Customer Churn Prediction

[Predict whether a customer will change telco Provider]

Introduction

In this blog-post, I will go through the whole process of creating a machine learning model on the **IBM Sample customer churn Dataset** which is used by many people all over the world.

Customer churn

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.

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 priorities focused marketing effort on that subset of their customer base.

Problem Definition

Based on the introduction the key challenge is to predict if an individual customer will churn or not. To accomplish that, machine learning models are trained based on 80% of the sample data. The remaining 20% are used to apply the trained models and assess their predictive power with regards to “churn / not churn”. A side question will be, which features actually drive customer churn. That information can be used to identify customer “pain points” and resolve them by providing goodies to make customers stay.

Data Collection

The data set for this classification problem is taken from Kaggle and stems from the IBM sample data set collection

https://www.kaggle.com/becksddf/churn-in-telecoms-dataset#bigml_59c28831336c6604c800002a.csv

Importing the Libraries

```
#importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")


```

Loading the Data

For this exercise, the data set (.csv format) is downloaded to a local folder, read into the Jupiter notebook and stored in a Pandas Data Frame.

```
df=pd.read_csv("avacado.csv")
df
```

Data Analysis

After data collection, several steps are carried out to explore the data. Goal of this step is to get an understanding of the data structure, conduct initial preprocessing, clean the data, identify patterns and inconsistencies in the data (i.e., skewness, outliers, missing values) and build and validate hypotheses.

Here I am going to evaluate the structure, columns included and data types. The goals of this step are to get a general understanding for the data set, check domain knowledge and get first ideas on topics to investigate.

```
1  df.info()
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
```

Number of Rows and Columns

```
1  df.shape
```

```
(7043, 21)
```

Column Names and Details

```
df.columns
```

```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

Customer ID – ID of the customers

Gender- Whether the customer is a male or a female

Senior Citizen - Whether the customer is a senior citizen or not (1, 0)

Partner - Whether the customer has a partner or not (Yes, No)

Dependents - Whether the customer has dependents or not (Yes, No)

Tenure- Number of months the customer has stayed with the company

Phone Service- Whether the customer has a phone service or not (Yes, No)

Multiple Lines - Whether the customer has multiple lines or not (Yes, No, No phone service)

Internet Service - Customer’s internet service provider (DSL, Fiber optic, No)

OnlineSecurity - Whether the customer has online security or not (Yes, No, No internet service)

OnlineBackup - Whether the customer has online backup or not (Yes, No, No internet service)

DeviceProtection - Whether the customer has device protection or not (Yes, No, No internet service)

TechProtection - Whether the customer has tech support or not (Yes, No, No internet service)

StreamingTV - Whether the customer has streaming TV or not (Yes, No, No internet service)

StreamingMovies - Whether the customer has streaming movies or not (Yes, No, No internet service)

Contract - The contract term of the customer (Month-to-month, One year, Two year)

PaperlessBilling - Whether the customer has paperless billing or not (Yes, No)

PaymentMethod - The customer’s payment method (Electronic check, mailed check, Bank transfer (automatic), Credit card (automatic))

MonthlyCharges - The amount charged to the customer monthly

TotalCharges - The total amount charged to the customer

Churn - Whether the customer churned or not (Yes or No)

The unique values for every feature are printed to the console to get a deeper understanding about the feature values.

```
for i in df.columns:
    print("unique value of",i,"is",df[i].unique(),"\n")
```

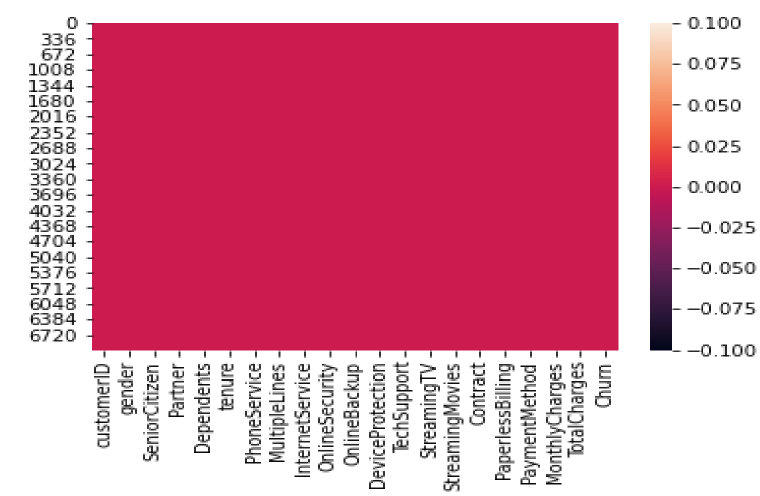
```
unique value of customerID is ['7590-VHVEG' '5575-GNVDE' '3668-QPYBK' ... '4801-
JZAZL' '8361-LTMKD' '3186-AJIEK']

