt.figure(figsize=(15, 7))
plt.plot(range(1,100),score_array, color = '#ec838a')
plt.ylabel('Range\n',horizontalalignment="center", fontstyle = "normal", fontsize = "large")
plt.xlabel('Score\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large")
plt.title('Optimal Number of K Neighbors \n', horizontalalignment="center", fontstyle = "normal",fontsize = "22")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()
```



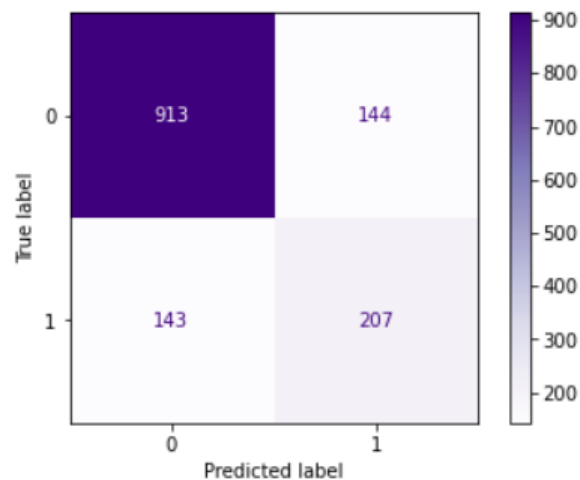
So we can deduce that for a better score k equal to 86.

Then we created an instance of the KNeighborsClassifier called knn with n\_neighbors=86 and then we trained it and got the scores as shown below:

```
knn=KNeighborsClassifier(n_neighbors=86)
knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)
result=pd.DataFrame([[ 'KNN',acc,prec,rec,f1,f2]],columns=[ 'Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])
Models=Models.append(result,ignore_index=True)
classifiers.append(knn)
Models
```

	Model	Accuracy	Precision	Recall	F1 Score	F2 Score
0	KNN	0.79602	0.589744	0.591429	0.590585	0.591091

After that we obtained the confusion matrix with 913 of true negative values, and 144 true positives, 143 true negatives and 207 values of false negatives.



and a classification report:

	precision	recall	f1-score	support
0	0.864583	0.863765	0.864174	1057
1	0.589744	0.591429	0.590585	350
accuracy			0.796020	1407
macro avg	0.727163	0.727597	0.727380	1407
weighted avg	0.796215	0.796020	0.796117	1407

### 3-4: XGBClassifier

XGBoost (**eXtreme Gradient Boosting**) is a library for supervised learning problems in machine learning which implements the [gradient boosting decision tree algorithm](#).

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made.

XGBClassifier uses many decision Trees to make a decision.

We try to use a naives XGBClassifier on our dataset

```
Entrée [179]: model = XGBClassifier()
               model.fit(x_train,y_train)
               print('le train score est {}'.format(model.score(x_train,y_train)))
               print('le test score est {}'.format(model.score(x_test,y_test)))

le train score est 0.8295408370581064
le test score est 0.8165876777251185
```

And we obtain a test score of 81.65%

After that we have tried to turn the parameters of the model with **GridSearchCv**

```
Entrée [40]: param_grid = {
              'max_depth' : np.arange(1,6),
              'min_child_weight' : np.arange(1,6)
            }

Entrée [41]: grid_d = GridSearchCV(XGBClassifier(),param_grid = param_grid,cv = 5,n_jobs = 4)
              grid_d.fit(x_train,y_train)

Out[41]: GridSearchCV(cv=5, estimator=XGBClassifier(), n_jobs=4,
                    param_grid={'max_depth': array([1, 2, 3, 4, 5]),
                                'min_child_weight': array([1, 2, 3, 4, 5])})

Entrée [42]: grid_d.best_params_

Out[42]: {'max_depth': 3, 'min_child_weight': 4}
```

GridsearchCv gave us this parameters:

1. max\_depth: the maximum depth of a tree (3)
2. min\_child\_weight: Minimum sum of instance weight (hessian) needed in a child (4)

But this parameters do not increase our test score

```
Entrée [179]: model = XGBClassifier()
              model.fit(x_train,y_train)
              print('le train score est {}'.format(model.score(x_train,y_train)))
              print('le test score est {}'.format(model.score(x_test,y_test)))

le train score est 0.8295408370581064
le test score est 0.8165876777251185

Entrée [ ]:

