## Importing the required Libraries

```
In [ ]: 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.metrics import accuracy_score
        from sklearn.tree import DecisionTreeClassifier
        from imblearn.combine import SMOTEENN
```

```
In [ ]: df = pd.read_csv('tel_churn.csv')
        df.drop('Unnamed: 0',axis=1,inplace=True)
```

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

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

## Decision Tree Classifier

```
In [ ]: model_dt=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_depth=6,
```

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

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

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

```
In [ ]: accuracy_score(y_test,y_pred)
```

```
Out[ ]: 0.7789623312011372
```

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

```
              precision    recall  f1-score   support

           0       0.83      0.88      0.85      1020
           1       0.62      0.52      0.56       387

    accuracy                           0.78      1407
   macro avg       0.72      0.70      0.71      1407
weighted avg       0.77      0.78      0.77      1407
```

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 1, i.e. churned customers.

Hence, moving ahead to call SMOTEENN (UpSampling + ENN)

In [ ]:
```
smote = SMOTEENN(random_state=42)
X_resampled, y_resampled = smote.fit_resample(x, y)
```

In [ ]:
```
xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled, y_resampled,test_si
```

In [ ]:
```
model_dt_smote=DecisionTreeClassifier(criterion = "gini",random_state = 100,max_dep
```

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

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

In [ ]:
```
yr_pred = model_dt_smote.predict(xr_test)
```

In [ ]:
```
accuracy_score(yr_test,yr_pred)
```

Out[ ]:
```
0.928087986463621
```

In [ ]:
```
print(classification_report(yr_test,yr_pred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.88      0.91       513
           1       0.92      0.96      0.94       669

    accuracy                           0.93      1182
   macro avg       0.93      0.92      0.93      1182
weighted avg       0.93      0.93      0.93      1182
```

In [ ]:
```
print(confusion_matrix(yr_test,yr_pred))
```

```
[[454  59]
 [ 26 643]]
```

## Random Forest Classifier

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

In [ ]:
```
model_rf = RandomForestClassifier(n_estimators=100,criterion='gini',random_state=10
In [ ]: model_rf.fit(x_train,y_train)
```

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

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

```
In [ ]: accuracy_score(y_test,y_pred)
```

```
Out[ ]: 0.7768301350390903
```

```
In [ ]: sm = SMOTEENN()
        X_resampled1, y_resampled1 = sm.fit_resample(x,y)
        xr_train1,xr_test1,yr_train1,yr_test1=train_test_split(X_resampled1, y_resampled1,t
        model_rf_smote=RandomForestClassifier(n_estimators=100, criterion='gini', random_st
        model_rf_smote.fit(xr_train1,yr_train1)
        yr_predict1 = model_rf_smote.predict(xr_test1)
        model_score_r1 = model_rf_smote.score(xr_test1, yr_test1)
        print(model_score_r1)
        print(metrics.classification_report(yr_test1, yr_predict1))
```

```
0.9369057908383751
              precision    recall  f1-score   support

           0       0.94      0.91      0.93       504
           1       0.93      0.96      0.94       653

    accuracy                           0.94      1157
   macro avg       0.94      0.93      0.94      1157
weighted avg       0.94      0.94      0.94      1157
```

```
In [ ]: print(metrics.confusion_matrix(yr_test1, yr_predict1))
```

```
[[458  46]
 [ 27 626]]
```