unique value of gender is ['Female' 'Male']

unique value of SeniorCitizen is [0 1]

unique value of Partner is ['Yes' 'No']

unique value of Dependents is ['No' 'Yes']

unique value of tenure is [ 1 34 2 45 8 22 10 28 62 13 16 58 49 25 69 52 71 21 12
30 47 72 17 27  5 46 11 70 63 43 15 60 18 66 9 3 31 50 64 56 7 42 35 48 29 65 38
68 32 55 37 36 41 6 4 33 67 23 57 61 14 20 53 40 59 24 44 19 54 51 26 0 39]

unique value of PhoneService is ['No' 'Yes']

unique value of MultipleLines is ['No phone service' 'No' 'Yes']

unique value of InternetService is ['DSL' 'Fiber optic' 'No']

unique value of OnlineSecurity is ['No' 'Yes' 'No internet service']

unique value of OnlineBackup is ['Yes' 'No' 'No internet service']

unique value of DeviceProtection is ['No' 'Yes' 'No internet service']

unique value of TechSupport is ['No' 'Yes' 'No internet service']

unique value of StreamingTV is ['No' 'Yes' 'No internet service']

unique value of StreamingMovies is ['No' 'Yes' 'No internet service']

unique value of Contract is ['Month-to-month' 'One year' 'Two year']

unique value of PaperlessBilling is ['Yes' 'No']

unique value of PaymentMethod is ['Electronic check' 'Mailed check' 'Bank
transfer (automatic)' 'Credit card (automatic)']

unique value of MonthlyCharges is [29.85 56.95 53.85 ... 63.1 44.2 78.7]

unique value of TotalCharges is ['29.85' '1889.5' '108.15' ... '346.45' '306.6'
'6844.5']

unique value of Churn is ['No' 'Yes']
```

Let's take a more detailed look at the Dataset

Before we can feed our data set into a machine learning algorithm, we have to remove missing values and split it into training and test sets.

Finding Missing Values

