Project by: Debanti Roy

Customer Churn Analysis

Problem Statement

Customer churn is when a company's customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals.

We will examine customer data from IBM Sample Data Sets with the aim of building and comparing several customer churn prediction models and also find out which factors are important for a company to focus on to retain customers.

Problem Definition

In today's world the whole equation of conducting a business has changed. As per Online Business vs Offline Business Statistics, 82% of consumers use online media every day compared to offline media (24%). Here we understand that business can see more profitability in an online platform when compared to their physical counterparts. Now this calls for a competition to keep the audiences entertained or rather engaged and convert the leads into profitable ROI. Customer retention is one of the main relationship marketing objectives that can help minimise the bounce rate and ensure companies stay in the market for a long time.

Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can prioritise focused marketing efforts on that subset of their customer base.

Preventing customer churn is critically important to the telecommunications sector, as the barriers to entry for switching services are so low.

Currently, the perception and application of customer retention is significantly valuable for companies. Thus, in order to perceive and apply marketing principles in practice relevantly, it is important to ground theoretically and assess empirically customer churn rate. Let's take into account customer data from IBM Sample Data Sets to understand how several factors can influence customer churning.

- **Understanding Customer Churn**

Customer churn is the percentage of customers that have stopped using the company's products or services. Simply put, these are the customers who refuse to continue paying for a service or fails to become repeat customers. It is also known as customer attrition.

- **One Reason for Customer Churn**

When we are talking about customer churn, the most important reason that comes to my mind is the price of a product or service the customer is paying in exchange. If they do not feel the quality of the product or service is not worth the price they would seek the same from a competitor site or company.

- **How to Calculate Churn Rate?**

One can calculate churn rate by dividing the number of customers lost during a time period -- say a quarter -- by the number of customers in the beginning of that time period.

Let us take into account the case of IBM Sample Data Sets and apply Machine Learning techniques to find out what are the factors that led to customer churn.

Data Analysis

- **Importing the necessary libraries**

In order to start with the analysis I have first loaded the necessary packages and libraries.

In [1]:
```
#Importing required packages & libraries.

import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Here pandas and numpy is used for data manipulation, matplotlip and seaborn is used to plot any data in a graph format. We have also used the function 'import warnings' so that our program can ignore warnings in case there is one found.

- **Loading dataset**

Next, we have used the '.read_csv()' method to import the sample dataset and '.head()' method to pint the head of the data.

```
In [2]: #Loading the dataset
        df=pd.read_csv("https://raw.githubusercontent.com/dsrscientist/DSData/master/Telecom_customer_churn.csv")

        #printing the head of the dataset
        df.head()
```

- **Checking shape**

Then we have used '.shape' method to check the shape of the dataset.

```
In [3]: #checking the shape of data
        df.shape

Out[3]: (7043, 21)
```

Here we can see the dataset has 7043 rows and 21 columns.

- **Checking data types**

```
In [4]: #checking the type of data
        df.dtypes

Out[4]: customerID          object
        gender              object
        SeniorCitizen        int64
        Partner             object
        Dependents          object
        tenure               int64
        PhoneService        object
        MultipleLines       object
        InternetService     object
        OnlineSecurity      object
        OnlineBackup        object
        DeviceProtection    object
        TechSupport         object
        StreamingTV         object
        StreamingMovies     object
        Contract            object
        PaperlessBilling    object
        PaymentMethod       object
        MonthlyCharges     float64
        TotalCharges        object
        Churn               object
        dtype: object
```

After we are done checking the shape of the dataset, it is time for us to have an indepth understanding of the data types. Using the '.dtypes' method we can find out the data type in each column. Here we can see that majority of columns (18 out of 21) has object type data. However, for our analysis we require numeric data so we will have to convert them to numeric values going forward.

- **Checking null values**

```
In [5]: #checking null values in the dataset
        df.isnull().sum()

