Let's Start The Model Building Part:

In [9]:
```
#Importing Libraries
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from imblearn.combine import SMOTEENN
```

Reading csv

In [11]:
```
df = pd.read_csv("CCA_DATA_MB.csv")
df.head()
```

Out[11]:

	Unnamed: 0	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	...	StreamingMovies_Yes	Contract_Month-to-month	Contract_One year	Contract_Two year	PaperlessBilling_No	PaperlessBilling_Yes	PaymentMethod_Bank transfer (automatic)	PaymentM... car
0	0	0	29	29	1	1	0	0	1	1	...	0	1	0	0	0	1	0	
1	1	0	56	1889	1	0	1	1	0	1	...	0	0	1	0	1	0	0	
2	2	0	53	108	0	0	1	1	0	1	...	0	1	0	0	0	1	0	
3	3	0	42	1840	1	0	1	1	0	1	...	0	0	1	0	1	0	1	
4	4	0	70	151	0	1	0	1	0	1	...	0	1	0	0	0	1	0	

5 rows × 46 columns

In [12]:
```
df.drop(columns="Unnamed: 0",axis=1,inplace=True)
```

In [13]:
```
df.head()
```

Out[13]:

	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	...	StreamingMovies_Yes	Contract_Month-to-month	Contract_One year	Contract_Two year	PaperlessBilling_No	PaperlessBilling_Yes	PaymentMethod_Bank transfer (automatic)	Pay
0	0	29	29	1	1	0	0	1	1	0	...	0	1	0	0	0	1	0	
1	0	56	1889	1	0	1	1	0	1	0	...	0	0	1	0	1	0	0	
2	0	53	108	0	0	1	1	0	1	0	...	0	1	0	0	0	1	0	
3	0	42	1840	1	0	1	1	0	1	0	...	0	0	1	0	1	0	1	
4	0	70	151	0	1	0	1	0	1	0	...	0	1	0	0	0	1	0	

5 rows × 45 columns

creating x and y variables

In [15]:
```
x = df.drop(columns="Churn",axis=1)
y = df["Churn"]
```

Train Test Split

In [17]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
```

In [18]:
```
x_train.head()
```

Out[18]:

	SeniorCitizen	MonthlyCharges	TotalCharges	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneService_No	...	StreamingMovies_Yes	Contract_Month-to-month	Contract_One year	Contract_Two year	PaperlessBilling_No	PaperlessBilling_Yes	PaymentMeth... transfer (a...
1127	0	66	1533	1	0	0	1	0	1	0	...	0	0	1	0	0	1	
6247	0	89	5231	0	1	1	0	1	0	0	...	0	0	0	1	0	1	
265	0	88	5526	1	0	0	1	0	1	0	...	1	0	0	1	1	0	
5791	0	102	6444	0	1	0	1	1	0	0	...	1	1	0	0	0	1	
4408	0	71	5025	1	0	0	1	0	1	0	...	0	0	0	1	0	1	

5 rows × 44 columns

In [19]:
```
x_test.head()
```

Out[19]:

	SeniorCitizen	MonthlyCharges	TotalCharges	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneService_No	...	StreamingMovies_Yes	Contract_Month-to-month	Contract_One year	Contract_Two year	PaperlessBilling_No	PaperlessBilling_Yes	PaymentMeth... transfer (a...
5895	0	59	2341	1	0	0	1	0	1	0	...	0	1	0	0	0	1	
4632	0	19	19	1	0	0	1	0	1	0	...	0	1	0	0	1	0	
5520	0	84	4589	1	0	1	0	1	0	0	...	0	0	1	0	0	1	
1698	0	84	6152	0	1	0	1	0	1	0	...	0	0	0	1	0	1	
4131	0	20	20	1	0	1	0	1	0	0	...	0	1	0	0	0	1	

5 rows × 44 columns

Decision Tree Classifier

In [21]:
```
dt = DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=6,min_samples_leaf=8)
```

In [22]:
```
dt.fit(x_train,y_train)
```

Out[22]:
```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [23]:
```
y_pred = dt.predict(x_test)
y_pred
```

Out[23]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [24]:
```
dt.score(x_test,y_test)
```

Out[24]:
```
0.7899219304471257
```

In [25]:
```
print(classification_report(y_test,y_pred,labels=[1,0]))
```

```
              precision    recall  f1-score   support

           1       0.83      0.90      0.86      1030
           0       0.65      0.48      0.55       379

    accuracy                           0.79      1409
   macro avg       0.74      0.69      0.71      1409
weighted avg       0.78      0.79      0.78      1409
```