```
sns.heatmap(df.isnull())
```



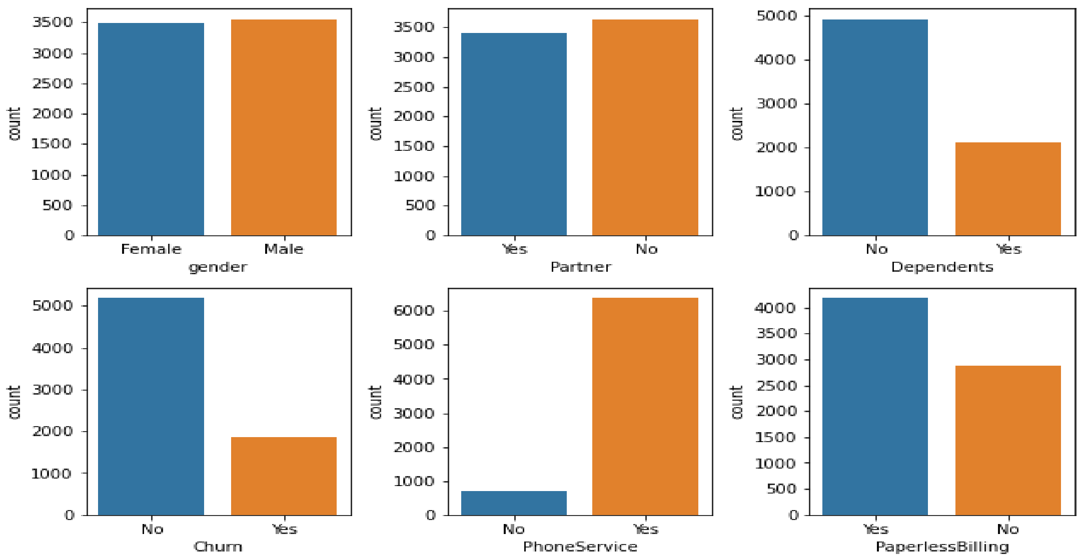
From the visualization heatmap, we find no Missing Values.so we proceed further

What features could contribute to find whether customer get churn or not?

To me it would make sense if everything except column "CustomerID" in the dataset would be correlated with whether a customer gets churn or not.

Data Exploration

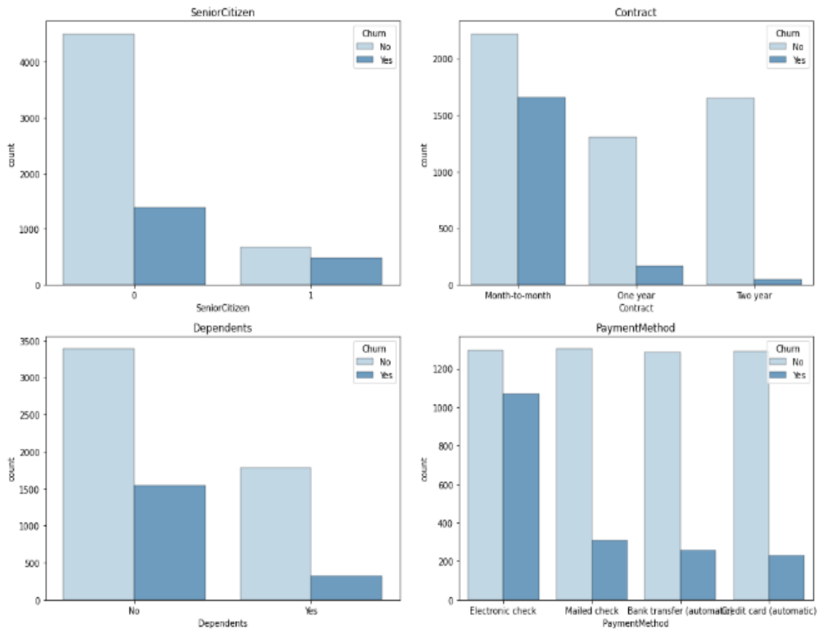
Curating datasets and turning them into visualizations is fascinating to look at, and it so happens to be what we do best. From dataset we gathered information on customer churn attributes including gender, Partner, dependents, churn, Phone service, Paperless Billing and we created visualizations for each of these categories, exposing patterns and outliers from the dataset.



Plot insights:

- Number of male customers are more than the female customers.
- There are less customers who have a partner than the ones who don't partnered with anyone.
- The customers who have dependents are less when compared with the non-dependents.
- The customers who chummed from the telecom service is less than the ones who choose to stay with the service
- There are lot of customers who have a Phone service
- There are a greater number of customers who choose Paperless billing

Next, we will proceed with the visualization on attributes which are corelated to our target churn.



Plot insights:

- Senior citizens churn rate is much higher than non-senior churn rate.
- Churn rate for month-to-month contracts much higher than for other contract durations.
- Moderately higher churn rate for customers without partners.
- Much higher churn rate for customers without children.
- Payment method electronic check shows much higher churn rate than other payment methods.
- Customers with Internet Service fiber optic as part of their contract have much higher churn rate.

Label encoding

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

The following features are categorical, yet not ordinal (no ranking) but take one or more than 2 values. For each value, a new variable is created with a binary integer indicating if the value occurred in a data entry or not (1 or 0).

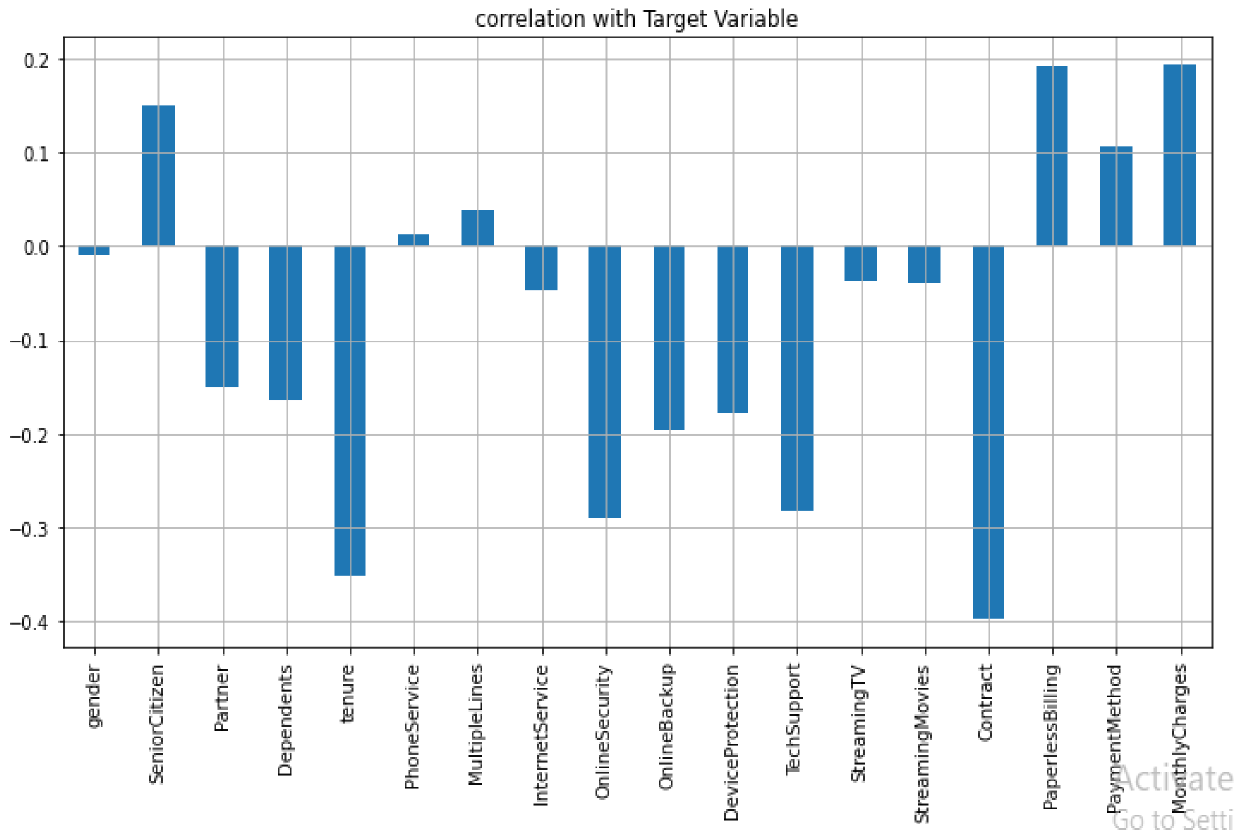
- gender
- Partner
- Dependents
- Churn
- PhoneService
- PaperlessBilling
- MultipleLines
- InternetService
- OnlineSecurity
- OnlineBackup
- DeviceProtection
- TechSupport
- StreamingTV
- StreamingMovies
- Contract
- PaymentMethod

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
values=["gender","Partner","Dependents","Churn","PhoneService","PaperlessBilling","MultipleLines",
        "InternetService","OnlineSecurity",
       "DeviceProtection","TechSupport","StreamingTV","Contract","PaymentMethod","OnlineBackup","StreamingMovies"]
for i in values:
    df[i]=le.fit_transform(df[i])