Entrée [180]: modele1 = XGBClassifier(max_depth=3,min_child_weight=4)
              modele1.fit(x_train,y_train)
              print('le train score est {}'.format(modele1.score(x_train,y_train)))
              print('le test score est {}'.format(modele1.score(x_test,y_test)))

le train score est 0.8277123120682649
le test score est 0.8151658767772512
```

The test score in our new model is 81.51%,so we will keep our naive XGB model.

## 3-5: Logistic Regression

### 3-5-1: Model Chosen:

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values).

It is used to assign observations to a discrete set of classes using the logistic sigmoid function. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign.

In order for us to apply to our model, we first start by determining the hyperparameters as seen in the following picture. But what are those parameters:

**penalty:** Used to specify the norm used in the penalization

**solver:** Algorithm to use in the optimization problem

**max\_iter:**Maximum number of iterations taken for the solvers to converge.

```
In [87]: from sklearn.model_selection import GridSearchCV
log = LogisticRegression(random_state=0)
param = {'penalty': ['l1', 'l2', 'elasticnet', 'none'],
         'C': np.linspace(0.10, 100),
         'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
         'max_iter': [100, 1000, 2000, 3000]}
grid = GridSearchCV(log, param_grid=param, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
grid.best_params_

Out[87]: {'C': 7.979797979797979,
         'max_iter': 100,
         'penalty': 'l1',
         'solver': 'liblinear'}

In [88]: final_model = LogisticRegression(random_state=0, C=7.979797979797979, penalty='l1', solver='liblinear', max_iter=100)
final_model.fit(X_train, y_train)
final_model.score(X_test, y_test)

Out[88]: 0.8073916133617626
```

The Result shows that for the **penalty**, the **solver** and the **max\_iter** the best values are respectively **100**, **'l1'** and **'liblinear'**.

### 3-5-2: Training model:

As we can see in the same picture, after training our model, we obtained a score of 0.80

```
In [87]: from sklearn.model_selection import GridSearchCV
log = LogisticRegression(random_state=0)
param = {'penalty': ['l1', 'l2', 'elasticnet', 'none'],
         'C': np.linspace(0.10, 100),
         'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
         'max_iter': [100, 1000, 2000, 3000]}
grid = GridSearchCV(log, param_grid=param, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
grid.best_params_

Out[87]: {'C': 7.979797979797979,
         'max_iter': 100,
         'penalty': 'l1',
         'solver': 'liblinear'}

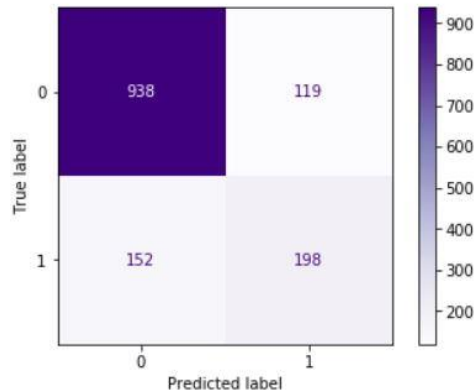
In [88]: final_model = LogisticRegression(random_state=0, C=7.979797979797979, penalty='l1', solver='liblinear', max_iter=100)
final_model.fit(X_train, y_train)
final_model.score(X_test, y_test)

