Importing Libraries

In [1]:

```
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 [2]:

```
df=pd.read_csv(r"F:\NCPL\Project\Python\tel_churn.csv")
df.head()
```

Out[2]:

	Unnamed: 0	SeniorCitizen	MonthlyCharges	TotalCharges	Churn	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	...	PaymentI transfe
0	0	0	29.85	29.85	0	1	0	0	1	1	...	
1	1	0	56.95	1889.50	0	0	1	1	0	1	...	
2	2	0	53.85	108.15	1	0	1	1	0	1	...	
3	3	0	42.30	1840.75	0	0	1	1	0	1	...	
4	4	0	70.70	151.65	1	1	0	1	0	1	...	

5 rows × 52 columns

In [3]:

```
df=df.drop('Unnamed: 0',axis=1)
```

In [4]:

```
x=df.drop('Churn',axis=1)
x
```

Out[4]:

	SeniorCitizen	MonthlyCharges	TotalCharges	gender_Female	gender_Male	Partner_No	Partner_Yes	Dependents_No	Dependents_Yes	PhoneServi
0	0	29.85	29.85	1	0	0	1	1	0	
1	0	56.95	1889.50	0	1	1	0	1	0	
2	0	53.85	108.15	0	1	1	0	1	0	
3	0	42.30	1840.75	0	1	1	0	1	0	
4	0	70.70	151.65	1	0	1	0	1	0	
...	...	...	...	...	...	...	...	...	...	
7027	0	84.80	1990.50	0	1	0	1	0	1	
7028	0	103.20	7362.90	1	0	0	1	0	1	
7029	0	29.60	346.45	1	0	0	1	0	1	
7030	1	74.40	306.60	0	1	0	1	1	0	
7031	0	105.65	6844.50	0	1	1	0	1	0	

7032 rows × 50 columns

In [5]:

```
y=df['Churn']
y
```

Out[5]:

```
0       0
1       0
2       1
3       0
4       1
       ..
7027    0
7028    0
7029    0
7030    1
7031    0
Name: Churn, Length: 7032, dtype: int64
```

Train Test Split

In [6]:

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Decision Tree Classifier

In [7]:

```
model_dt=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [8]:

```
model_dt.fit(x_train,y_train)
```

Out[8]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [9]:

```
y_pred=model_dt.predict(x_test)
y_pred
```

Out[9]:

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

In [12]:

```
model_dt.score(x_test,y_test)
```

Out[12]:

```
0.7803837953091685
```

In [15]:

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

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.86      1035
           1       0.61      0.47      0.53       372

    accuracy                           0.78      1407
   macro avg       0.72      0.68      0.69      1407
weighted avg       0.77      0.78      0.77      1407
```

In [37]:

```
from imblearn.combine import SMOTEENN
sm = SMOTEENN()
X_resampled, y_resampled = sm.fit_resample(x,y)
```

In [38]:

```
xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_size=0.2)
```

In [39]:
```
model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [42]:
```
model_dt_smote.fit(xr_train,yr_train)
yr_predict = model_dt_smote.predict(xr_test)
model_score_r = model_dt_smote.score(xr_test, yr_test)
print(model_score_r)
print(metrics.classification_report(yr_test, yr_predict))
```

```
0.9390862944162437
              precision    recall  f1-score   support

           0       0.96      0.90      0.93       526
           1       0.92      0.97      0.95       656

    accuracy                           0.94      1182
   macro avg       0.94      0.94      0.94      1182
weighted avg       0.94      0.94      0.94      1182
```

In [41]:
```
print(metrics.confusion_matrix(yr_test, yr_predict))
```

```
[[474  52]
 [ 20 636]]
```

Now we can see quite better results, i.e. Accuracy: 92 %, and a very good recall, precision & f1 score for minority class.

Let's try with some other classifier.