```

CORRELATION

Correlation analysis is a widely used statistical measure through which different studies have efficiently identified interesting collinear relations among different attributes of datasets. Correlation analysis is an extensively used technique that identifies interesting relationships in data. These relationships help us realize the relevance of attributes with respect to the target class to be predicted.

```
plt.figure(figsize=(10,10))
sns.heatmap(cor,annot=True)
plt.show()
```



from the above result it is clear that some columns making positive correlation while some has negative correlation to the target variable

columns making positive correlation were “SeniorCitizen”, “phoneService”, “MultipleLines”, “PaperlessBilling”, “PaymentMethod”, “MonthlyCharges”.

columns making negative correlation were "Gender", "Partner", "Dependents", "InternetService", "OnlineSecurity", "OnlineBackup", "DeviceProtection", "Techsupport", "streamingTV", "StreamingMovies", "Contarct".

Here we see lot of our attributes have negatively correlated with our target attribute. If we drop more variables than necessary, less information will be available. From our observation we can see all the variables are needed. So, we decide not to drop any variables

Data Preprocessing

First, I will drop “CustomerID”,” from the Dataset, because it does not contribute to our prediction whether the customer gets churn or not.

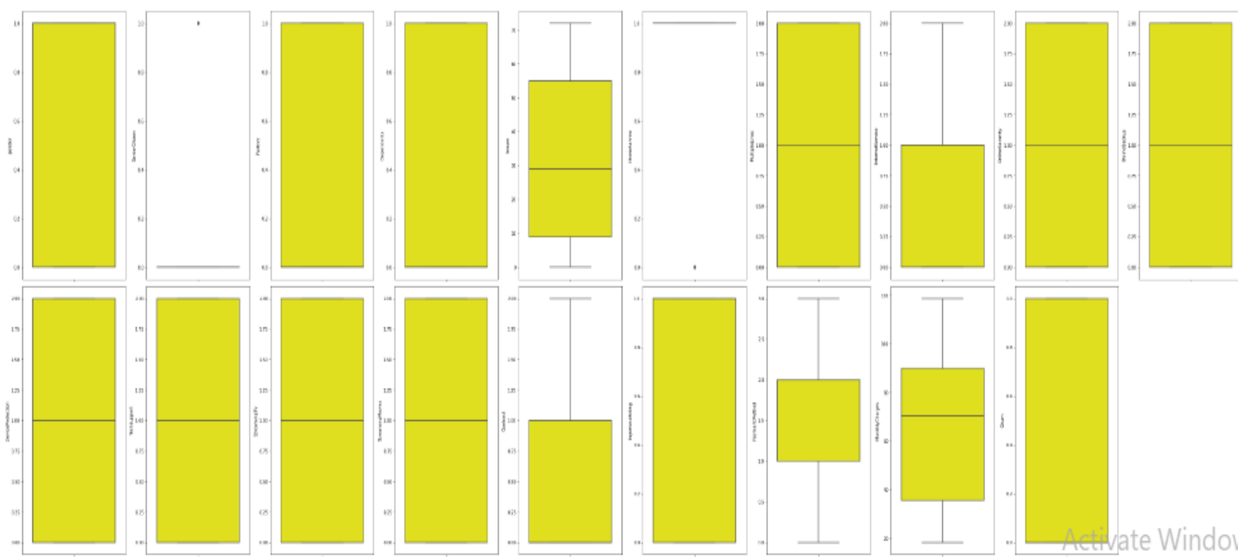
Once we drop the column which is no needed, we will proceed with Data cleaning

```
df.drop(columns=["customerID"],inplace=True)
```

Data cleaning

The difference between a good and an average machine learning model is often its ability to clean data. One of the biggest challenges in data cleaning is the identification and treatment of outliers. In simple terms, outliers are observations that are significantly different from other data points. Even the best machine learning algorithms will underperform if outliers are not cleaned from the data because outliers can adversely affect the training process of a machine learning algorithm, resulting in a loss of accuracy.

1) Outliers



There can be many reasons for the presence of outliers in our data. Sometimes the outliers may be genuine, while in other cases, they could exist because of data entry errors. It is important to understand the reasons for the outliers before cleaning them.

We will start the process of finding outliers by running the summary statistics on the variables. This is done using the *describe ()* function below, which provides a statistical summary of all the quantitative variables.

```
df.describe()
```

OnlineBackup	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling	PaymentMethod	MonthlyCharges	Churn
7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000
0.906432	0.904444	0.797104	0.985376	0.992475	0.690473	0.592219	1.574329	64.761692	0.265370
0.880162	0.879949	0.861551	0.885002	0.885091	0.833755	0.491457	1.068104	30.090047	0.441561
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	18.250000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	35.500000	0.000000
1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	2.000000	70.350000	0.000000
2.000000	2.000000	2.000000	2.000000	2.000000	1.000000	1.000000	2.000000	89.850000	1.000000
2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	1.000000	3.000000	118.750000	1.000000

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity
count	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.504756	0.162147	0.483033	0.299588	32.371149	0.903166	0.940508	0.872923	0.790004
std	0.500013	0.368612	0.499748	0.458110	24.559481	0.295752	0.948554	0.737796	0.859848
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000	9.000000	1.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	0.000000	0.000000	29.000000	1.000000	1.000000	1.000000	1.000000
75%	1.000000	0.000000	1.000000	1.000000	55.000000	1.000000	2.000000	1.000000	2.000000
max	1.000000	1.000000	1.000000	1.000000	72.000000	1.000000	2.000000	2.000000	2.000000

There are no missing values. Here we find that the median is higher than mean in gender,PhoneService, MultipleLines,InternetService, OnlineSecurity,OnlineBackup, PaymentMethod, MonthlyCharges.**If the mean is less than the median, the distribution is negatively skewed.** The maximum and the 75% had a very small range of difference. we infer that **we may have very few outliers and skewness in some of the attributes.** These outliers were easy to detect, but that will not always be the case.