Out[88]: 0.8073916133617626
```

### 3-5-3: Matrix confusion and classification report:

After that we obtained the confusion matrix with **938** true negatives values, and **198** true positives, **119** false positives and **152** values of false negatives.

	precision	recall	f1-score	support
0	0.860550	0.887417	0.873777	1057
1	0.624606	0.565714	0.593703	350
accuracy			0.807392	1407
macro avg	0.742578	0.726566	0.733740	1407
weighted avg	0.801858	0.807392	0.804107	1407



## 3-6: Support Vector Machines

The Support Vector Machine is a supervised learning algorithm mostly used for classification but it can be used also for regression. The main idea is that based on the labeled data (training data) the algorithm tries to find the optimal hyperplane which can be used to classify new data points. In two dimensions the hyperplane is a simple line.

### 3-6-1 Hyper parameter tuning using GridSearchCV

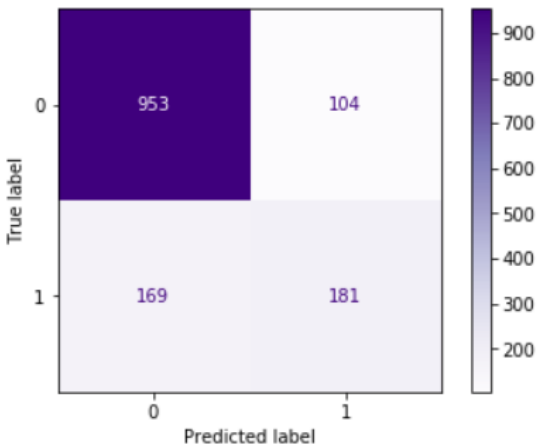
```
from sklearn.svm import SVC
SVM=SVC()
tuned_parameters = [{'kernel': ['rbf'], 'gamma': [1e-3, 1e-4],
                        'C': [1, 10, 100, 1000]},
                    {'kernel': ['linear'], 'C': [1, 10, 100, 1000]}]
grid = GridSearchCV(SVM, param_grid=tuned_parameters, cv=5, n_jobs=-1)
grid.fit(X_train, y_train)
grid.best_params_

{'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
```

### 3-6-2 Classification Report and Confusion Matrix

The confusion matrix gave us **953** of true negatives values, and **104** false positives, **181** true positives and **169** values of false negatives.

	precision	recall	f1-score	support
0	0.849376	0.901608	0.874713	1057
1	0.635088	0.517143	0.570079	350
accuracy			0.805970	1407
macro avg	0.742232	0.709376	0.722396	1407
weighted avg	0.796071	0.805970	0.798933	1407



## 3-7: Random Forest

### 3-7-1: Model Chosen:

Random forest is an ensemble classifier made using many decision tree models. It is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity. Thanks to its well-known ability to provide a great accuracy, we choose, in addition to the other models already tested, to try it on our model.

### 3-7-2: Number of trees:

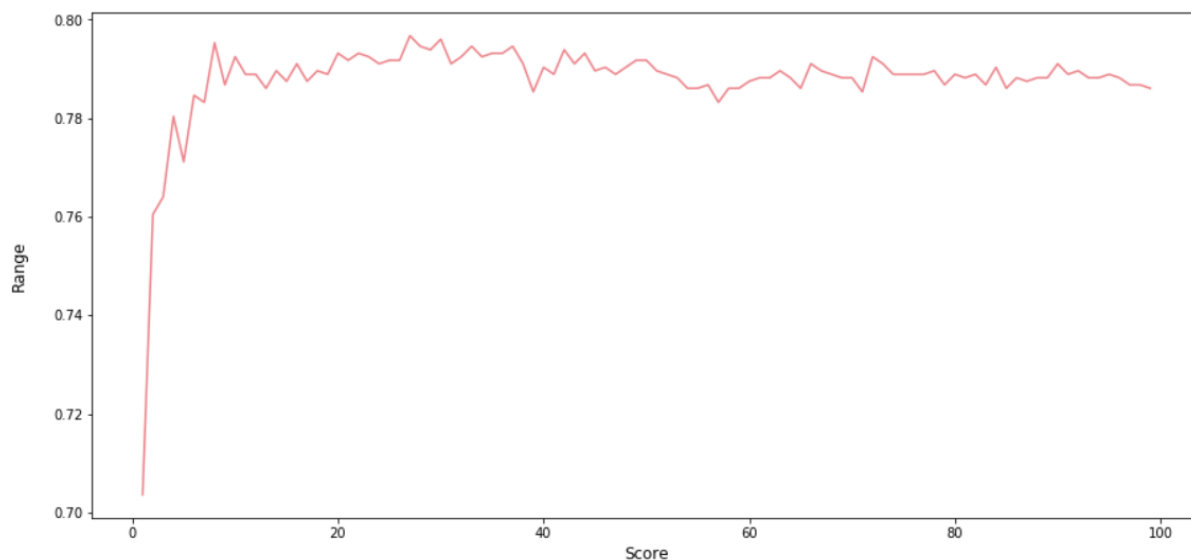
In Order for us to apply the random forest to our data, first we have to determine the optimal number of trees



## Random Forest

