SUPERVISED ML - CLASSIFICATION MODELS

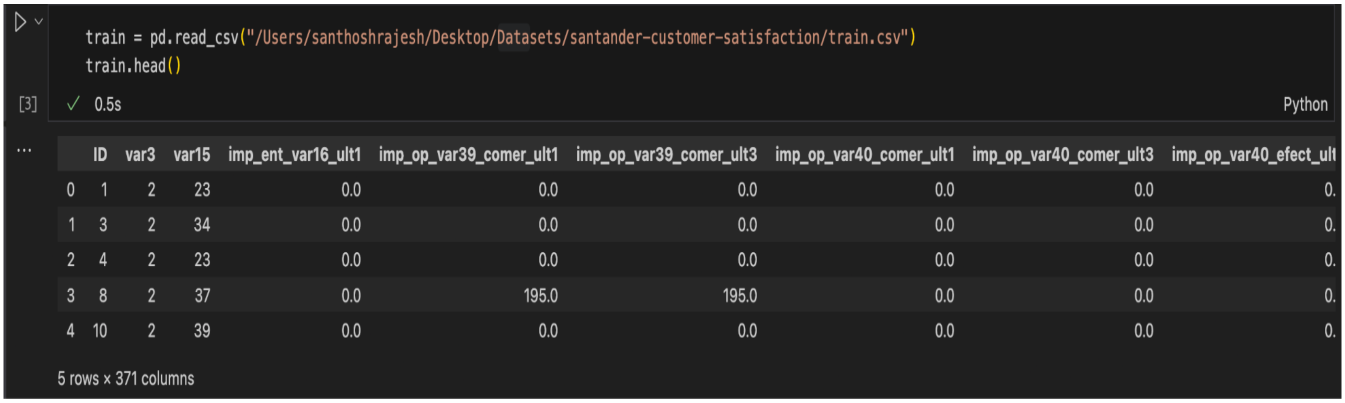
Problem Statement

Customer satisfaction Prediction :

The task is to predict the probability that each customer is an unsatisfied customer or satisfied customer with the features given. It is mostly to classify customer satisfaction.

About the Dataset

An anonymized dataset containing a large number of numeric variables. The "TARGET" column is the variable to predict. It equals one for unsatisfied customers and 0 for satisfied customers.



```
train = pd.read_csv("/Users/santhoshrajesh/Desktop/Datasets/santander-customer-satisfaction/train.csv")
train.head()
```

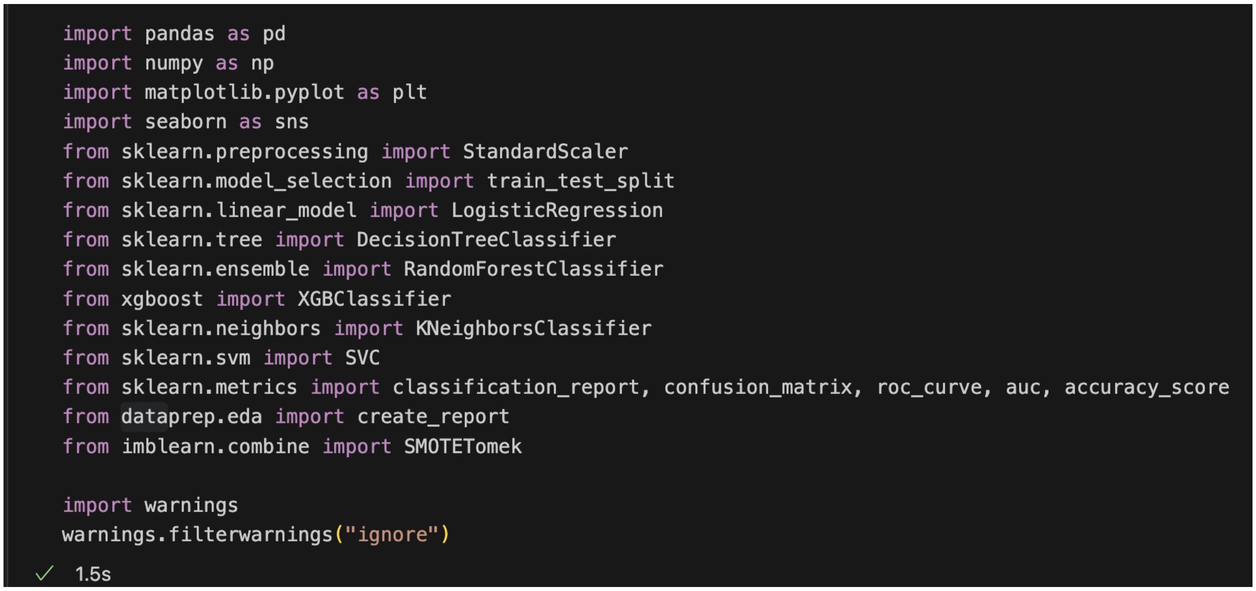
[3] ✓ 0.5s Python

```
   ID  var3  var15  imp_ent_var16_ult1  imp_op_var39_comer_ult1  imp_op_var39_comer_ult3  imp_op_var40_comer_ult1  imp_op_var40_comer_ult3  imp_op_var40_efect_ult
0   1     2     23                 0.0                      0.0                      0.0                      0.0                      0.0                      0.
1   3     2     34                 0.0                      0.0                      0.0                      0.0                      0.0                      0.
2   4     2     23                 0.0                      0.0                      0.0                      0.0                      0.0                      0.
3   8     2     37                 0.0                    195.0                    195.0                      0.0                      0.0                      0.
4  10     2     39                 0.0                      0.0                      0.0                      0.0                      0.0                      0.

5 rows × 371 columns
```

