TASK – 3:

Random forest classifier works well with the categorial variable ‘churn’. The missing values in the dataset are automated by this algorithm. As it is a rule-based approach, we don’t need to normalise the data, for it provides better accuracy in the predictions. It reduces the case of overfitting as it takes the average of many decision trees. These are the advantages of using Random Forest for this use case.

Some of the disadvantages include, more training timing, specifying the parameters as many decision trees are involved. This was a disadvantage in this use case as the model’s aim is to predict the churn, while doing so it needed many features and the interpretability was tedious.

```
x = df4.iloc[:, 1:-1].values
y = df4.iloc[:, -1].values
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)
```

```
from imblearn.combine import SMOTEENN
oversample = SMOTEENN()
x_train,y_train = oversample.fit_resample(x_train,y_train)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
```

```
from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators=20, random_state=0)
classifier.fit(x_train, y_train)
y_pred = classifier.predict(x_test)
print(y_pred)
```

The dataset has class variable “Churn” with imbalanced value counts. Therefore to fit the predictive model in an unbiased way and to evaluate them providing the best fit, the dataset had to be trained with SMOTE technique. This is an oversampling method that transforms the imbalanced dataset into a balanced dataset which ultimately provides results which are unbiased.

The model was able to provide a accuracy rate of 54% when it was not balanced. After incorporating imbalanced dataset measures, the model gave an accuracy output of nearly 84%. It underperformed when the practicality of the class variables where not defined well beforehand.

Random forest classifier along with the SMOTE-ENN(edited nearest neighbour) is appropriate for boosting the model's performance.

Evaluation metrics such as Accuracy rate, Precision, Recall, F1-score is utilised for evaluating the model's performance. This can be derived using a classification report method from the *'sklearn.metrics'* library. In a scenario of choosing only the accuracy would have pushed us with unreliable and dilemma where the accuracy would be more than 95% per se but for actually predicting it would have underperformed. Therefore all of the above-mentioned metrics together would be appropriate measure for evaluating the results of the model's capability.

Client's financial performance can also be evaluated by feeding in the data such as Price elasticity of demand in the excel workbook before loading in the dataset. Thereby we can include them as one of the features while conducting feature engineering process.

Finally, the model's performance is satisfactory as it has been balanced and random forest classifier has provided the following evaluation results which ultimately makes the predictive model reliable.

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

```
[[3037  273]
 [ 309   33]]
              precision    recall  f1-score   support

           0       0.91      0.92      0.91      3310
           1       0.11      0.10      0.10       342

    accuracy                           0.84      3652
   macro avg       0.51      0.51      0.51      3652
weighted avg       0.83      0.84      0.84      3652

0.8406352683461117
```