```
In [20]: from sklearn.ensemble import RandomForestClassifier
score_array = []
for each in range(1,100):
    rf_loop = RandomForestClassifier(
n_estimators = each, random_state = 1)
    rf_loop.fit(X_train,y_train)
    score_array.append(rf_loop.score(X_test,y_test))
fig = plt.figure(figsize=(15, 7))
plt.plot(range(1,100),score_array, color = '#ec838a')
plt.ylabel('Range\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large")
plt.xlabel('Score\n',horizontalalignment="center",fontstyle = "normal", fontsize = "large")
plt.title('Optimal Number of Trees for Random Forest Model \n',horizontalalignment="center", fontstyle = "normal", fontsize = "22")
plt.xticks(rotation=0, horizontalalignment="center")
plt.yticks(rotation=0, horizontalalignment="right")
plt.show()
```

Optimal Number of Trees for Random Forest Model



And according to this graph. we find it around **20 (n\_estimators)**;

### 3-7-3: Training Model:

The number of trees found, we now start by training our model.

```
In [21]: RF= RandomForestClassifier(n_estimators=30)
RF.fit(X_train, y_train)

y_pred = RF.predict(X_test)
acc = accuracy_score(y_test, y_pred )
prec = precision_score(y_test, y_pred )
rec = recall_score(y_test, y_pred )
f1 = f1_score(y_test, y_pred )
f2 = fbeta_score(y_test, y_pred, beta=2.0)
result=pd.DataFrame([[ 'Random Forest',acc,prec,rec,f1,f2]],columns=[ 'Model', 'Accuracy', 'Precision', 'Recall', 'F1 Score', 'F2 Score'])
Models=Models.append(result,ignore_index=True)
classifiers.append(RF)
Models
```

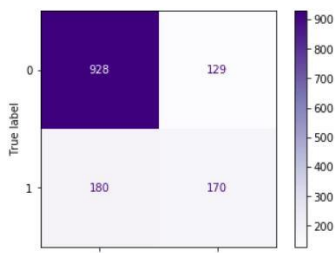
```
Out[21]:
```

	Model	Accuracy	Precision	Recall	F1 Score	F2 Score
0	KNN	0.796020	0.589744	0.591429	0.590585	0.591091
1	Logistic Regression	0.807392	0.624606	0.585714	0.593703	0.576587
2	Naive Bayes	0.732054	0.476844	0.794286	0.595927	0.700958
3	Decision Tree	0.791756	0.612648	0.442857	0.514096	0.468845
4	Random Forest	0.780384	0.568562	0.485714	0.523883	0.500294

As shown in the image, we found that our model has given a score of **0.78** next to **KNN,Decision Tree** and **Logistic Regression**.