Removing Outliers

Z score for Outlier treatment:

Z score is an important concept in statistics. Z score is also called standard score. This score helps to understand if a data value is greater or smaller than mean and how far away it is from the mean. More specifically, Z score tells how many standard deviations away a data point is from the mean. If the z score of a data point is more than 3, it indicates that the data point is quite different from the other data points. Such a data point can be an outlier.

```
#using z-score technique
#Removing Outliers
#Z-score Techinique
from scipy.stats import zscore
z=np.abs(zscore(df))
df_new=df[(z<3).all(axis=1)]
```

Skewness

Skewness refers to a distortion or asymmetry that deviates from the symmetrical bell curve, or normal distribution, in a set of data. ... A normal distribution has a skew of zero, while a lognormal distribution, for example, would exhibit some degree of right-skew.

Checking for Skewness

```
#checking for skewness
df_new.skew()
```

```
gender             -0.014781
SeniorCitizen       1.823376
Partner             0.056316
Dependents          0.876594
tenure              0.237945
PhoneService        0.000000
MultipleLines       0.132058
InternetService     0.049126
OnlineSecurity      0.422032
OnlineBackup        0.167910
DeviceProtection    0.183254
TechSupport         0.409833
StreamingTV        -0.002734
StreamingMovies    -0.010025
Contract            0.629701
PaperlessBilling   -0.386613
PaymentMethod      -0.169889
MonthlyCharges     -0.399139
Churn               1.053055
dtype: float64
```

Normally the threshold value of the skewness is +/-0.55

Here we can see skewness in columns like “Senior Citizen”,” Dependents”,” Contract". We will remove the skewness once we separate the target value. The target attribute should not be touched because skewing it may result in change of values since our target variable is of a classifier type.

Separating Target Variable

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

```
x_l=df_new.drop(["Churn"],axis=1)
y=df_new["Churn"]
```

Removing Skewness

```
#Skewness treatment
#treating using log
threshold=0.55
import numpy as np
for i in x_l.columns:
    if x_l[i].skew()>0.65:
        x_l[i]=np.log1p(x_l[i])

```

Scaling Input

Many machine learning algorithms perform better when numerical input variables are scaled to a standard range. This includes algorithms that use a weighted sum of the input, like logical regression, and algorithms that use distance measures, like k-nearest neighbors.

```
#using StandardScaler techinique
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_l=sc.fit_transform(x_line)
x_l
```

Once scaling is done, we are ready for the fun part

Train-Test-Split

For conduction of model training and testing steps, the data set is split into 80% training data and 20% test data. The “Churn” column is defined as the class (the “y”), the remaining columns as the features (the “X”).

```
#importing Libraries
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
#Train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.20,random_state=42)
```

Building Machine Learning Models

Now we will train several Machine Learning models and compare their results. Note that because the dataset does not provide labels for their testing-set, we need to use the predictions on the training set to compare the algorithms with each other. Later on, we will use cross validation.

For the **predictive models**, I used the following 7 models to train the datasets, test the predicted churn, and compared them against actual to score the accuracies.

Popular algorithms that can be used for classification Prediction include:

- k-Nearest Neighbors.
- Decision Trees.
- Support Vector Machine
- Random Forest Regressor
- Gradient Boosting
- Logistic Regression

```
#importing our model library
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
from sklearn.tree import DecisionTreeClassifier
dt=DecisionTreeClassifier()
from sklearn.svm import SVC
svc=SVC()
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier()
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier()
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()
```

```
model=[lr,knn,dt,svc,rf,gb]
for m in model:
    m.fit(x_train,y_train)
    pred_train=m.predict(x_train)
    pred_test=m.predict(x_test)
    print("the score of ",m,"is")
    print("training accuracy score  :",accuracy_score(y_train,pred_train)*100)
    print("testing accuracy score  :",accuracy_score(y_test,pred_test)*100)
    print("**********************************************************************")
    print("\n\n")
```

Score Board

score	Logistic Regression	k-Nearest Neighbors	Decision Trees	Support Vector Machine	Random Forest classifier	Gradient Boosting
Training accuracy score	80.11	83.33	99.68	82.09	99.68	82.54
Testing accuracy score	80.67	77.14	73.60	79.49	79.34	80.98

Based on the above chart, **Gradient Boosting model** did quite better than other models in which I combined all seven model's outputs and averaged them out. This looks much more realistic than before. Our model has an average accuracy of 80%. I think the accuracy is still really good and since **Gradient Boosting model is** an easy-to-use model, we will try to increase its performance even further in the following section.