Steps involved in the classification process

1) Import the required libraries



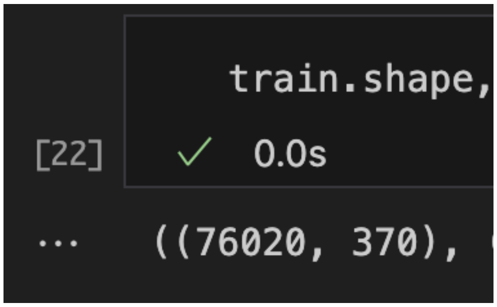
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, roc_curve, auc, accuracy_score
from dataprep.eda import create_report
from imblearn.combine import SMOTETomek

import warnings
warnings.filterwarnings("ignore")
```

✓ 1.5s

2) Exploratory data analysis (EDA)

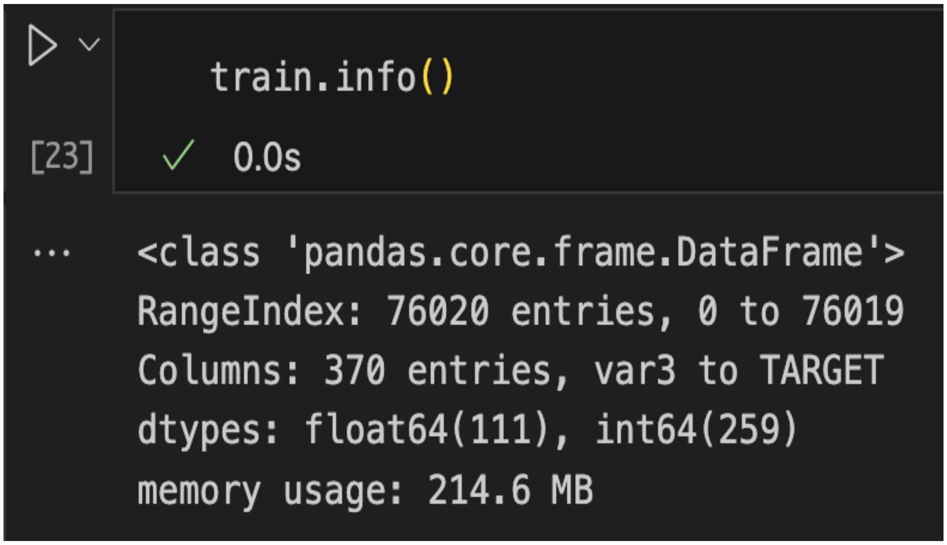
- Shape of the data



```
      train.shape,
[22]   ✓  0.0s

...  ((76020, 370),
```

- Info about the dataset about the columns and its data types

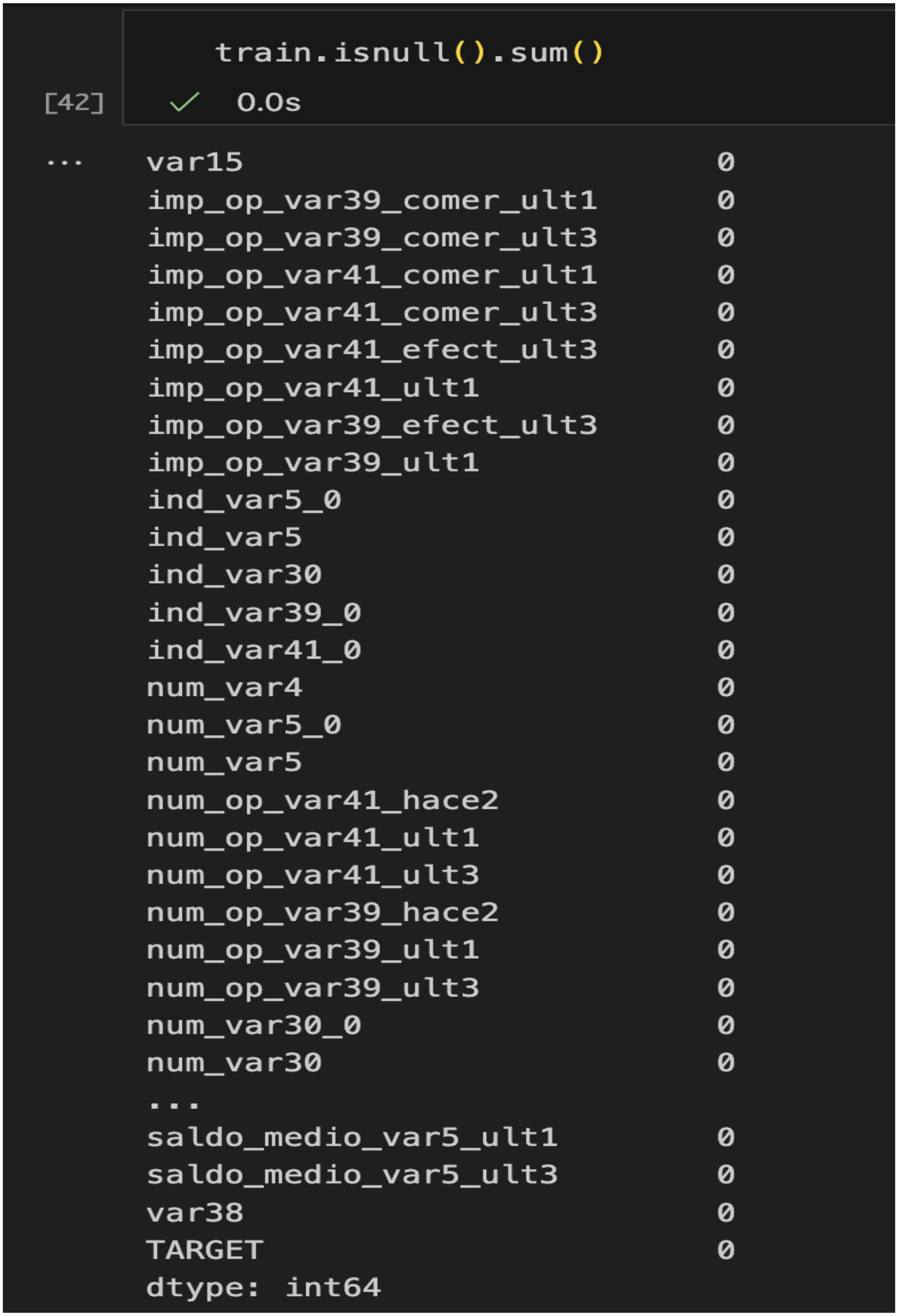