```
In [22]: plot_confusion_matrix(RF,X_test,y_test,cmap="Purples")
y_pred=RF.predict(X_test)
print(classification_report(y_test, y_pred,digits=6))
```

	precision	recall	f1-score	support
0	0.837545	0.877956	0.857275	1057
1	0.568562	0.485714	0.523883	350
accuracy			0.780384	1407
macro avg	0.703053	0.681835	0.690579	1407
weighted avg	0.770634	0.780384	0.774342	1407



### 3-7-4: Confusion matrix:

The confusion matrix gave us **928** true negative values, **170** true positive, **129** false positives and **180** values of false negatives.

## 4- Model Selection

### 4-1: Confusion matrix and classification report

The final step in process is to evaluate the models that we build in the modeling step and to make a comparison between them in order to choose the more suitable one for our business objective.

Because we are dealing with a classification problem, we choose the confusion matrix as a first metric to evaluate the trained models.

The confusion matrix is based on representing true positive users and false positive users. Less False positive values the model have better its performance will be.

	Predicted <b>0</b>	Predicted <b>1</b>
Actual <b>0</b>	TN	FP
Actual <b>1</b>	FN	TP

1 — **Accuracy**: One of the more obvious metrics, it is the measure of all the correctly identified cases. It is most used when all the classes are equally important.

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{(\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative})}$$

2 — **Precision**: It is implied as the measure of the correctly identified positive cases from all the predicted positive cases. Thus, it is useful when the costs of False Positives is high..

$$\text{Precision} = \frac{|\text{True Positive}|}{|\text{True Positive}| + |\text{False Positive}|}$$

3 — **Recall**: It is the measure of the correctly identified positive cases from all the actual positive cases. It is important when the cost of False Negatives is high.

$$\text{Recall} = \frac{|\text{True Positive}|}{|\text{True Positive}| + |\text{False Negative}|}$$

4 — The **F<sub>1</sub>** score is the **harmonic mean** of the precision and recall

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

Accuracy is used when the True Positives and True negatives are more important while F1-score is used when the False Negatives and False Positives are crucial.

In our case, we want to minimize the False negatives, so we will consider F1- score as our main metric, we can also use the F2-score which is an example of the Fbeta-measure with a beta value of 2.0.

It has the effect of lowering the importance of precision and increase the importance of recall.