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

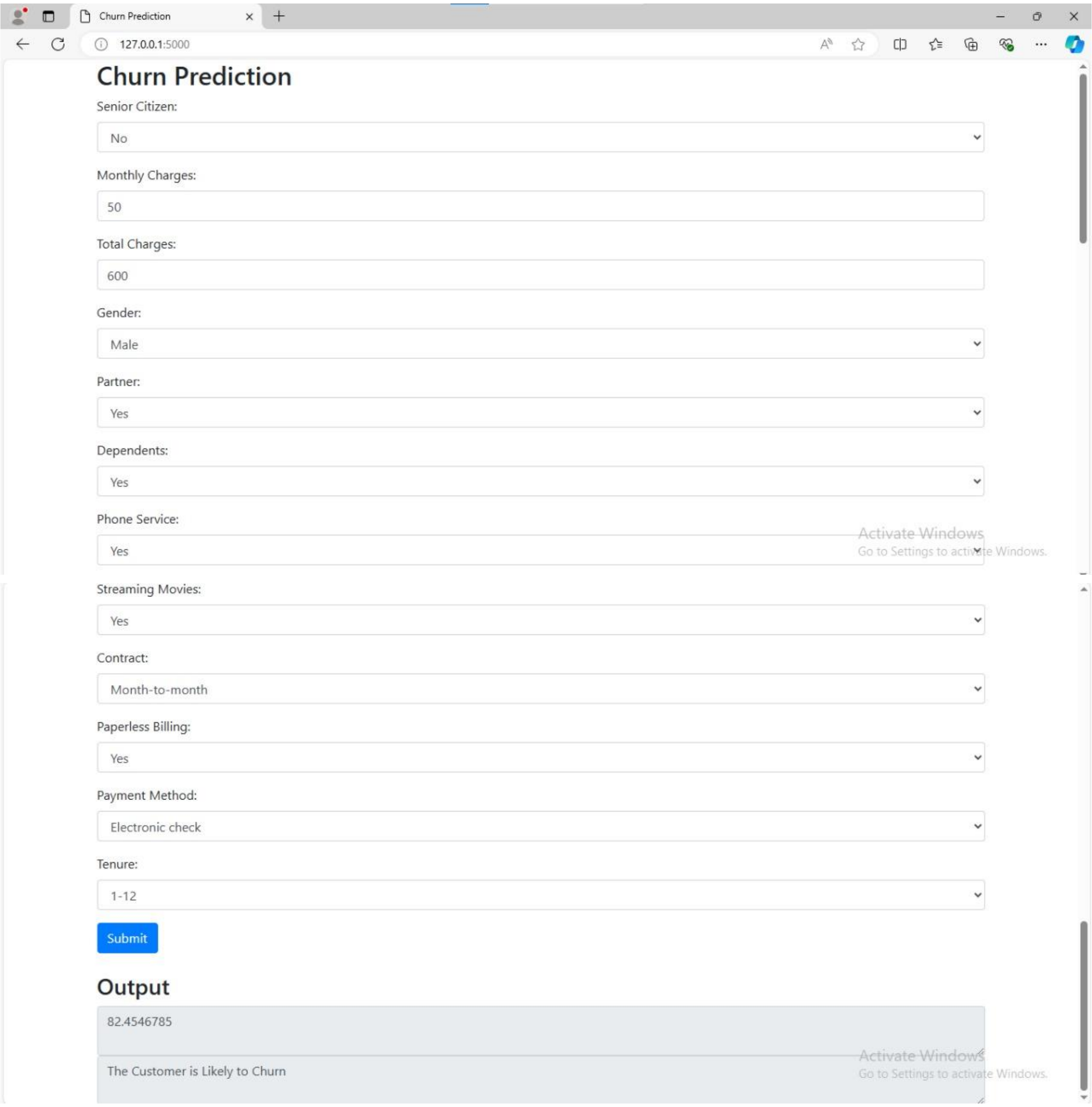
```
In [ ]: import pickle
```

```
In [ ]: filename = 'model.sav'
```

```
In [ ]: pickle.dump(model_rf_smote,open(filename,'wb'))
```

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

```
In [ ]:
```



## Churn Prediction

Senior Citizen:

No

Monthly Charges:

50

Total Charges:

600

Gender:

Male

Partner:

Yes

Dependents:

Yes

Phone Service:

Yes

Streaming Movies:

Yes

Contract:

Month-to-month

Paperless Billing:

Yes

Payment Method:

Electronic check

Tenure:

1-12

## Output

82.4546785

The Customer is Likely to Churn