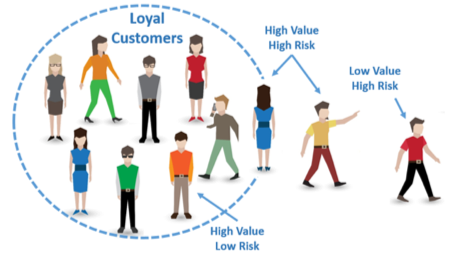
**Classification Model (Customer Churn)**



Customer Churn means prediction which customer are likely to cancel the subscription to a service based on how they use the service. It mainly happens in Tele communication company. Different customer exhibit different behaviours and preference, so they cancel the subscription. It will mostly affect in tele communication department.

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.

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.

Import the Basic Libraires

Pandas – Allow Importing Data from various files like csv, json etc.

Numpy – Numerical Python Consisting of Array.

Seaborn - It is a Data Visualization Library.

Matplotlib – It is used for Plotting the Graph.

Warnings – To filter the warnings.

*##### Load the Basic Libraries*

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **seaborn** **as** **sns**

**import** **matplotlib.pyplot** **as** **plt**

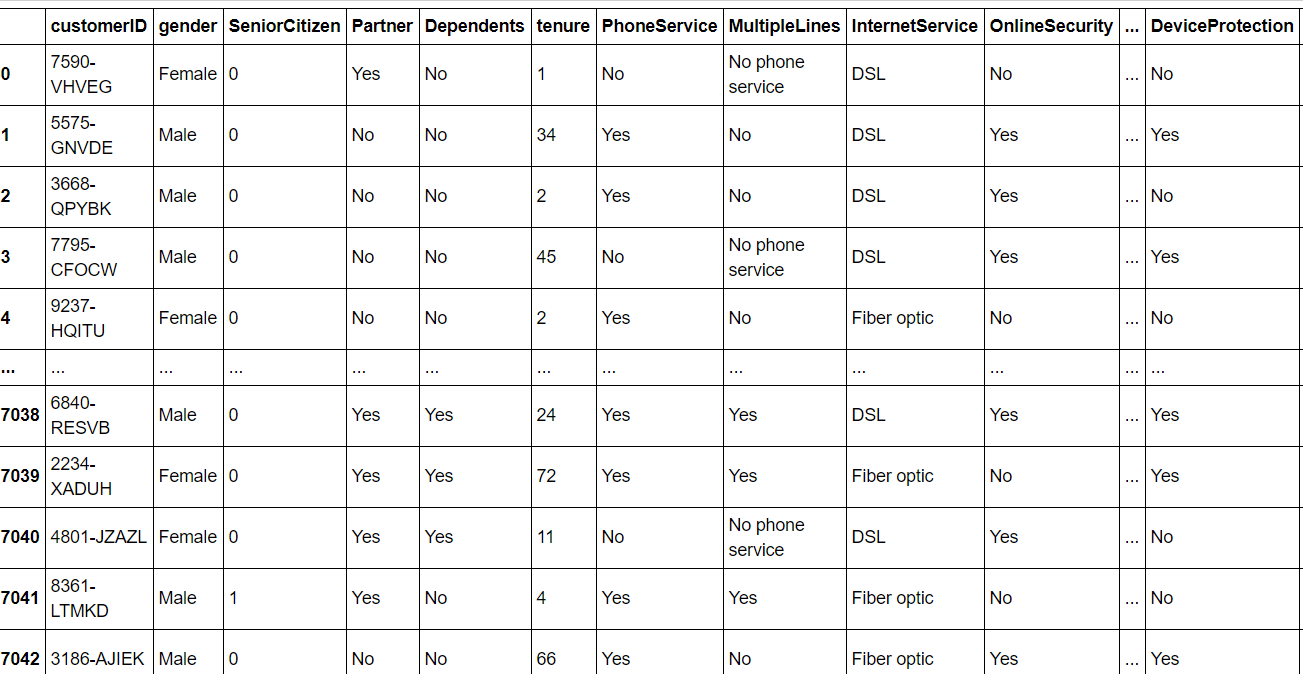
**import** **warnings**

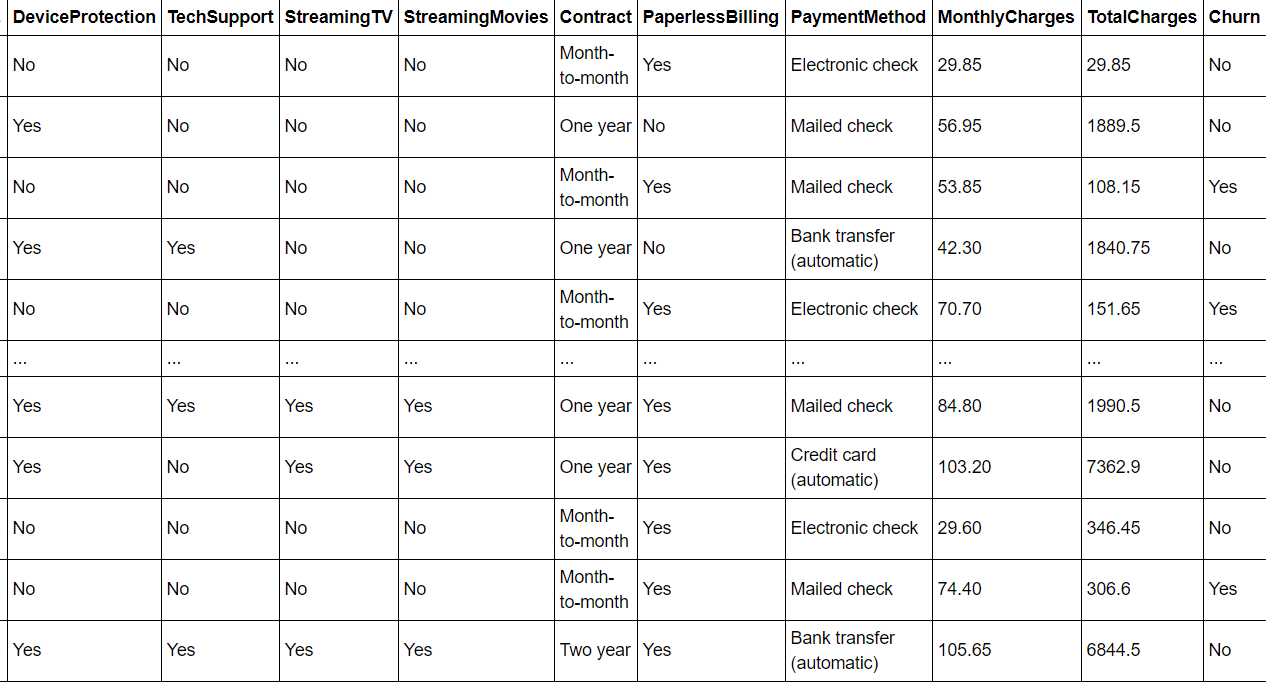
warnings.filterwarnings('ignore')