$$F_{\beta} = \frac{(1 + \beta^2) \cdot \text{true positive}}{(1 + \beta^2) \cdot \text{true positive} + \beta^2 \cdot \text{false negative} + \text{false positive}} .$$

## 4-2: ROC curve and AUC score

### 4-2-1 ROC curve

When TP rate is plotted as against FP rate, one obtains a receiver operating characteristics (ROC) graph. Each classifier is represented by a point on ROC graph. A perfect classifier is represented by point (0,1) on ROC graph which classifies all positive and negative instances correctly with 100% TP rate and 0% FP rate. The diagonal line demonstrates classification that is based completely on random guesses.

In that case, one can achieve the desired TP rate but unfortunately also gain equally high FP rate.

The major goal of churn prediction is to detect churn. Therefore, a suitable classifier is the one having high TP rate and low FP rate given that churn is the positive class. Such classifier is located at the upper left corner of ROC graph.

The next figure shows our ROC curve. The black points are random guess classifiers. The other colors points represent the performance of the other classifiers. The higher area, the more conservative a classifier becomes. Conservative classifiers locate at the lower part of ROC graph. In contrast, the lower area, the more liberal a classifier becomes. Liberal classifiers locate at the upper part of ROC graph. Given two ROC curves, the one that is further to the left of the random diagonal is preferred. For this reason, area under ROC curve (AUC), a quantity that measures the overall average performance of a classifier is introduced. The advantage of AUC is unlike many other evaluation metrics such as the overall.

### 4-2-2 AUC

The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

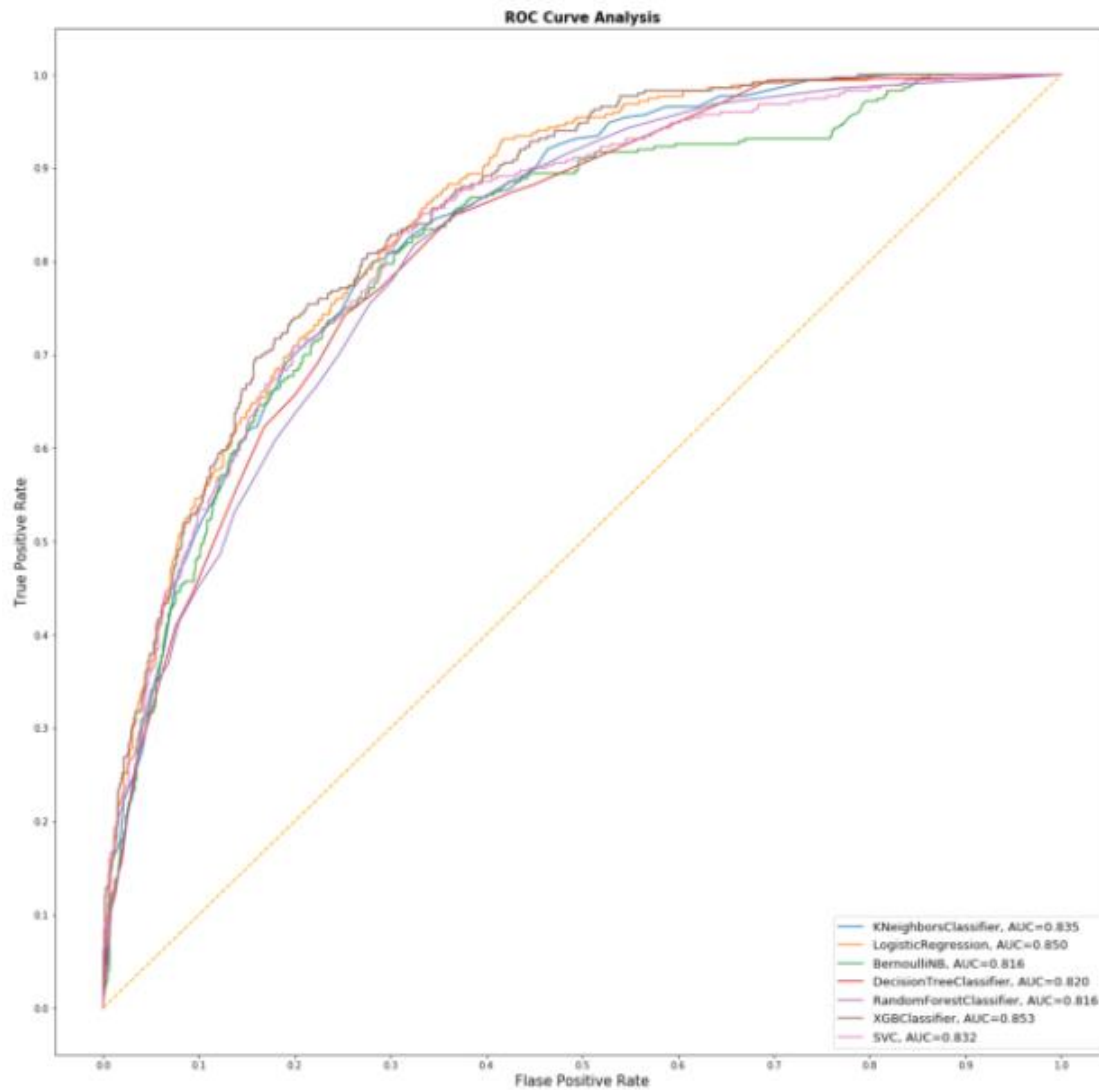


Figure 7: ROC curve Analysis

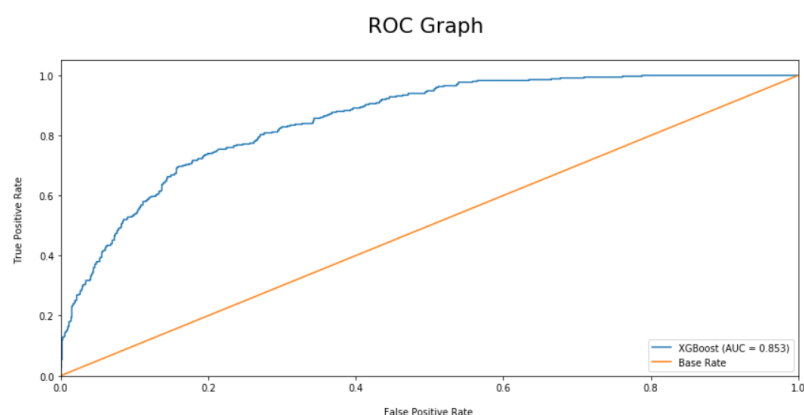
It is evident from the plot that the AUC for the XGBoost ROC curve is higher. Therefore, we can say that XGBoost did a better job of classifying the positive class in the dataset.

#### 4-3: Approved model

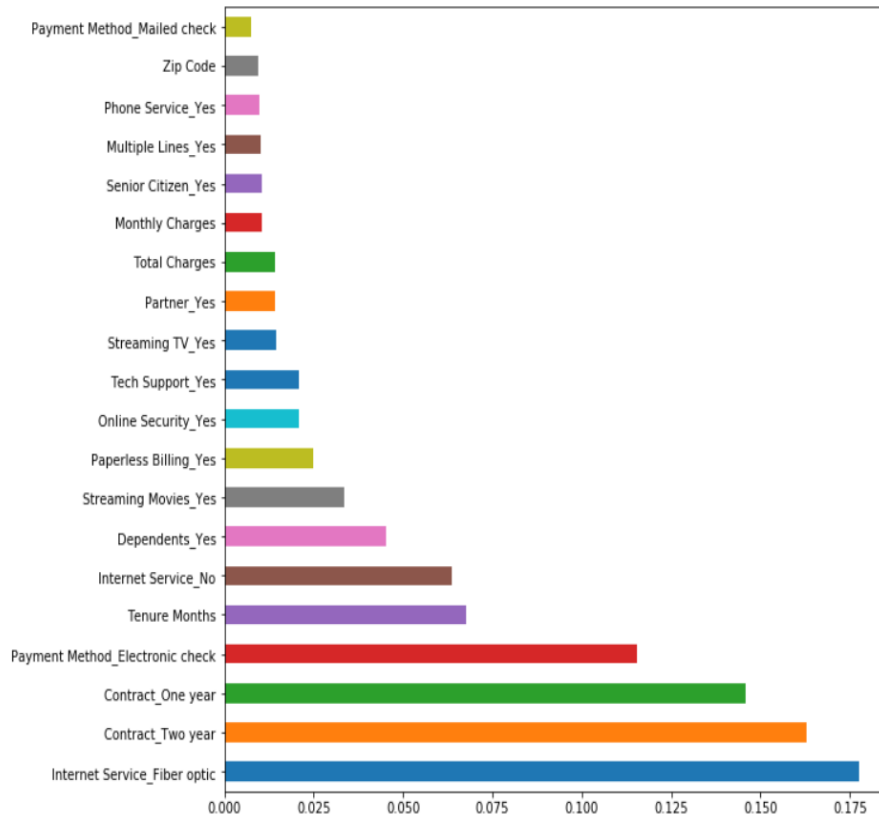
Model	Accuracy	Precision	Recall	F1 Score	F2 Score
XGBoost	0.810235	0.634304	0.560000	0.594841	0.573435
Logistic Regression	0.807392	0.624606	0.565714	0.593703	0.576587
SVC	0.805970	0.635088	0.517143	0.570079	0.537092
KNN	0.796020	0.589744	0.591429	0.590585	0.591091
Decision Tree	0.791756	0.612648	0.442857	0.514096	0.468845
Random Forest	0.780384	0.568562	0.485714	0.523883	0.500294
Naive Bayes	0.732054	0.476844	0.794286	0.595927	0.700958