```
       train.info()
[23]   ✓  0.0s

...  <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 76020 entries, 0 to 76019
     Columns: 370 entries, var3 to TARGET
     dtypes: float64(111), int64(259)
     memory usage: 214.6 MB
```

- The variable **var3** has an extreme value **-999999** which is removed.
- The outliers are detected and removed from the dataset. The points which are less than 1st percentile and above the 99 th percentile are removed. This is a common technique to remove extreme values.
- The features with the same 10th and 90th percentile show less variability of the dataset. Features show more variability are selected. Here around **62 features** show more variability.

- To check the missing values in the dataset

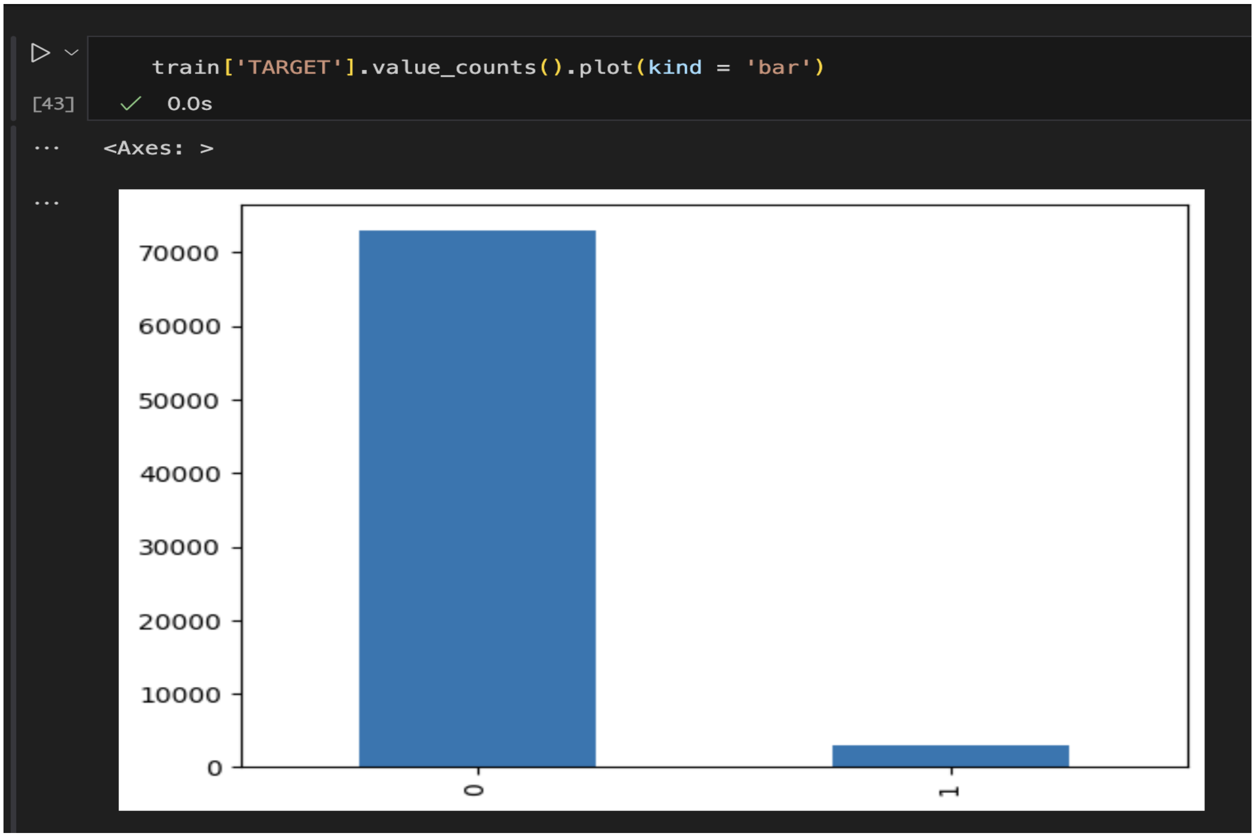