* Load the Dataset

df = pd.read\_csv('customer\_churn.csv')

df





**Start with Exploratory Data Analysis**

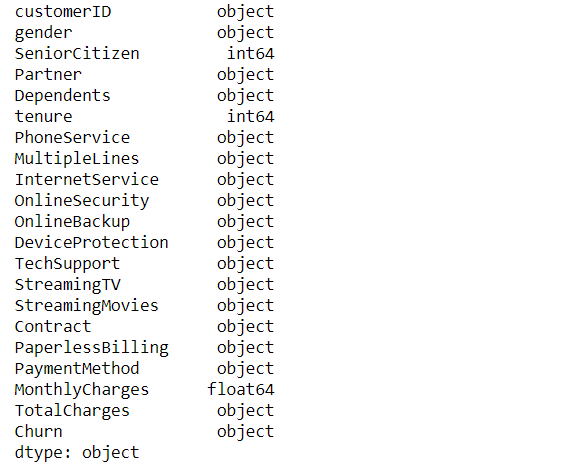
Check the shape of columns

df.shape



Check for Datatypes of all columns

df.dtypes

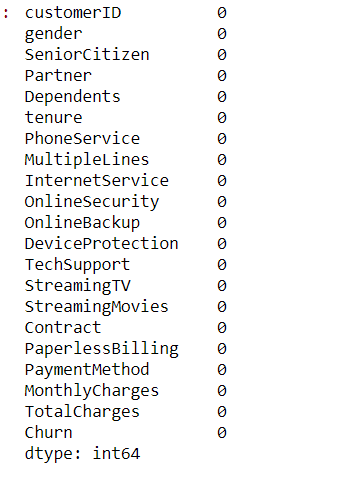


Most of the columns having object type data type. We will handle separately for categorical columns and Continuous columns. Separation helps to apply encoding techniques easy.

Check for null values.

*#### check the count of the null values*

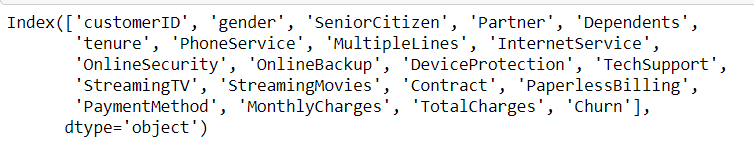
df.isnull().sum()



It seems to no Null value is showing but Total Charges is in Object type means it have some string value so during Pre-Processing we replace string value with null and then fill the value with mean/median Techniques.

See the columns present in Data

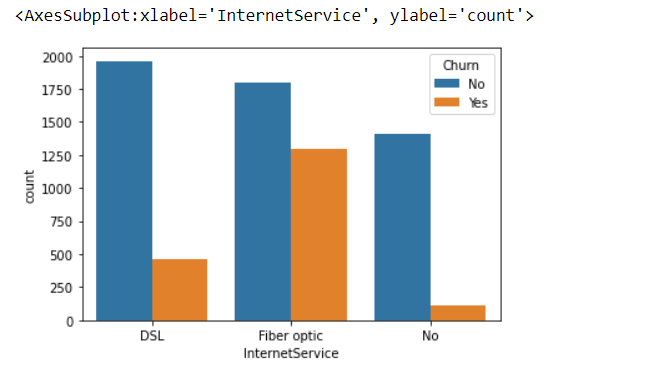
df.columns



**Visualization**

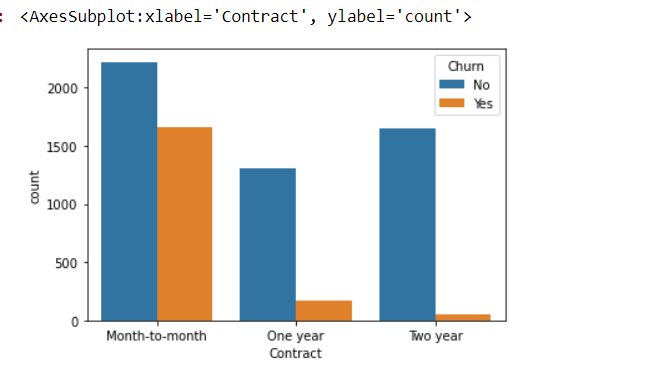
We use seaborn and Matplotlib for visualization of Data. Let’s Understand how dependent variable is related with independent variable.

sns.countplot(x='InternetService',hue='Churn',data=df)