Out[5]: 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        0
        Churn               0
        dtype: int64
```

The next and most important part of data analysis is finding out null values. This step is very important because having null values in our dataset means our analysis will deflect from the actual results. Thus, in order to make sure our analysis is correct we will have to ensure our data is fee from null values.

During my analysis I have checked the null values using the '.isnull().sum()' method. Having the '.sum()' method gives us a count of the total number of nulls present in each column. Here in the dataset I found that there were no null values present. Thus, it was a good sign for me to proceed forward.

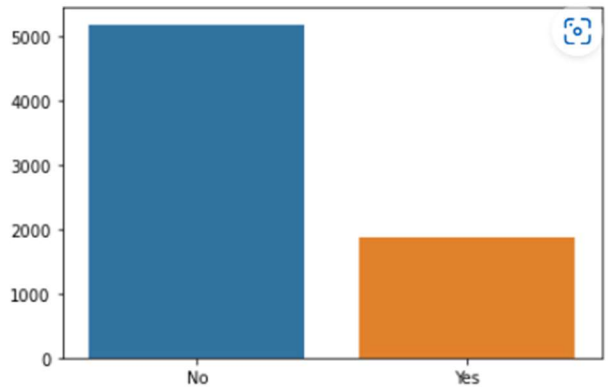
- **Data Visualization**

  Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

```
In [6]: y = df["Churn"].value_counts()

        sns.barplot(y.index, y.values)

Out[6]: <AxesSubplot:>
```



Here, I took the target variable that is 'Churn' to find out the distribution of data.

As per the plot we can see around 2000 customers out of the data of 7043 had actually switched to a competitor business. Though the number seems almost half from the customers which the company retained, it is essential to find out what factors may have influenced customer churn and which are the features IBM here should focus on to make sure they retain more customers.

- **Statistical Analysis**