```
        train.isnull().sum()
[42]  ✓  0.0s

...   var15                       0
      imp_op_var39_comer_ult1     0
      imp_op_var39_comer_ult3     0
      imp_op_var41_comer_ult1     0
      imp_op_var41_comer_ult3     0
      imp_op_var41_efect_ult3     0
      imp_op_var41_ult1           0
      imp_op_var39_efect_ult3     0
      imp_op_var39_ult1           0
      ind_var5_0                  0
      ind_var5                    0
      ind_var30                   0
      ind_var39_0                 0
      ind_var41_0                 0
      num_var4                    0
      num_var5_0                  0
      num_var5                    0
      num_op_var41_hace2          0
      num_op_var41_ult1           0
      num_op_var41_ult3           0
      num_op_var39_hace2          0
      num_op_var39_ult1           0
      num_op_var39_ult3           0
      num_var30_0                 0
      num_var30                   0
      ...
      saldo_medio_var5_ult1       0
      saldo_medio_var5_ult3       0
      var38                       0
      TARGET                      0
      dtype: int64
```

3) Data Preprocessing

- Bar chart of counts of unique values in target variable is plotted.

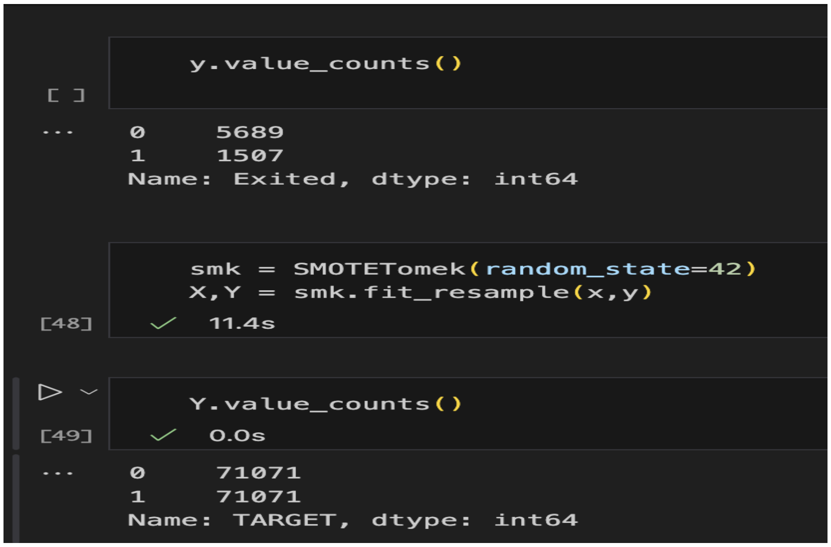


```
train['TARGET'].value_counts().plot(kind = 'bar')
```

```
[43] 0.0s
```

```
<Axes: >
```

- Here clearly the value - 1(satisfied) 3% is dominated by value - 0(unsatisfied) 97%. It indicated the dataset was **imbalanced**. We use SMOTETomek which is a hybrid (combination of both undersampling and oversampling) is used.



```
y.value_counts()
```

```
0     5689
1     1507
Name: Exited, dtype: int64
```

```
smk = SMOTETomek(random_state=42)
X,Y = smk.fit_resample(x,y)
```

```
[48] 11.4s
```

```
Y.value_counts()
```

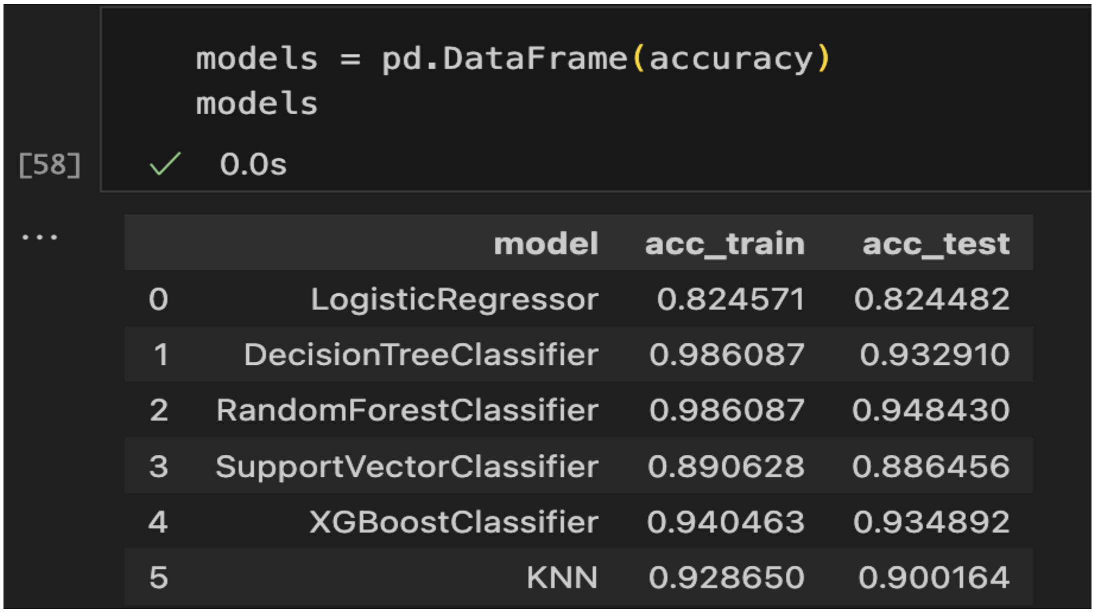
```
[49] 0.0s
```

```
0     71071
1     71071
Name: TARGET, dtype: int64
```

- **Scaling** - The variables are scaled with a standard scaler which uses z-score to scale the values to a specific -ve and +ve range.
- The data is separated into X and Y data and splitted into training and testing sets.

4) Model training

- The models are trained and the accuracy of testing and training sets are calculated.

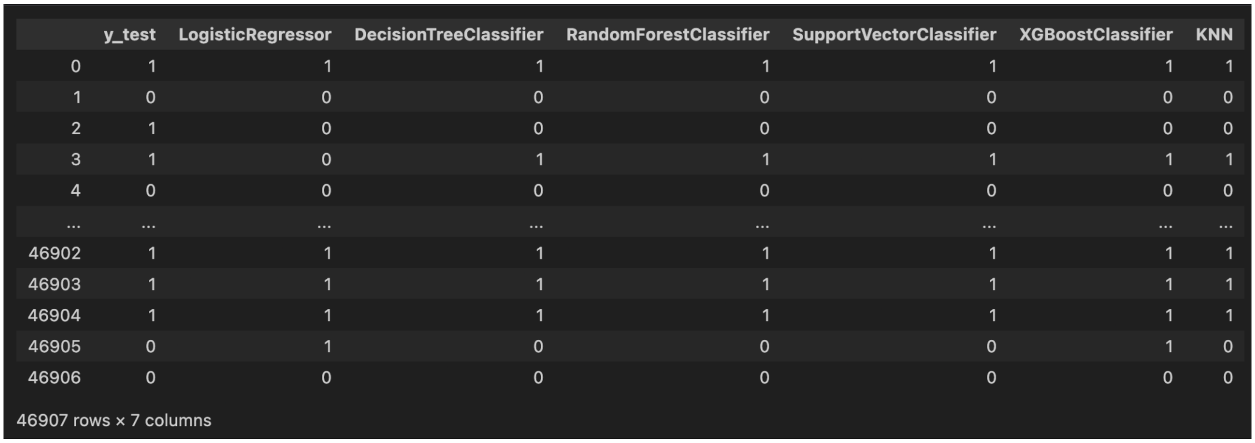