Random Forest Classifier

In [32]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [33]:
```
model_rf=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [34]:
```
model_rf.fit(x_train,y_train)
```

Out[34]:
```
▾                       RandomForestClassifier
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [28]:

```
y_pred=model_rf.predict(x_test)
```

In [29]:

```
model_rf.score(x_test,y_test)
```

Out[29]:

```
0.7924662402274343
```

In [30]:

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

```
              precision    recall  f1-score   support

           0       0.82      0.92      0.87      1035
           1       0.67      0.43      0.52       372

    accuracy                           0.79      1407
   macro avg       0.74      0.68      0.70      1407
weighted avg       0.78      0.79      0.78      1407
```

In [ ]:

In [ ]:

In [44]:

```
sm = SMOTEENN()
X_resampled1, y_resampled1 = sm.fit_resample(x,y)
```

In [45]:

```
xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(X_resampled1, y_resampled1,test_size=0.2)
```

In [46]:

```
model_rf_smote=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [47]:

```
model_rf_smote.fit(xr_train1,yr_train1)
```

Out[47]:

```
▾ RandomForestClassifier
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [51]:

```
yr_predict1 = model_rf_smote.predict(xr_test1)
```

In [52]:

```
model_score_r1 = model_rf_smote.score(xr_test1, yr_test1)
```

In [53]:

```
print(model_score_r1)
print(metrics.classification_report(yr_test1, yr_predict1))
```

```
0.9382924767540152
              precision    recall  f1-score   support

           0       0.95      0.91      0.92       497
           1       0.93      0.96      0.95       686

    accuracy                           0.94      1183
   macro avg       0.94      0.93      0.94      1183
weighted avg       0.94      0.94      0.94      1183
```

In [54]:

```
print(metrics.confusion_matrix(yr_test1, yr_predict1))
```

```
[[450  47]
 [ 26 660]]
```

Performing PCA

In [55]:

```
# Applying PCA
from sklearn.decomposition import PCA
pca = PCA(0.9)
xr_train_pca = pca.fit_transform(xr_train1)
xr_test_pca = pca.transform(xr_test1)
explained_variance = pca.explained_variance_ratio_
```

In [56]:

```
model=RandomForestClassifier(n_estimators=100, criterion='gini', random_state = 100,max_depth=6, min_samples_leaf=8)
```

In [57]:

```
model.fit(xr_train_pca,yr_train1)
```

Out[57]:

```
▾                     RandomForestClassifier
RandomForestClassifier(max_depth=6, min_samples_leaf=8, random_state=100)
```

In [58]:

```
yr_predict_pca = model.predict(xr_test_pca)
```

In [59]:

```
model_score_r_pca = model.score(xr_test_pca, yr_test1)
```

In [60]:

```
print(model_score_r_pca)
print(metrics.classification_report(yr_test1, yr_predict_pca))
```

```
0.7032967032967034
              precision    recall  f1-score   support

           0       0.66      0.60      0.63       497
           1       0.73      0.78      0.75       686

    accuracy                           0.70      1183
   macro avg       0.70      0.69      0.69      1183
weighted avg       0.70      0.70      0.70      1183
```

With PCA, we couldn't see any better results, hence finalising the model which was created by RF Classifier,

Logistic Regression

In [64]:

```
from sklearn.linear_model import LogisticRegression
```

In [65]:

```
model_LR = LogisticRegression()
```

In [66]:

```
model_LR.fit(x_train,y_train)
```

```
C:\ProgramData\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[66]:

```
▾ LogisticRegression
LogisticRegression()
```

In [67]:

```
y_pred=model_LR.predict(x_test)
y_pred
```

Out[67]:
```
array([1, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [68]:
```python
print(classification_report(y_test, y_pred, labels=[0,1]))
```

```
              precision    recall  f1-score   support

           0       0.82      0.91      0.86      1035
           1       0.64      0.45      0.53       372

    accuracy                           0.79      1407
   macro avg       0.73      0.68      0.70      1407
weighted avg       0.77      0.79      0.77      1407
```

By this we can conclude With RF Classifier, also got good results, infact better than Decision Tree.

In [ ]:
```python

```