```
In [8]: df.describe() #checking statistical information
```

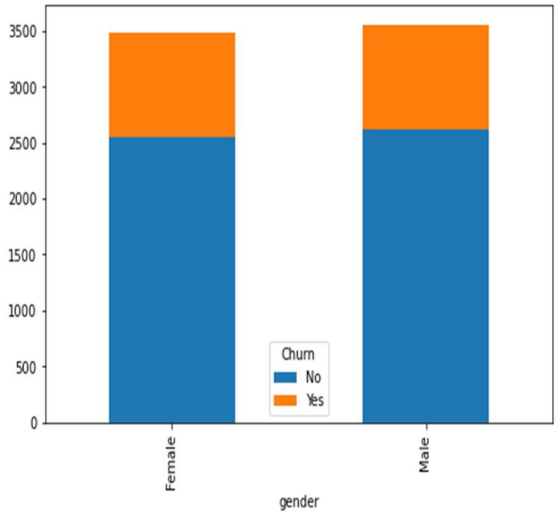
Out[8]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

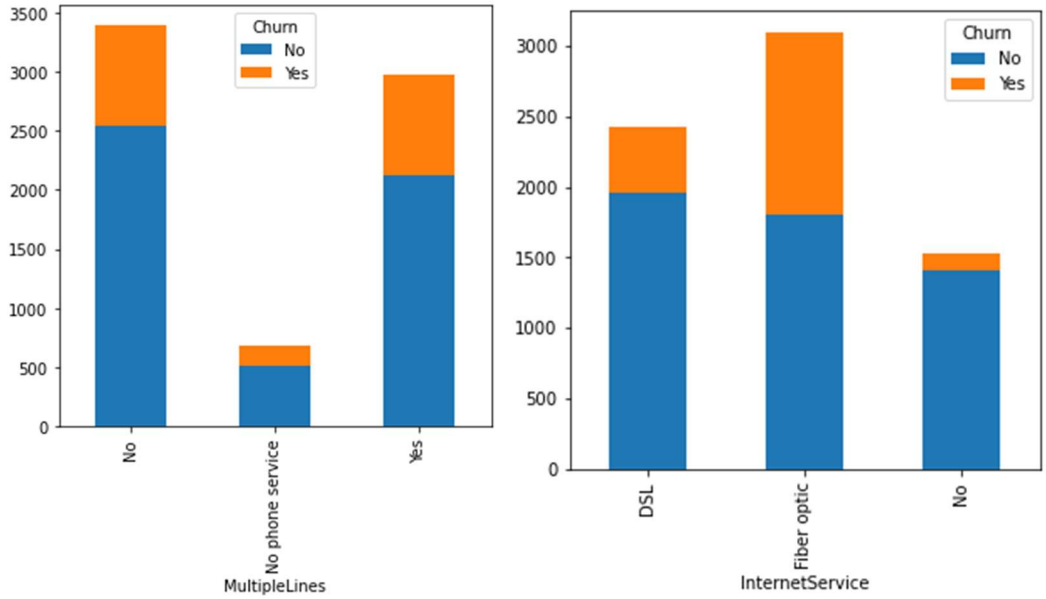
We have also performed a statistical analysis of the data using the '.describe()' method where the columns with numerical values are displayed giving us an idea of count, mean, standard deviation, min, max, 25%, 50% and 75%.

As per my understanding the most relevant factor here is Monthly Charges which has a mean of 64.76, standard deviation of 30.09, min=18.25, max=118.75, 25% percentile of 35.50, 50% percentile of 70.35 and 75% percentile of 89.85.

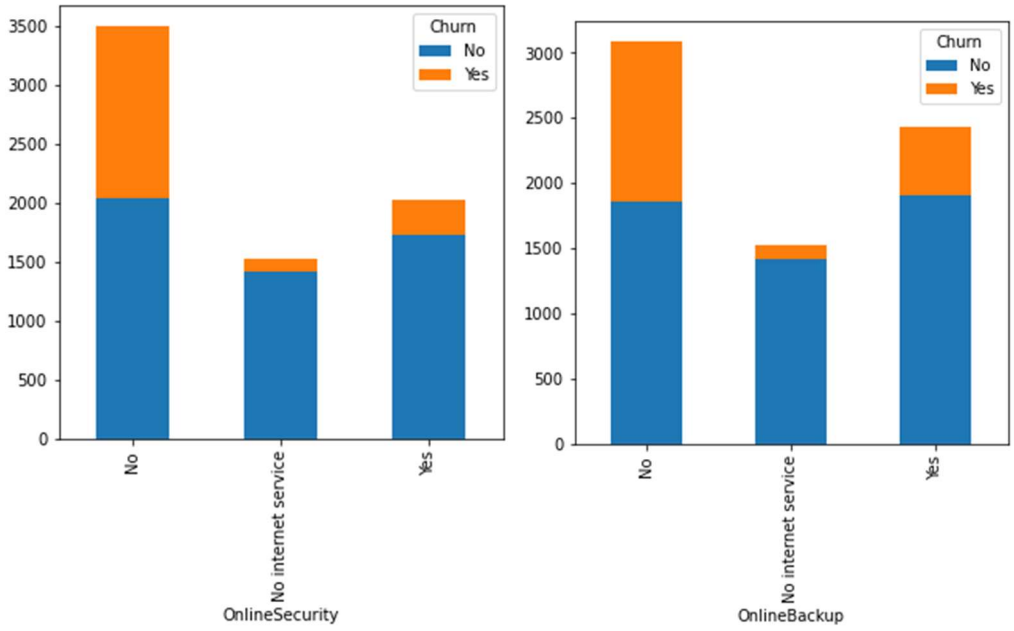
To visualize the data I have plotted the following plots that helped me understand the relation between Churn and other columns. Here are some of the plots:



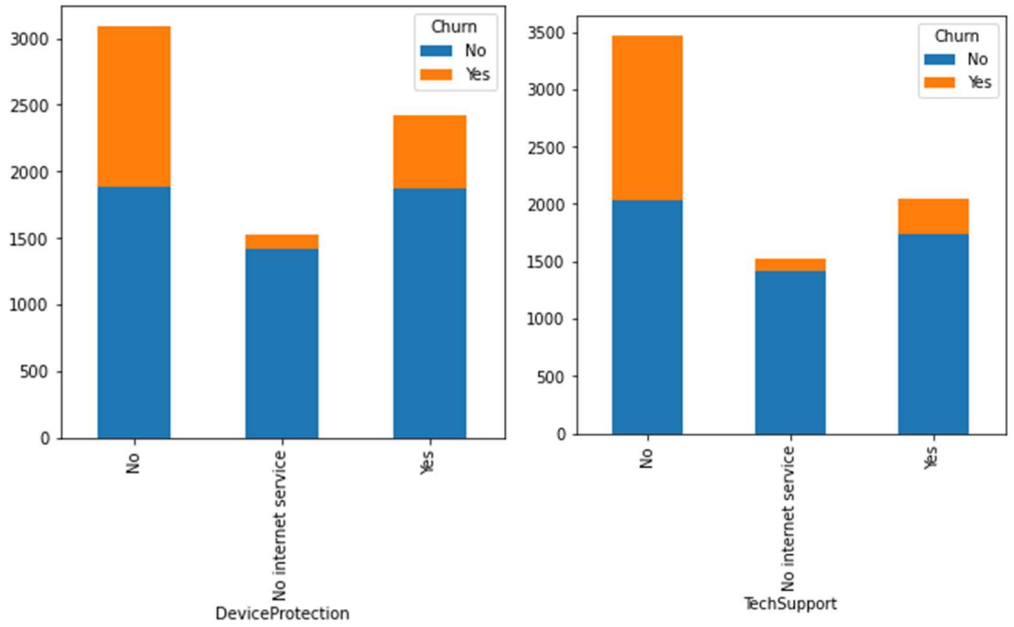
Observation: We can see that the distribution rate in terms of male and female customers whom the company retained and others who left is almost the same.