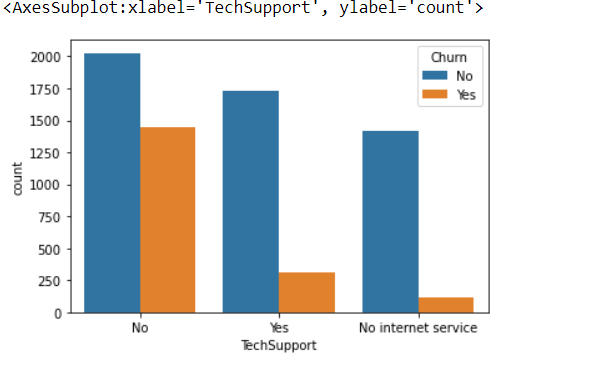
Internet services with fiber optic has maximum customer churned

sns.countplot(x='Contract',hue='Churn',data=df)



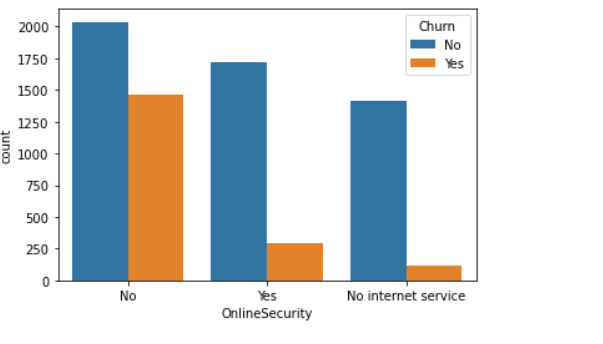
Customer Churned more with Month to Month Contract.

sns.countplot(x='TechSupport',hue='Churn',data=df)

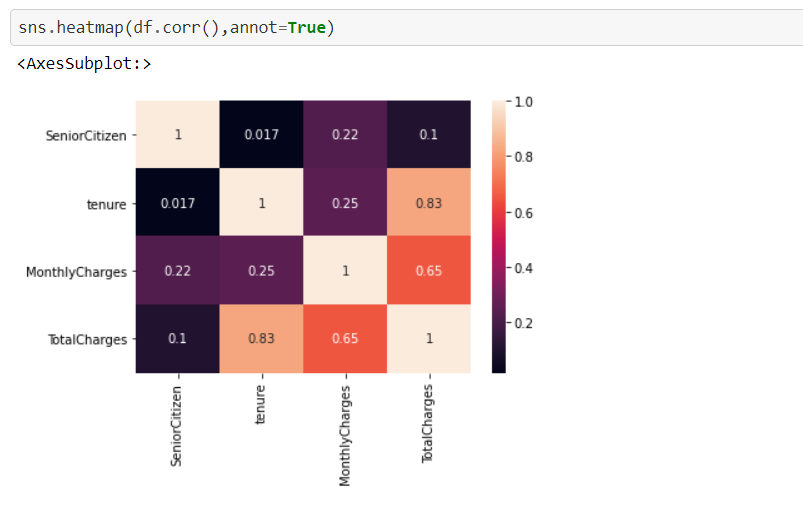


Tech Support services with No has maximum Number customer churned count.

sns.countplot(x='OnlineSecurity',hue='Churn',data=df)

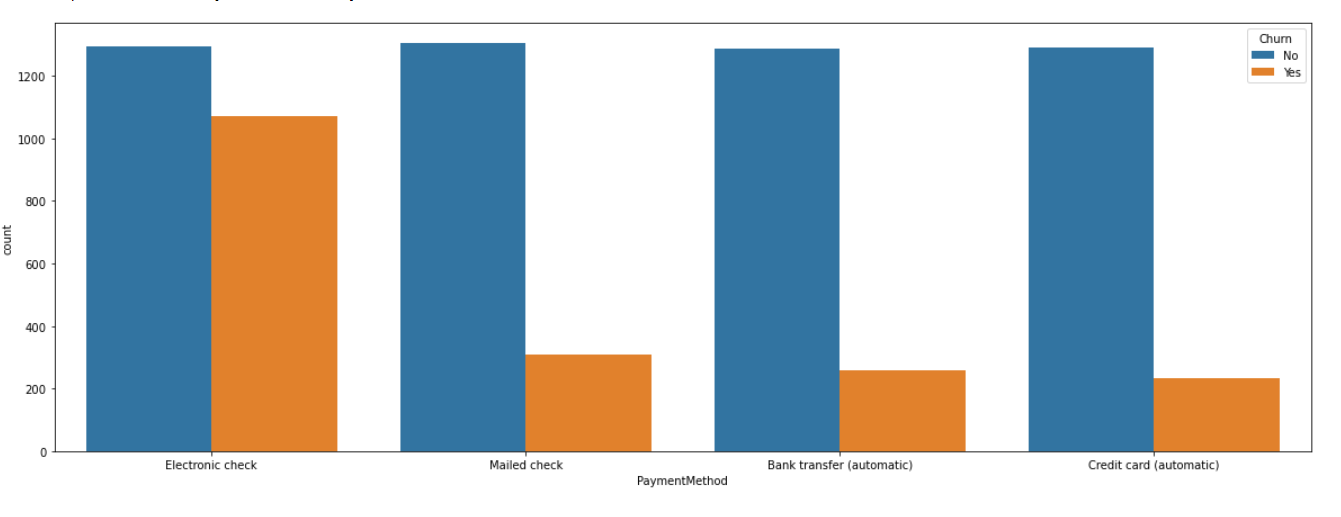


Customer Churned More with No On line Security.