Model Evaluation

```
#importing
#classification Report
#confusion_matrix
#f1_score
#roc_auc_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score
from sklearn.metrics import roc_auc_score
#Model Evaluatin
model=[lr,knn,dt,svc,rf,gb]
for m in model:
    m.fit(x_train,y_train)
    pred_train=m.predict(x_train)
    pred_test=m.predict(x_test)
    print("\nReport of ",m, "is")
    print("confussion matrix \n",confusion_matrix(y_test,pred_test))
    print("classification_report  ",classification_report(y_test,pred_test))
    print("f1_score \n",f1_score(y_test,pred_test))
    print("roc auc score \n",roc_auc_score(y_test,pred_test))

```

```
Report of  LogisticRegression() is
confussion matrix
 [[829 104]
 [142 198]]
classification_report                precision    recall  f1-score   support

           0       0.85      0.89      0.87       933
           1       0.66      0.58      0.62       340

    accuracy                           0.81      1273
   macro avg       0.75      0.74      0.74      1273
weighted avg       0.80      0.81      0.80      1273

f1_score
 0.616822429906542
roc auc score
 0.735442279805813

Report of  KNeighborsClassifier() is
confussion matrix
 [[798 135]
 [156 184]]
classification_report                precision    recall  f1-score   support

           0       0.84      0.86      0.85       933
           1       0.58      0.54      0.56       340

    accuracy                           0.77      1273
   macro avg       0.71      0.70      0.70      1273
weighted avg       0.77      0.77      0.77      1273

f1_score
 0.5584218512898331
roc auc score
 0.6982409684130887
```

```
Report of  DecisionTreeClassifier() is
confussion matrix
 [[739 194]
 [144 196]]
classification_report               precision    recall  f1-score   support

           0       0.84      0.79      0.81       933
           1       0.50      0.58      0.54       340

    accuracy                           0.73      1273
   macro avg       0.67      0.68      0.68      1273
weighted avg       0.75      0.73      0.74      1273

f1_score
 0.5369863013698629
roc auc score
 0.6842695920812054

Report of  SVC() is
confussion matrix
 [[840  93]
 [168 172]]
classification_report               precision    recall  f1-score   support

           0       0.83      0.90      0.87       933
           1       0.65      0.51      0.57       340

    accuracy                           0.79      1273
   macro avg       0.74      0.70      0.72      1273
weighted avg       0.78      0.79      0.79      1273

f1_score
 0.5685950413223141
roc auc score
 0.7031019481747683

Report of  RandomForestClassifier() is
confussion matrix
 [[832 101]
 [168 172]]
classification_report               precision    recall  f1-score   support

           0       0.83      0.89      0.86       933
           1       0.63      0.51      0.56       340

    accuracy                           0.79      1273
   macro avg       0.73      0.70      0.71      1273
weighted avg       0.78      0.79      0.78      1273

f1_score
 0.5611745513866232
roc auc score
 0.6988147027299666
```

```
Report of  GradientBoostingClassifier() is
confussion matrix
 [[840  93]
 [149 191]]
classification_report               precision    recall  f1-score   support

           0       0.85      0.90      0.87       933
           1       0.67      0.56      0.61       340

    accuracy                           0.81      1273
   macro avg       0.76      0.73      0.74      1273
weighted avg       0.80      0.81      0.80      1273

f1_score
 0.6121794871794871
roc auc score
 0.7310431246453566
```

Hyper Tuning

Hyperparameters are crucial as they control the overall behavior of a machine learning model. The ultimate goal is to find an optimal combination of hyperparameters that minimizes a predefined loss function to give better results. Here we first find the best parameters for the Random Forest Model and indulge it to the model to improve our predicting accuracy. There are several ways for finding the parameters, here we use the most powerful and most commonly used method named GridSearchCV.

```
#Hyper parameter Tuning
#Gradient Boosting
#using GridSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier
parameters={"n_estimators" :[1,10,50,100,200], "random_state": list(range(0,10)),
               'learning_rate':[0.15,0.1,0.05,0.01,0.005,0.001]}
gb=GradientBoostingClassifier()
clf=GridSearchCV(gb,parameters)
clf.fit(x_train,y_train)
print(clf.best_params_)