```
models = pd.DataFrame(accuracy)
models
```

[58] ✓ 0.0s

	model	acc_train	acc_test
0	LogisticRegressor	0.824571	0.824482
1	DecisionTreeClassifier	0.986087	0.932910
2	RandomForestClassifier	0.986087	0.948430
3	SupportVectorClassifier	0.890628	0.886456
4	XGBoostClassifier	0.940463	0.934892
5	KNN	0.928650	0.900164

- The predicted values of all the models are compared with the actual values.

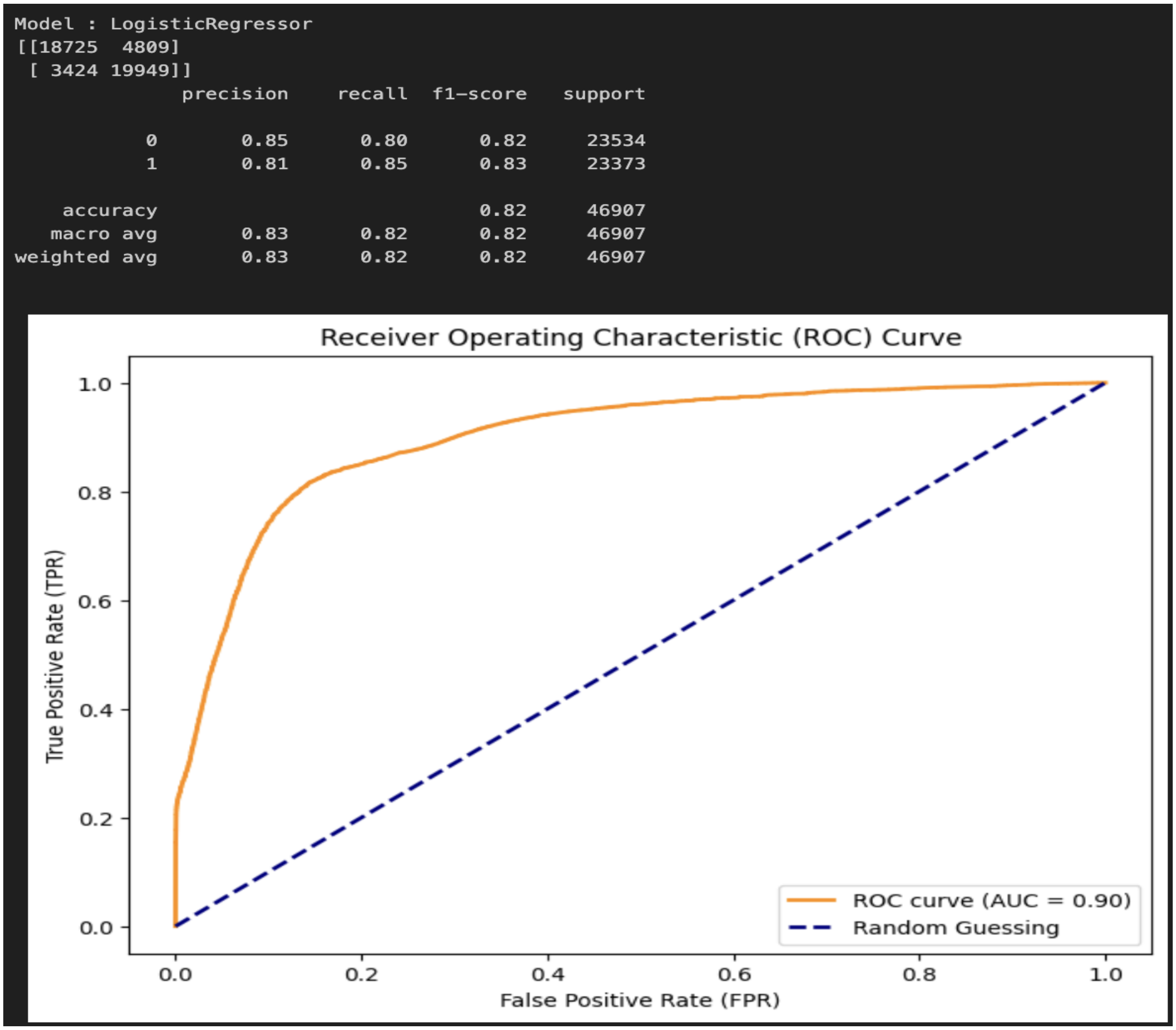


	y_test	LogisticRegressor	DecisionTreeClassifier	RandomForestClassifier	SupportVectorClassifier	XGBoostClassifier	KNN
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	1	0	1	1	1	1	1
4	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...
46902	1	1	1	1	1	1	1
46903	1	1	1	1	1	1	1
46904	1	1	1	1	1	1	1