Corelation Graph using heatmap shows how the columns are corelated with target columns .Monthly charges with total charges showing good relation ,similarly tenure with total charges .

plt.figure(figsize=(20,7)) sns.countplot(x='PaymentMethod',hue='Churn',data=df)



Customer use Electronic Check payment method Churned More.

**Pre-Processing Techniques**

* Pre-Processing is a Techniques to evaluate date in understandable format. Real world is often Incomplete, Inconsistence, lots of Missing Value may or may not depends on Data.
* Pre-Processing deals how data is handling so it can be able to predict good score.
* For Better Model Pre-Processing is one of the major role to deal with value contain unwanted stuff like missing value, unwanted keys like ?, ‘-‘ etc,
* In my case Datasets having some string value is Present in total charges because it is in object type so in such case drop string with the Missing value and then fill value.

In case more missing value is present in Data fill missing value with Mode, Median and Mean.

Mode – Fill Categorical Value with Mode.

Median/Mean - For Numerical use Median and Mean.

Let’s check for total charges replace the string value null value and then replace it with by median.

df['TotalCharges'] = df['TotalCharges'].replace([" "],np.nan)

df['TotalCharges'].isnull().sum()



median = df['TotalCharges'].median()

df['TotalCharges']=df['TotalCharges'].fillna(median)

df['TotalCharges']=pd.to\_numeric(df['TotalCharges'])

df['TotalCharges'].dtype

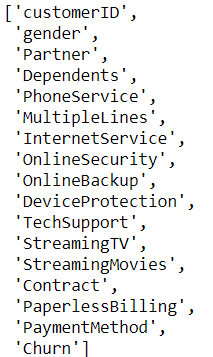


Check for the Categorical Columns

*##### check catagorical columns*

columns =[columns **for** columns **in** df.columns **if** df[columns].dtypes=='object']

columns



Check for the continuous columns

*#### check the continous columns*

cont\_col = [cont\_col **for** cont\_col **in** df.columns **if** df[cont\_col].dtypes!='object']

cont\_col



Check for the outliers

*#### lets handle the outliers* *##### check for the outliers in continous column*

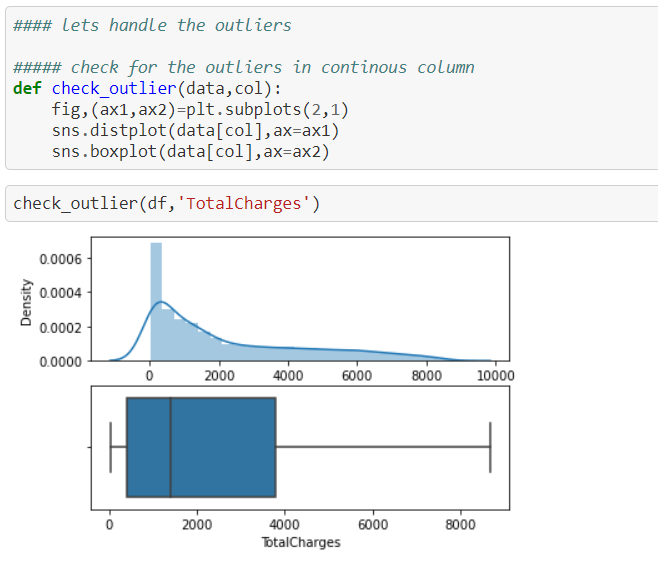
**def** check\_outlier(data,col):

fig,(ax1,ax2)=plt.subplots(2,1)

sns.distplot(data[col],ax=ax1)

sns.boxplot(data[col],ax=ax2)

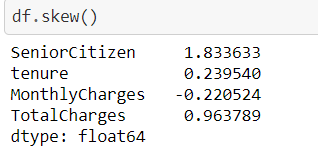
check\_outlier(df,'TotalCharges')



No Outliers is present.

Handle the skewness

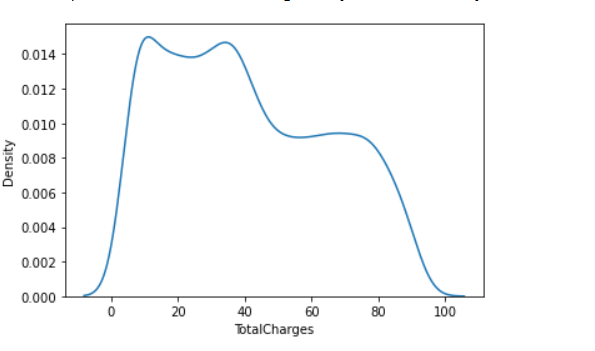
df.skew()