```

```
{'learning_rate': 0.05, 'n_estimators': 100, 'random_state': 0}
```

Optimized Model by tuning their hyperparameters

```
#using GradientBoostingClassifier with best Result
from sklearn.ensemble import GradientBoostingClassifier
gb=GradientBoostingClassifier(random_state=0,n_estimators=100,learning_rate=0.05)
gb.fit(x_train,y_train)
gb_test_pred=gb.predict(x_test)
gb_train_pred=gb.predict(x_train)
gb_test_acc=accuracy_score(y_test,gb_test_pred)
gb_train_acc=accuracy_score(y_train,gb_train_pred)
print("training accuracy : ",gb_train_acc*100)
print("final accuracy : ",gb_test_acc*100)

```

```
training accuracy :  81.76100628930818
final accuracy :  80.9897879025923
```

Further Evaluation

1)Cross-validation

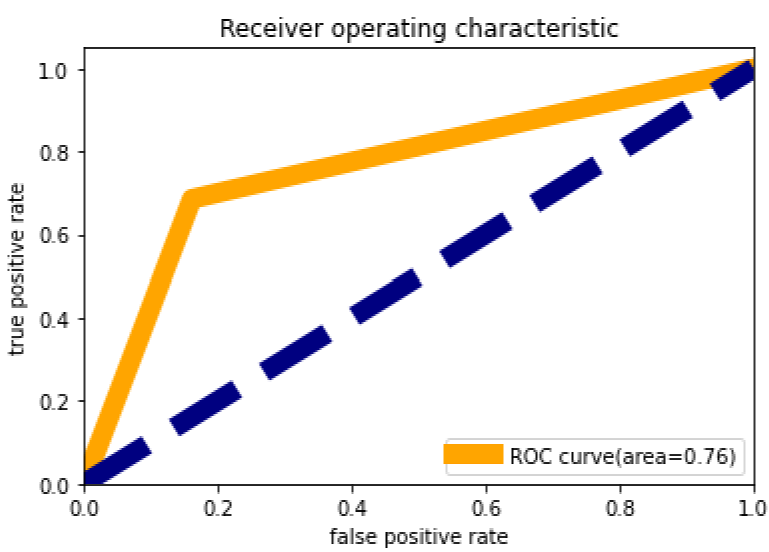
The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and to give an insight on how the model will generalize to an independent dataset (i.e., an unknown dataset, for instance from a real problem).

```
At cv :  8
cross val score is :  80.44327375873075
accuracy score is:  80.67556952081696
******************************************************
```

Here we find the cross-validation score and the accuracy score is almost nearer which shows that we are approaching in a good way.

2) AUC ROC CURVE FOR GRADIENTBOOSTING CLASSIFIER

```
from sklearn.metrics import roc_curve,auc
fpr,tpr,thresholds=roc_curve(gb_test_pred,y_test)
roc_auc=auc(fpr,tpr)
plt.figure()
plt.plot(fpr,tpr,color="orange",lw=10,label="ROC curve(area=%0.2f)"%roc_auc)
plt.plot([0,1],[0,1],color="navy",lw=10,linestyle="--")
plt.xlim([0.0,1.0])
plt.ylim([0.0,1.05])
plt.xlabel("false positive rate")
plt.ylabel("true positive rate")
plt.title("Receiver operating characteristic")
plt.legend(loc="lower right")
plt.show()
```



Summary

We started with the data exploration where we got a feeling for the dataset, checked about missing data and learned which features are important. During this process we used seaborn, matplotlib to do the visualizations. During the data preprocessing part, we computed missing values, converted features into numeric ones, grouped values into categories and created a few new features. Afterwards we started training 6 different machine learning models, picked one of them (Gradient Boost) and applied cross validation on it. Then we discussed how Gradient Boost works, took a look at the importance it assigns to the different features and tuned its performance through optimizing it's hyperparameter values.

Conclusion

Customer churn analysis allows to minimize acquisition costs and increase marketing efficiency, preparing a solid base for future marketing analysis and campaigns. Customer churn analysis opens new opportunities for cross-selling and upselling and serves as one of the starting points for customer-driven product development, keeping customers engaged and loyal over time.