46905	0	1	0	0	0	1	0
46906	0	0	0	0	0	0	0

46907 rows × 7 columns

- The metrics like **:**
    1. **confusion_matrix** - To find the TP (True positive), FP (False Positive), TN (True Negative) and FN (False Negative).
    2. **Classification report** - To find the precision (Values the are correctly predicted among the predicted values), Recall (Values that are correctly predicted among actual values), f1-score and support.
    3. **ROC curve** - The curve where a random line is drawn and a curve above it refers to a good classification model and the area under it is **AUC** and the curve is plotted with different threshold values. It implies only for binary classification.

Logistic regression

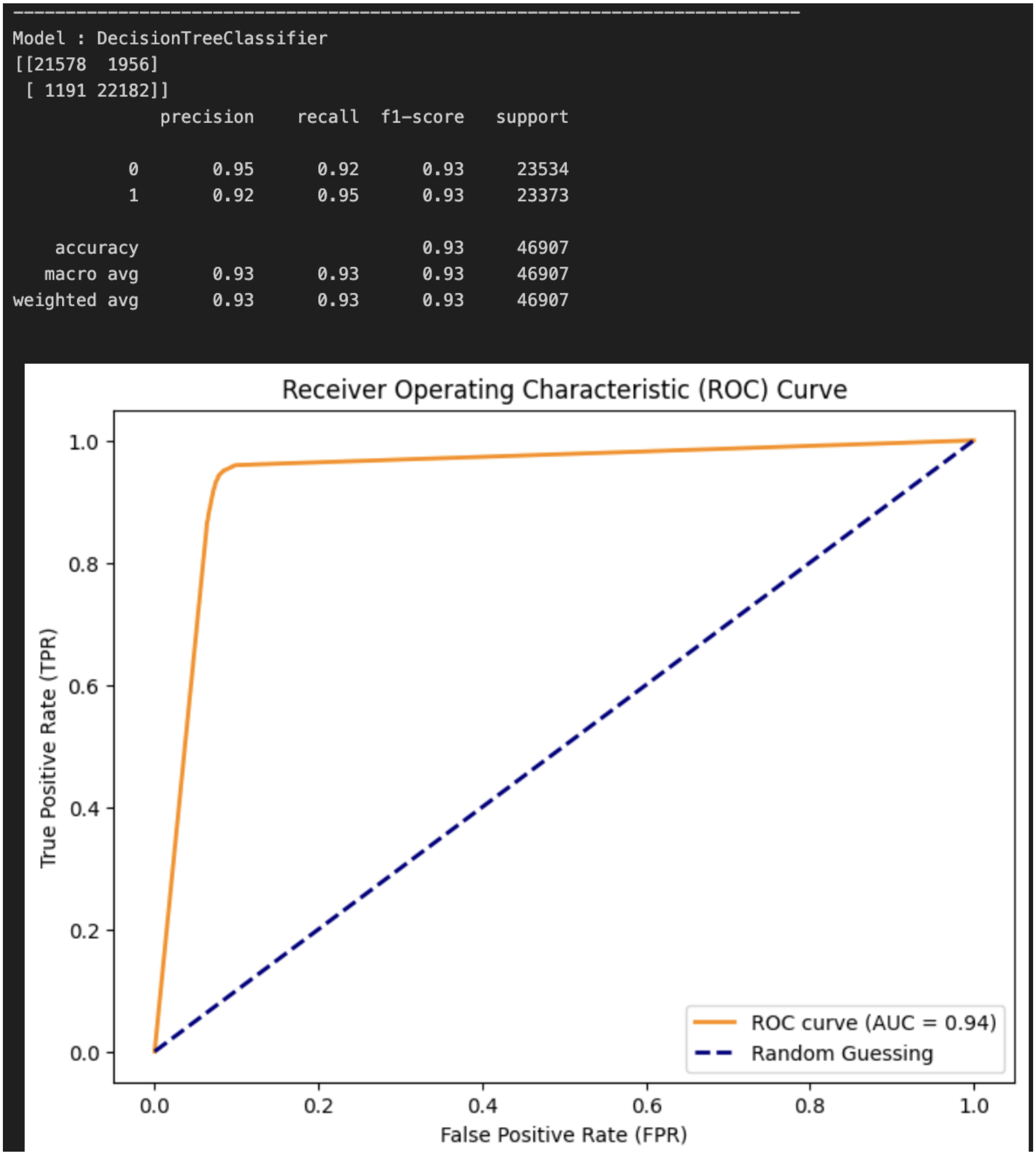