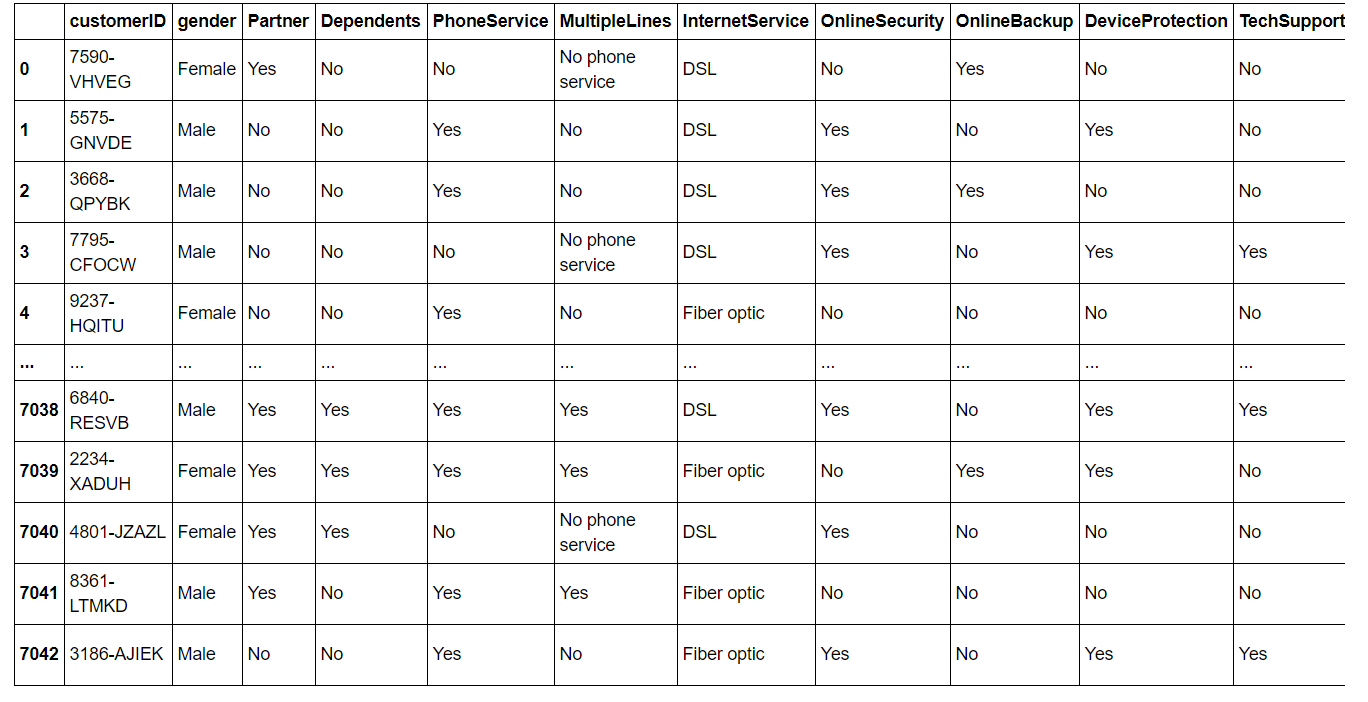
*#### found skewness in total charges which is continous is nature*

df['TotalCharges'] = np.sqrt(df['TotalCharges'])

sns.kdeplot(df['TotalCharges'])

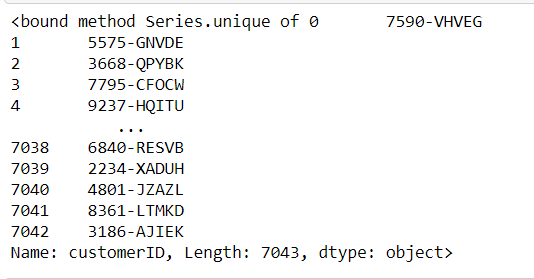


catagorical\_col



Customer ID seems to be unique one

catagorical\_col['customerID'].unique



continous\_column['SeniorCitizen'].unique()



Drop the columns having unique value. Many times it seems to be some columns have unique value and its not effect the data if they can be dropped. In my case I have dropped the unwanted columns.

catagorical\_col.drop(['customerID'],axis=1,inplace=**True**)

*##### drop the Senoior Citizen from Cotinous\_col*

continous\_column.drop(['SeniorCitizen'],axis=1,inplace=**True**)

Scaling is one of the most important techniques.

Data are in order use Label Encoder

For Continuous use standard scaler / min max scaler and more

I have used standardscaler

Import the libraries for scaling

Scaling scale the data. For categorical used Label Encoder for order data and unorder can be handle using One Hot Encoding Techniques.

**import** **sklearn** **from** **sklearn.preprocessing** **import** LabelEncoder

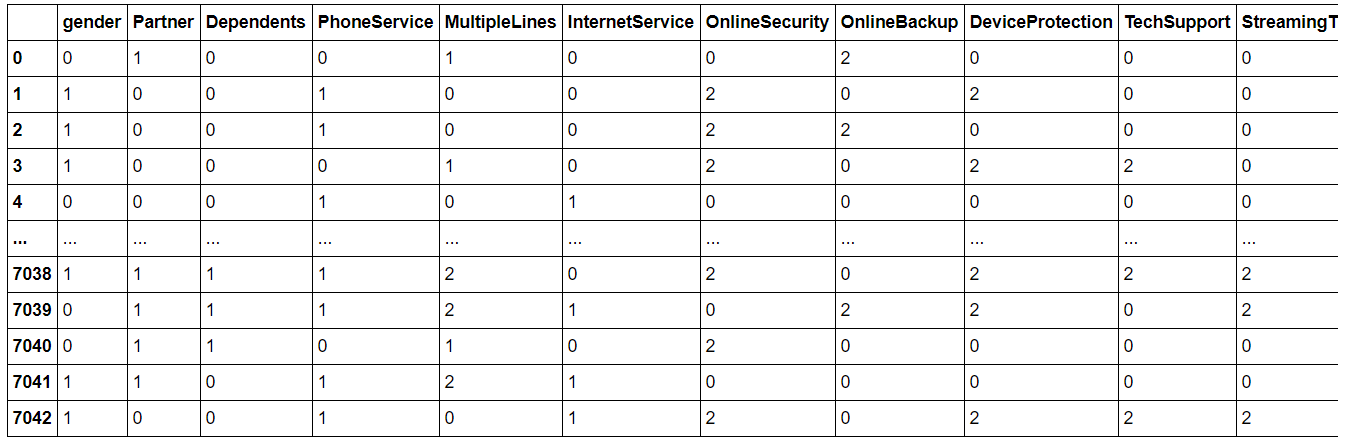
**from** **sklearn.preprocessing** **import** StandardScaler

le=LabelEncoder()

**for** i **in** catagorical\_col.columns:

catagorical\_col[i] = le.fit\_transform(catagorical\_col[i])

catagorical\_col



For continuous Columns I have used standard Scaler.