We finally decided to choose our XGBoost model because it has shown a relatively high F1 score, accuracy, and AUC score compared to the other models.



We then decided to plot the feature importance graph.



## Conclusion

We have successfully generated models capable of giving the services we have promised. We have learned among other things the usage of sampling, parameter tuning, and the implementation and assessment of several models to find out the most efficient. We should now move on to generating a more user-friendly interface for our models.

Algorithm	Training	Testing
<b>Propose KNN</b>	80.45%	<b>97.78%</b>
<b>Random Forest</b>	76.36%	76.85%
<b>SVM</b>	<b>83.67%</b>	79.41%

Model	Accuracy	Precision	Recall	F1 Score	F2 Score
XGBoost	0.810235	0.634304	0.560000	0.594841	0.573435
Logistic Regression	0.807392	0.624606	0.565714	0.593703	0.576587
SVC	0.805970	0.635088	0.517143	0.570079	0.537092
KNN	0.796020	0.589744	0.591429	0.590585	0.591091
Decision Tree	0.791756	0.612648	0.442857	0.514096	0.468845
Random Forest	0.780384	0.568562	0.485714	0.523883	0.500294
Naive Bayes	0.732054	0.476844	0.794286	0.595927	0.700958



# Chapter V: Deployment

## 1-Introduction

In this step, we are going to deploy our models within web-based applications in order to provide the services we have promised to give and make it easier for our users to actually use the app. first we saved our model and scaler using the pickle module.

### Saving the model and scaler for deployment

```
dump(knn, open('model.pkl', 'wb'))  
dump(scaler, open('scaler.pkl', 'wb'))
```

## 2 -Deployment environment


### 2-1 DJANGO Framework

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design, due to the work done with python programming language and the use of multiple python machine learning libraries such as keras, tensorflow, sklearn and many other for scalability and rapid integration the choice of Django as the web framework was systematic

### 2-2 HEROKU

Heroku is a cloud platform as a service. it's helps Host Web Site developed with Django framework Also they provide the first 750 computation hours free of charge which means you can have one process (aka Dyno) at no cost. Also performance is very good e.g. simple web applications can handle around 60 - 70 requests per second.

## 3 -Web Application:



## Customer Churn Prediction

1

2

3

### Basic Information

📍 Zip Code :

👤 Gender :

☒ Male ☐ Female

👴 Senior Citizen :

☐ Yes ☒ No


👫 Partner :

☐ Yes ☒ No

👶 Dependents :


☐ Yes ☒ No

Next



## Customer Churn Prediction

You Customer has a 55.29% chance of churning !



Home

Project Repo