```
Model : LogisticRegressor
[[18725  4809]
 [ 3424 19949]]
              precision    recall  f1-score   support

           0       0.85      0.80      0.82     23534
           1       0.81      0.85      0.83     23373

    accuracy                           0.82     46907
   macro avg       0.83      0.82      0.82     46907
weighted avg       0.83      0.82      0.82     46907
```

Decision Tree

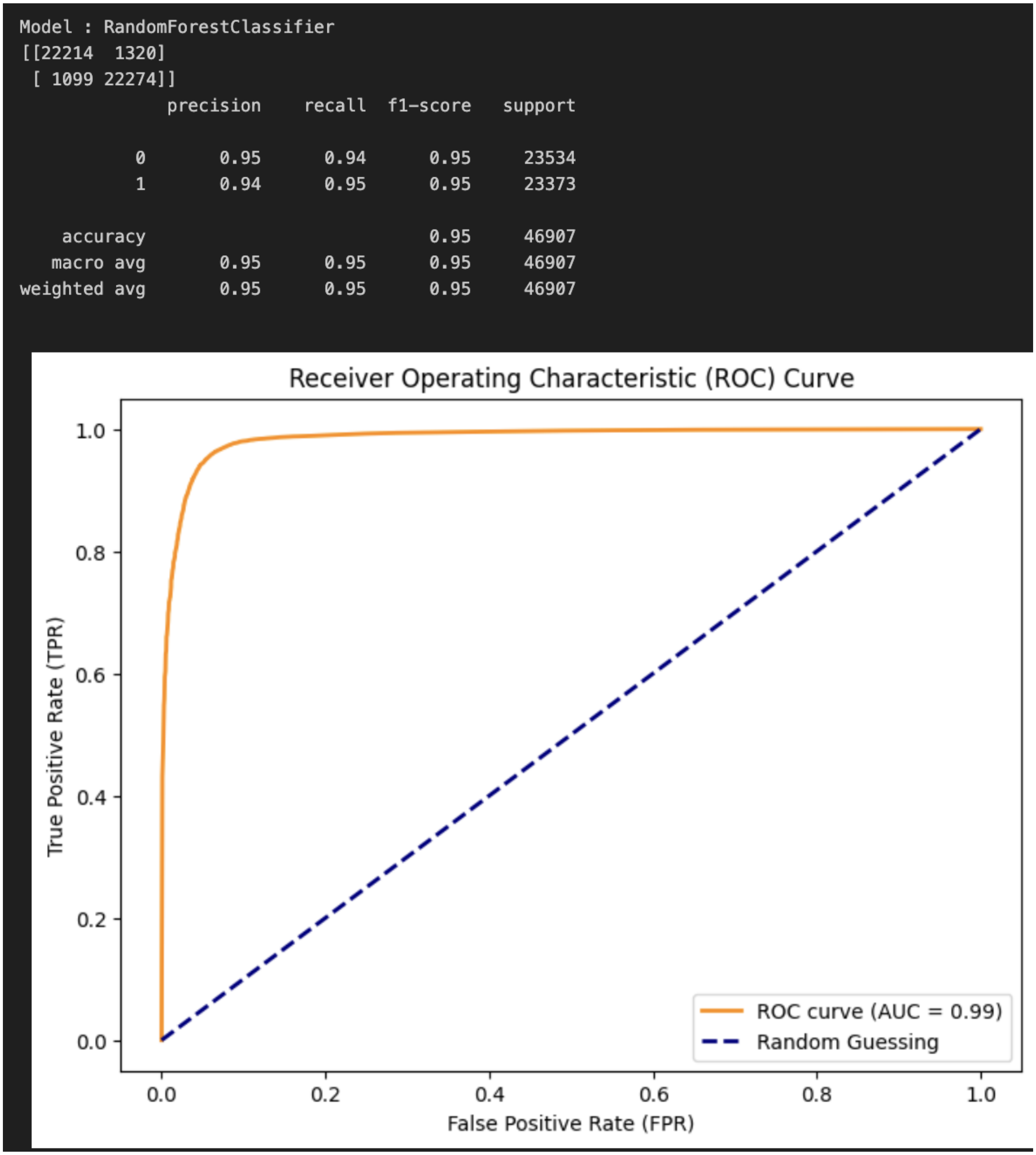


```
--------------------------------------------------------------------------------
Model : DecisionTreeClassifier
[[21578  1956]
 [ 1191 22182]]
              precision    recall  f1-score   support

           0       0.95      0.92      0.93     23534
           1       0.92      0.95      0.93     23373

    accuracy                           0.93     46907
   macro avg       0.93      0.93      0.93     46907
weighted avg       0.93      0.93      0.93     46907
```

Random Forest

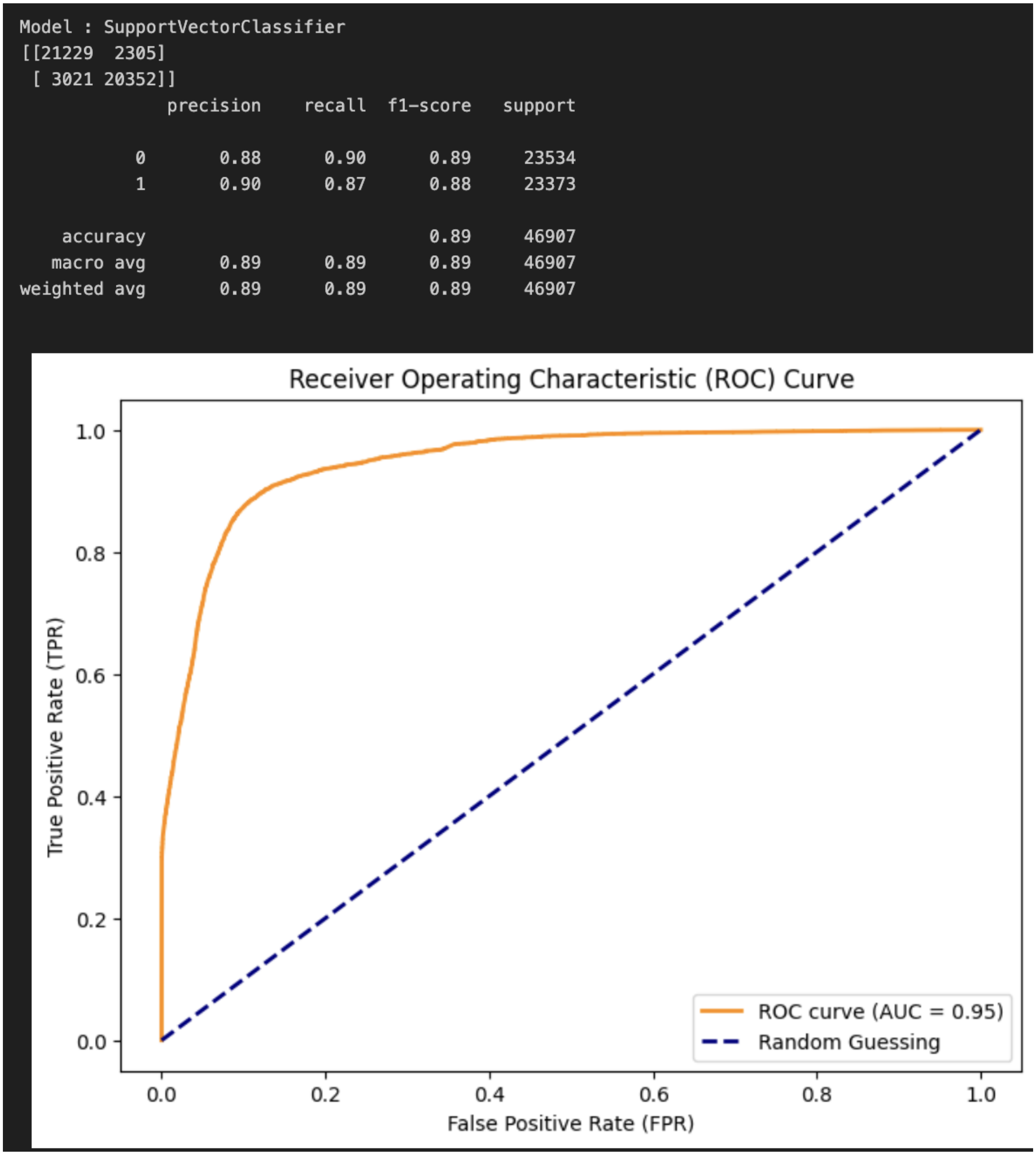