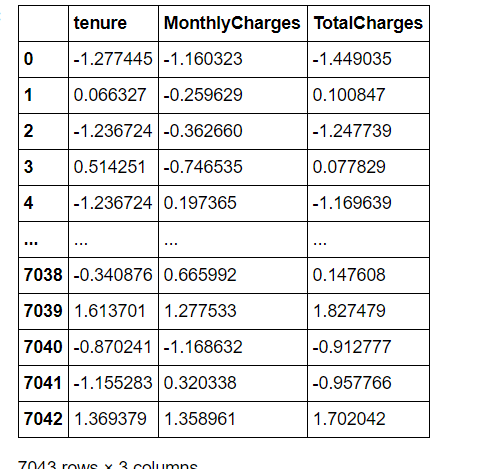
StandardScaler is robust for outliers. It helps to get standardized distribution. It standardizes features by subtracting the mean value from the feature and then dividing the result by feature standard deviation.

sc = StandardScaler()

continous\_column = sc.fit\_transform(continous\_column)

continous\_column = pd.DataFrame(continous\_column,columns=['tenure', 'MonthlyCharges', 'TotalCharges'])

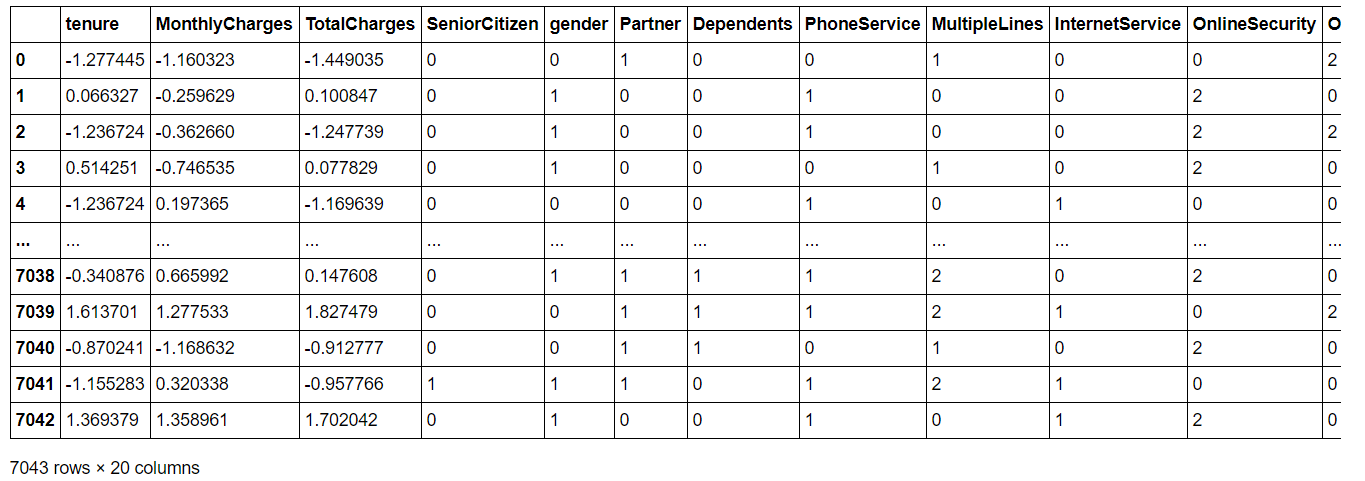
continous\_column



Let’s Prepare the the final Dataset

*##### lets peprare the final dataset* final\_df = pd.concat([continous\_column,senior\_citizen,catagorical\_col],axis=1)

final\_df



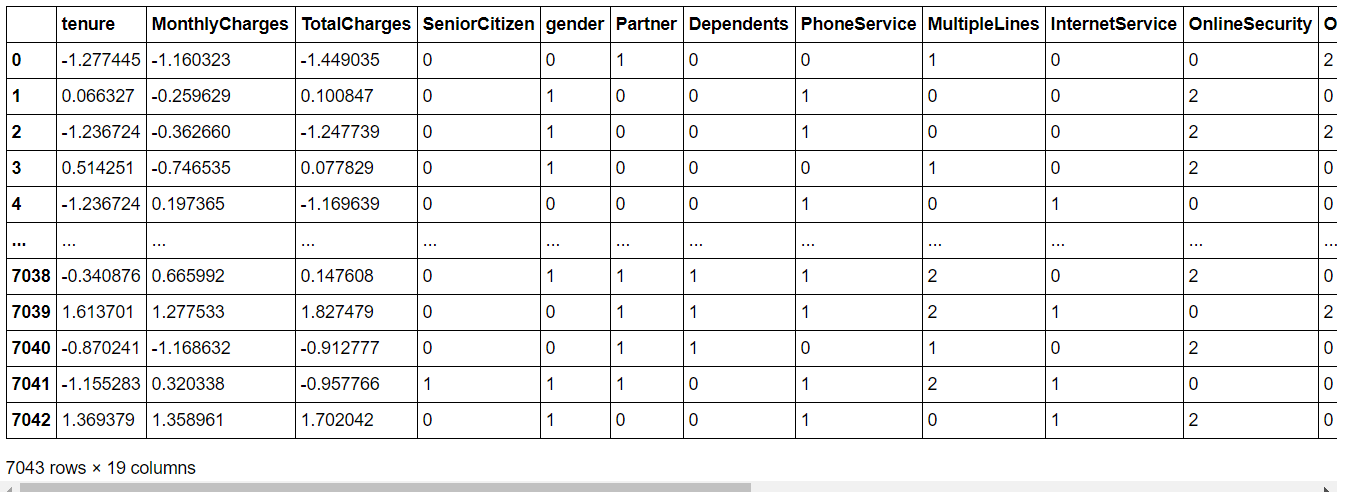
**Modelling**

**Separate the independent columns and Dependent Columns**

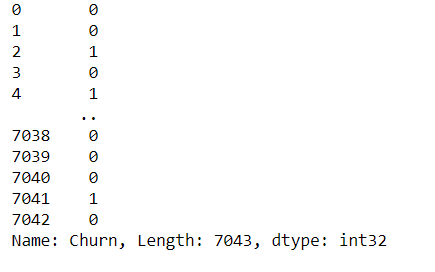
*#### separate x and y from final\_df as feature and target valriable*

x = final\_df.drop(['Churn'],axis=1) y= final\_df['Churn']

x



y



**Import the training model libraries**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **sklearn.ensemble** **import** RandomForestClassifier