As you can see that the accuracy is quite low, and as it's an imbalanced dataset, we shouldn't consider Accuracy as our metrics to measure the model, as Accuracy is cursed in imbalanced datasets. Hence, we need to check recall, precision & f1 score for the minority class, and it's quite evident that the precision, recall & f1 score is too low for Class 0, i.e. churned customers. Hence, moving ahead to call SMOTEENN (UpSampling + ENN)

In [27]:
```
sm = SMOTEENN()
x_resampled,y_resampled = sm.fit_resample(x,y)
```

In [28]:
```
xr_train,xr_test,yr_train,yr_test = train_test_split(x_resampled,y_resampled,test_size=0.2)
```

In [29]:
```
dt_smote = DecisionTreeClassifier(criterion="gini",random_state=100,max_depth=6,min_samples_leaf=8)
```

In [30]:
```
dt_smote.fit(xr_train,yr_train)
```

Out[30]:
```
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [31]:
```
y_pred_smote = dt_smote.predict(xr_test)
y_pred_smote
```

Out[31]:
```
array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

In [32]:
```
dt_smote.score(xr_test, yr_test)
```

Out[32]:
```
0.9368600682593856
```

In [33]:
```
print(classification_report(yr_test,y_pred_smote,labels=[1,0]))
```

```
              precision    recall  f1-score   support

           1       0.95      0.90      0.93       524
           0       0.92      0.96      0.94       648

    accuracy                           0.94      1172
   macro avg       0.94      0.93      0.94      1172
weighted avg       0.94      0.94      0.94      1172
```

In [34]:
```
print(confusion_matrix(yr_test,y_pred_smote))
```

```
[[625  23]
 [ 51 473]]
```

Now we can see quite better results, i.e. Accuracy: 91 %, and a very good recall, precision & f1 score for minority class. Let's try with some other classifier.

Random Forest Classifier

In [37]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [38]:
```
rfc = RandomForestClassifier(n_estimators=100,criterion="gini",random_state=100,max_depth=6,min_samples_leaf=8)
```

In [39]:
```
rfc.fit(x_train,y_train)
```

Out[39]:
```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [40]:
```
y_pred = rfc.predict(x_test)
y_pred
```

Out[40]:
```
array([1, 1, 1, ..., 1, 0, 1], dtype=int64)
```

In [41]:
```
rfc.score(x_test,y_test)
```

Out[41]:
```
0.7913413768630234
```

In [42]:
```
print(classification_report(y_test,y_pred,labels=[1,0]))
```

```
              precision    recall  f1-score   support

           1       0.82      0.92      0.87      1030
           0       0.67      0.45      0.53       379

    accuracy                           0.79      1409
   macro avg       0.74      0.68      0.70      1409
weighted avg       0.78      0.79      0.78      1409
```

In [43]:
```
print(confusion_matrix(y_test,y_pred))
```

```
[[169 210]
 [ 84 946]]
```

Calling SMOTEENN

In [45]:
```
sm = SMOTEENN()
x_resampled1, y_resampled1 = sm.fit_resample(x,y)
```

In [46]:
```
xr_train1,xr_test1,yr_train1,yr_test1 = train_test_split(x_resampled1,y_resampled1,test_size=0.2)
```

In [47]:
```
model_rfc = RandomForestClassifier(n_estimators=100,criterion="gini",max_depth=6,random_state=100,min_samples_leaf=8)
```

In [48]:
```
model_rfc.fit(xr_train1,yr_train1)
```

Out[48]:
```
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [49]:
```
yr_pred1 = model_rfc.predict(xr_test1)
yr_pred1
```

Out[49]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [50]:
```
model_rfc.score(xr_test1,yr_test1)
```

Out[50]:
```
0.9188034188034188
```

In [52]:
```
print(classification_report(yr_test1,yr_pred1,labels=[1,0]))
```

```
              precision    recall  f1-score   support

           1       0.94      0.88      0.91       525
           0       0.91      0.95      0.93       645

    accuracy                           0.92      1170
   macro avg       0.92      0.92      0.92      1170
weighted avg       0.92      0.92      0.92      1170
```

In [53]:
```
print(confusion_matrix(yr_test1,yr_pred1))
```

```
[[614  31]
 [ 64 461]]
```

1167

Accuracy = 93.91%

With Random Forest Classifier, also we are able to get quite good results, infact better than Decision Tree.

Pickling the model

In [66]:
```
import pickle
```

In [70]:
```
filename = "model.sav"
```

In [71]:
```
pickle.dump(model_rfc,open(filename, "wb"))
```

In [72]:
```
load_model = pickle.load,(open(filename, "rb"))
```

Our final model Random Forest Classifier, with SMOTEENN, is now ready and dumped in cca_mb.sav, which we will use and prepare API's so that we can access our model from UI.