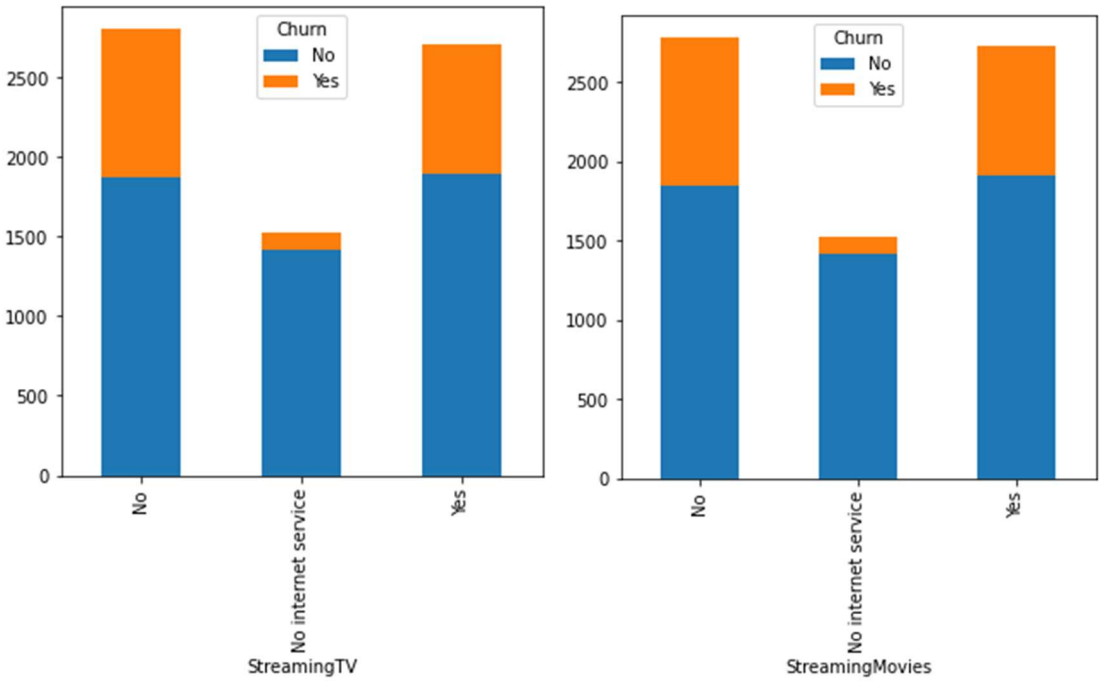
Observation: We can see customers who did not have multiple lines were the most to stay in comparison to customers who had multiple lines and without any phone service. In the second plot we can see rate of churn was higher in customers who fiber optic internet service than DSL or none.



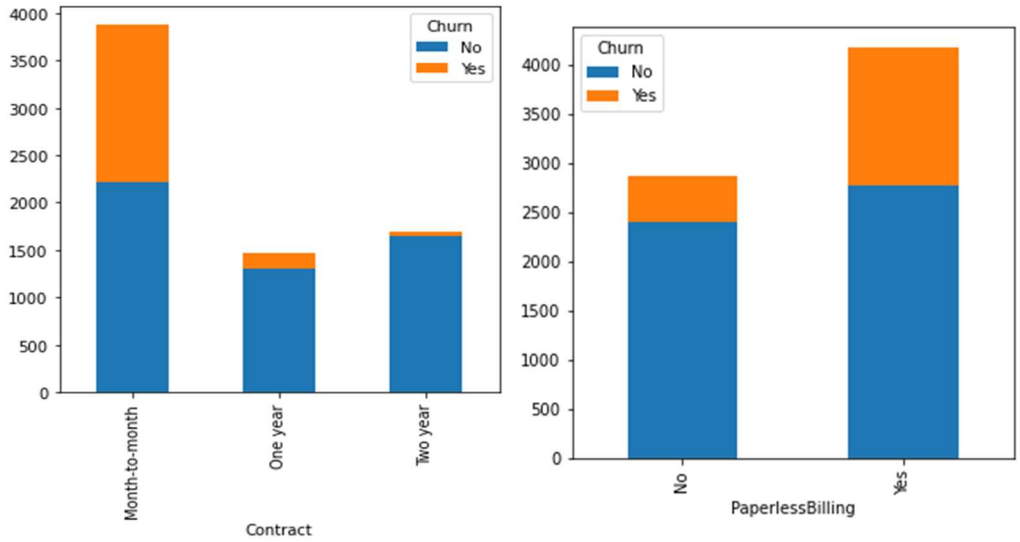
Observation: In the first plot we can see that customers preferred online security. However, there were also customers who did not care about online security while they left. In the second plot, we find that customers who left did not have online backup left more than the ones who had backup and no internet service.



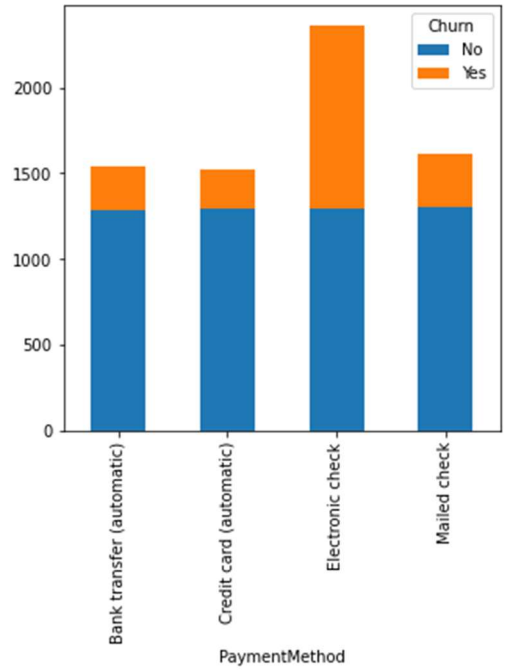
Observation: In the first plot we find that churn was more in customers where they had less device protection than the others with protection and no internet service. The second plot shows that churn was more in customers where they had less device protection than the others with protection and no internet service.



Observation: In the first plot we find that churn was almost equal for customers who had/had not access to Streaming TV than the ones with no internet. The second plot shows that churn was almost equal for customers who had/had not access to Streaming Movies than the ones with no internet.



Observation: We can see churn was more with people who had month-to-month contract than the customers who had subscribed for one or two years. Also, we notice that more people who got paperless billing switched to other companies than others.



Observation: In the above plot we can see that more people who preferred Electronic check as Payment Method switched more than other modes of payment like automatic bank transfer, credit card, mailed check.

- **Data Cleaning**

While going through the data thoroughly I found that there were certain cells with missing values in the Total Charges column. Hence, I have dropped the column using drop method.