**from** **sklearn.ensemble** **import** ExtraTreesClassifier

**from** **sklearn.ensemble** **import** AdaBoostClassifier

**from** **sklearn.svm** **import** SVC

**from** **sklearn.model\_selection** **import** cross\_val\_score

**from** **sklearn.metrics** **import** accuracy\_score,confusion\_matrix,classification\_report

Find the Best Random State. It shows on which Random state the model is scoring good .

*#### Find the best Random State*

maxAcc =0

maxRs =0

**for** i **in** range(1,200):

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.25,random\_state=i)

classifier = LogisticRegression()

classifier.fit(x\_train,y\_train)

pred = classifier.predict(x\_test)

accu = accuracy\_score(y\_test,pred)

**if** accu>maxAcc:

maxAcc = accu

maxRs = i

print('Best Accuracy =',maxAcc,'Best Random State =',maxRs)



Train Test Split

* The **train**-**test** **split** is a technique for evaluating the performance of a machine learning algorithm. It can be used for classification or regression problems and can be used for any supervised learning algorithm. The procedure involves taking a dataset and dividing it into two subsets.

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.25,random\_state=14)

Define Function for Model to Predict

**def** predict(ml\_model):

print('Model is : **{}**'.format(ml\_model))

model = ml\_model.fit(x\_train,y\_train)

prediction = model.predict(x\_test)

Accuracy = accuracy\_score(y\_test,prediction)

classification\_score = classification\_report(y\_test,prediction)

cv\_score = cross\_val\_score(model,x,y,cv=5).mean()

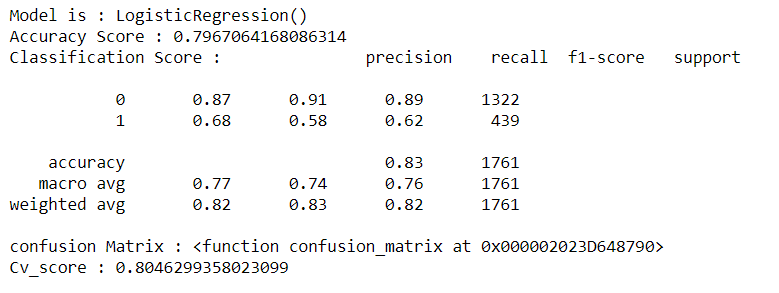
print('Accuracy Score :', accu)

print('Classification Score :', classification\_score)

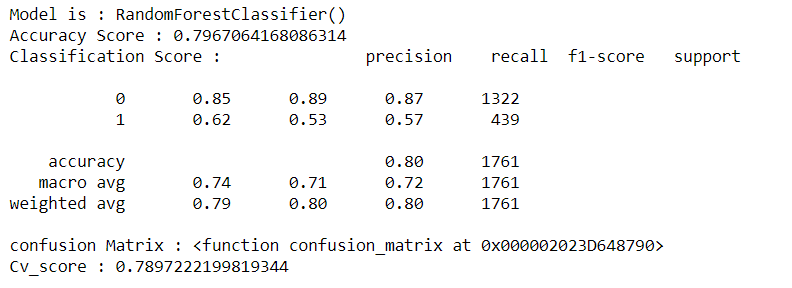
print('confusion Matrix :', confusion\_matrix)

print('Cv\_score :',cv\_score)

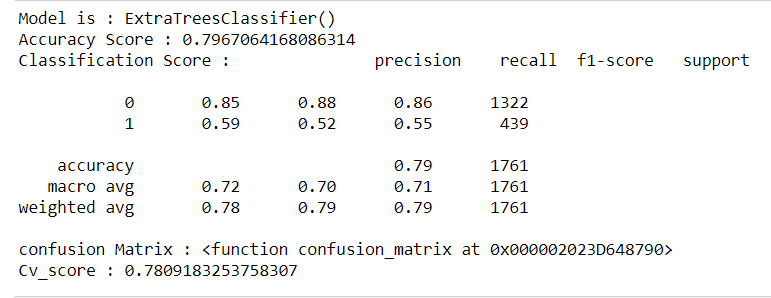
predict(LogisticRegression())



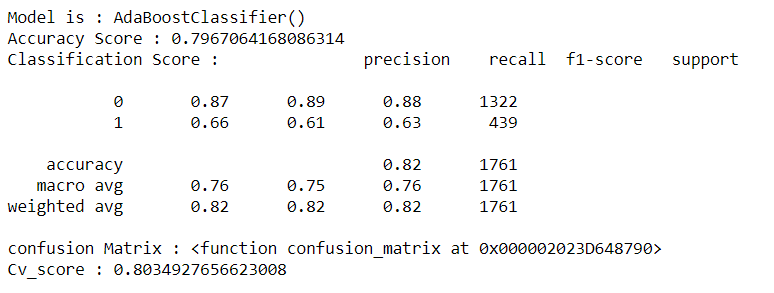
predict(RandomForestClassifier())



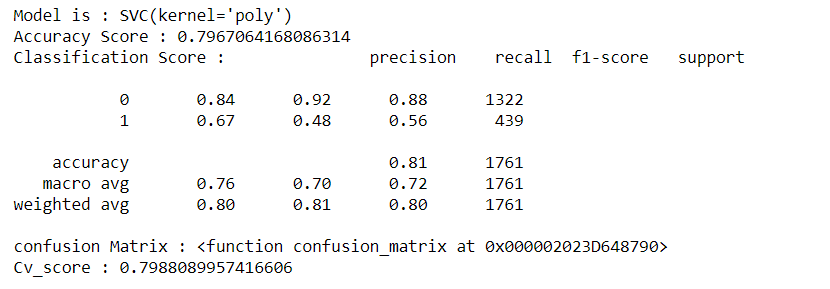
predict(ExtraTreesClassifier())



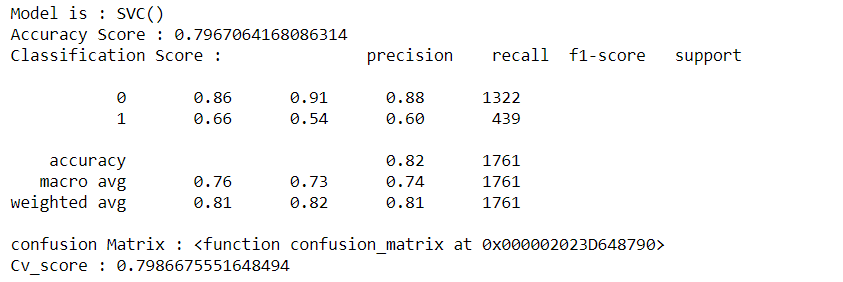
predict(AdaBoostClassifier())



predict(SVC(kernel='poly'))



predict(SVC(kernel='rbf'))



I have Tried different model to predict the score. Found Logistic Regression is best model with good accuracy.

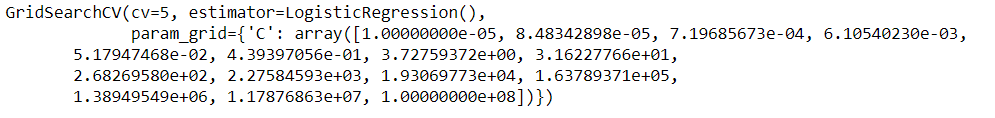
Let’s Hyper tune the model to achieve good score

**from** **sklearn.model\_selection** **import** GridSearchCV

*# Creating the hyperparameter grid*

c\_space = np.logspace(-5, 8, 15) param\_grid = {'C': c\_space}

GCV=GridSearchCV(LogisticRegression(),param\_grid,cv=5) GCV.fit(x\_train,y\_train)



GCV.best\_params\_



Final\_model=LogisticRegression(C=0.4393970560760795)

Final\_model.fit(x\_train,y\_train)

pred=Final\_model.predict(x\_test)

accu=accuracy\_score(y\_test,pred)

print(accu\*100)



**ROC** **curve** is a performance measurement for the classification problems at various threshold settings. **ROC** is a probability curve and **AUC** represents the degree or measure of separability.

Load libraries from sklearn.

*##### Auc And Roc*

**from** **sklearn.metrics** **import** roc\_curve

**import** **matplotlib.pyplot** **as** **plt**

**from** **sklearn.metrics** **import** roc\_auc\_score

Define the model and fit the training data

lr = LogisticRegression()

lr.fit(x\_train,y\_train)

y\_pred\_prob=lr.predict\_proba(x\_test)[:,1]

y\_pred\_prob

fpr,tpr,thresholds=roc\_curve(y\_test,y\_pred\_prob)

plt.plot([0,1],[0,1],'k--')

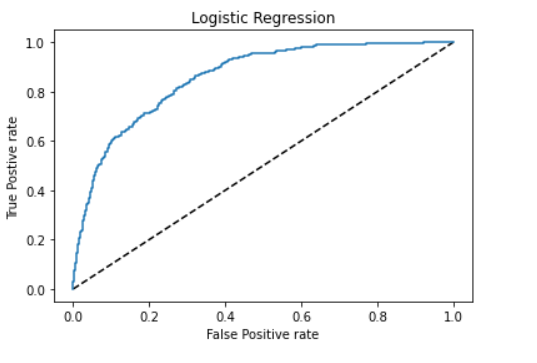
plt.plot(fpr,tpr,label='Logistic Regression')

plt.xlabel('False Positive rate')

plt.ylabel('True Postive rate')

plt.title('Logistic Regression')

plt.show()



auc\_score=roc\_auc\_score(y\_test,lr.predict(x\_test))

auc\_score



**Conclusion**

We have Started with Data Exploratory Analysis where we see how data is Showing. We checked for Missing Value, Columns Present in Datasets, Statistical Information of Data sets.

Use Seaborn and Matplotlib for visualization of data.

During the Data Pre-Processing Part we computed missing values, Some unwanted stuffs present in data like question mark, check for the importance of columns and remove the columns having no importance. We use encoding techniques.

Most of time takes Pre-processing techniques model is good trained when pre-processing is done well.

Predict the model and find the confusion Matrix, CV score, precision, recall , f1 – score

Hyper tune the model to get the better accuracy with Grid search Cv

Check for AUC/ROC Curve how model is representing and it can be more to be predict with very different methods. More visualization helps to understand the dataset clearly.

There is a scope of Improvement Lots of Eda and Data Pre-processing can be done with various techniques.

Feature selection can be done.

Hope this help you .please upvote if like my blog.

Thanks

Kumar Aman