```
In [22]: df.drop('TotalCharges', axis=1, inplace=True)
```

- **Data Transformation**

Now, that our data looked good we can apply here other preprocessing techniques. Here, it is important to note that most of the columns have two to three values or they rather are the options that the customers have chosen. Hence, we can easily endode them with numerical values like 0, 1, 2, etc. In order to do that we have used the method 'LabelEncoder()'. We have called the method using the code 'from sklearn import preprocessing'.

```
In [24]: # Discreet value integer encoder
         from sklearn import preprocessing

         label_encoder = preprocessing.LabelEncoder()
```

I have applied Label Encoder in all the object columns, viz., DeviceProtection, MultipleLines, TechSupport, PhoneService, InternetService, OnlineSecurity, Streaming

TV, StreamingMovies, Contract, PaperlessBilling, Partner, Dependents, gender, OnlineBackup and Churn.

In [25]:
```
# State is string and we want discreet integer values
df['DeviceProtection'] = label_encoder.fit_transform(df['DeviceProtection'])
df['MultipleLines'] = label_encoder.fit_transform(df['MultipleLines'])
df['TechSupport'] = label_encoder.fit_transform(df['TechSupport'])
df['PhoneService'] = label_encoder.fit_transform(df['PhoneService'])
df['InternetService'] = label_encoder.fit_transform(df['InternetService'])
df['OnlineSecurity'] = label_encoder.fit_transform(df['OnlineSecurity'])
df['StreamingTV'] = label_encoder.fit_transform(df['StreamingTV'])
df['StreamingMovies'] = label_encoder.fit_transform(df['StreamingMovies'])
df['Contract'] = label_encoder.fit_transform(df['Contract'])
df['PaperlessBilling'] = label_encoder.fit_transform(df['PaperlessBilling'])
df['PaymentMethod'] = label_encoder.fit_transform(df['PaymentMethod'])
df['Partner'] = label_encoder.fit_transform(df['Partner'])
df['Dependents'] = label_encoder.fit_transform(df['Dependents'])
df['gender'] = label_encoder.fit_transform(df['gender'])
df['OnlineBackup'] = label_encoder.fit_transform(df['OnlineBackup'])
df['Churn'] = label_encoder.fit_transform(df['Churn'])
```

```
In [26]: #checking the datatype after applying label encoder
         print (df.dtypes)

customerID          object
gender               int32
SeniorCitizen        int64
Partner              int32
Dependents           int32
tenure               int64
PhoneService         int32
MultipleLines        int32
InternetService      int32
OnlineSecurity       int32
OnlineBackup         int32
DeviceProtection     int32
TechSupport          int64
StreamingTV          int32
StreamingMovies      int32
Contract             int32
PaperlessBilling     int32
PaymentMethod        int32
MonthlyCharges     float64
Churn                int32
dtype: object
```

To ensure our data is transformed I have checked the datatypes again using the '.dtypes' method. And here we can clearly see that all the columns have been transformed into integer from object type.

- **Making Data Ready for Model Building**

Next, it was time to split the data and making it ready for training and testing.

```
In [28]: #checking the shape of dataset
         df_new.shape

Out[28]: (7043, 20)

In [29]: #saving our target column in a y to make our analysis easier.
         y = df_new['Churn']
         y.size

Out[29]: 7043

In [30]: # now dropping customerID and Churn column
         df_new.drop(["customerID","Churn"], axis = 1, inplace=True)
```

We will save the rest of the data from df_new dataframe in x

```
In [31]: X = df_new
```

Here, I have saved the target column, i.e., 'Churn' in y and the independent columns in X. Additionally, I have also dropped the 'customerID' and 'Churn' column for saving in X.

Note that here "customerID' has no relevance in regards to our prediction so we could have dropped it earlier as well.

- **Data Normalization**

I have used StandardScaler to normalise the data. NormaliZation is the process of transforming the columns in a dataset to the same scale.

```
In [33]: #Using StandardScalar to normalise the dataset

         from sklearn.preprocessing import StandardScaler

         scaler = StandardScaler()
```

- **Building Machine Learning Models**

Moving forward, we have arrived to the main part of our analysis, i.e., building our Machine Learning Models. For this I have evaluated that data using Logistic Regression, MLP Classifier, SVC, Decision Tree Classifier, Ada Boost Classifier, bagging Classifier, Gradient Boosting Classifier, Random Forest Classifier.

Also to split that data into training and testing I have used train, test, split.

```
from sklearn import linear_model

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier, GradientBoostingClassifier, RandomForestClassifier


from sklearn.model_selection import train_test_split
```

The results of the testing were as follows:

```
LogisticRegression: 0.8131
SVC: 0.7998
MLPClassifier: 0.8064
DecisionTreeClassifier: 0.7449
AdaBoostClassifier: 0.8107
BaggingClassifier: 0.7908
GradientBoostingClassifier: 0.8088
RandomForestClassifier: 0.7974
```

- **Finding The Best Parameter**

I have used 'GridSearchCV' to find out the best parameters.

```
from sklearn.model_selection import GridSearchCV

params = {'n_estimators': list(range(50, 200, 25)), 'max_features': ['auto', 'sqrt', 'log2'],
        'min_samples_leaf': list(range(50, 200, 50))}

grid_search_cv = GridSearchCV(RandomForestClassifier(random_state=42), params, n_jobs=-1)
grid_search_cv.fit(X_train, y_train)
```

- **Saving the Model**

I found that Logistic Regression was showing the highest accuracy in predicting data, thus, I have saved the model using 'import joblib'.

```
In [43]: #selecting the best model
         lr = LogisticRegression()
         lr.fit(X, y)

Out[43]: LogisticRegression()

In [44]: from joblib import Parallel, delayed
         import joblib

         #Saving the best model
         joblib.dump(lr, 'Customer_Churn_Analysis.pkl')

Out[44]: ['Customer_Churn_Analysis.pkl']
```

I also called the model to find out the prediction of the analysis.

```
# Load the model from the file
lr_from_joblib = joblib.load('Customer_Churn_Analysis.pkl')

# Use the loaded model to make predictions
lr_from_joblib.predict(X_test)
```

```
array([0, 1, 1, ..., 0, 0, 1])
```

- **Predicting Most Useful Features**

In my analysis I have also tried to find out which feature variables are the most important in order to predict Customer Churn. I applied 'feature_selection' to find the same.

```
from sklearn.feature_selection import mutual_info_classif
```

As per the analysis, the following table shows us the relevance of each variable in descending order, starting with the most relevant variable from the top.

	importance
Contract	0.103234
tenure	0.079835
OnlineSecurity	0.061837
TechSupport	0.059470
InternetService	0.053062
PaymentMethod	0.050888
DeviceProtection	0.048891
OnlineBackup	0.045879
StreamingTV	0.041966
MonthlyCharges	0.040933
StreamingMovies	0.030915
PaperlessBilling	0.022002
Partner	0.012586
Dependents	0.011120
gender	0.008226
SeniorCitizen	0.008116
MultipleLines	0.001054
PhoneService	0.000000

Concluding Remarks

It has been found that online businesses run the highest risk of customer churn because of competitive prices, interesting and engaging WebApp features, relevant content, etc. Contrarily, businesses should try to keep their churn rate as low as possible because more churn is directly proportional to higher revenue lost. Additionally, the costs to replace old customers with new ones using marketing strategies is much higher than retaining them.

From the above data we understand that contract, online security, tenure, tech support, online backup are the top five important features that can impact customer churn than any other. Thus, IBM or for that matter any company having business online needs to put on their efforts on improving these factors so that they can retain more customers in the long run.