```
Model : RandomForestClassifier
[[22214  1320]
 [ 1099 22274]]
              precision    recall  f1-score   support

           0       0.95      0.94      0.95     23534
           1       0.94      0.95      0.95     23373

    accuracy                           0.95     46907
   macro avg       0.95      0.95      0.95     46907
weighted avg       0.95      0.95      0.95     46907
```

Support Vector Machine

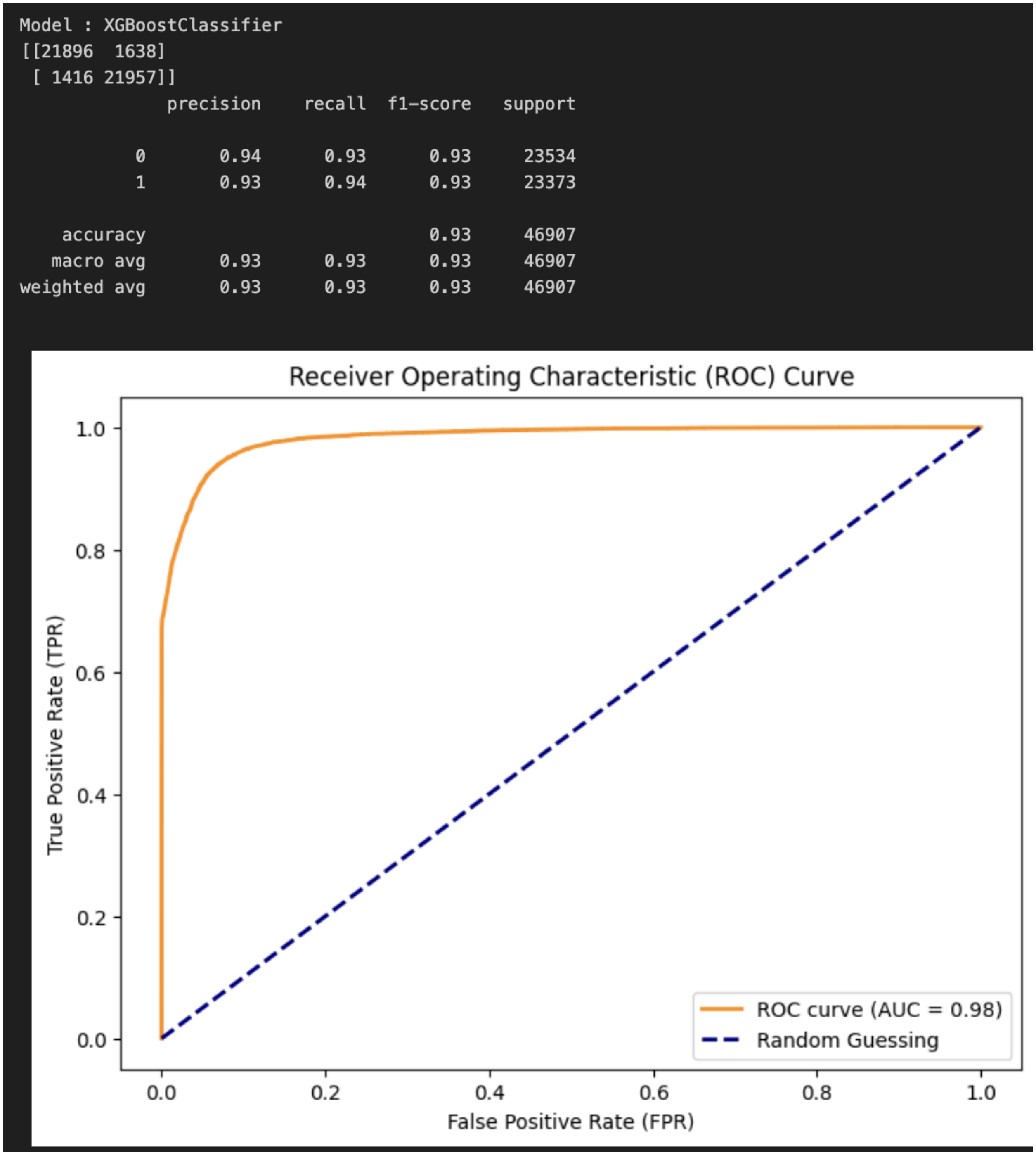


```
Model : SupportVectorClassifier
[[21229  2305]
 [ 3021 20352]]
              precision    recall  f1-score   support

           0       0.88      0.90      0.89     23534
           1       0.90      0.87      0.88     23373

    accuracy                           0.89     46907
   macro avg       0.89      0.89      0.89     46907
weighted avg       0.89      0.89      0.89     46907
```

XGBoost

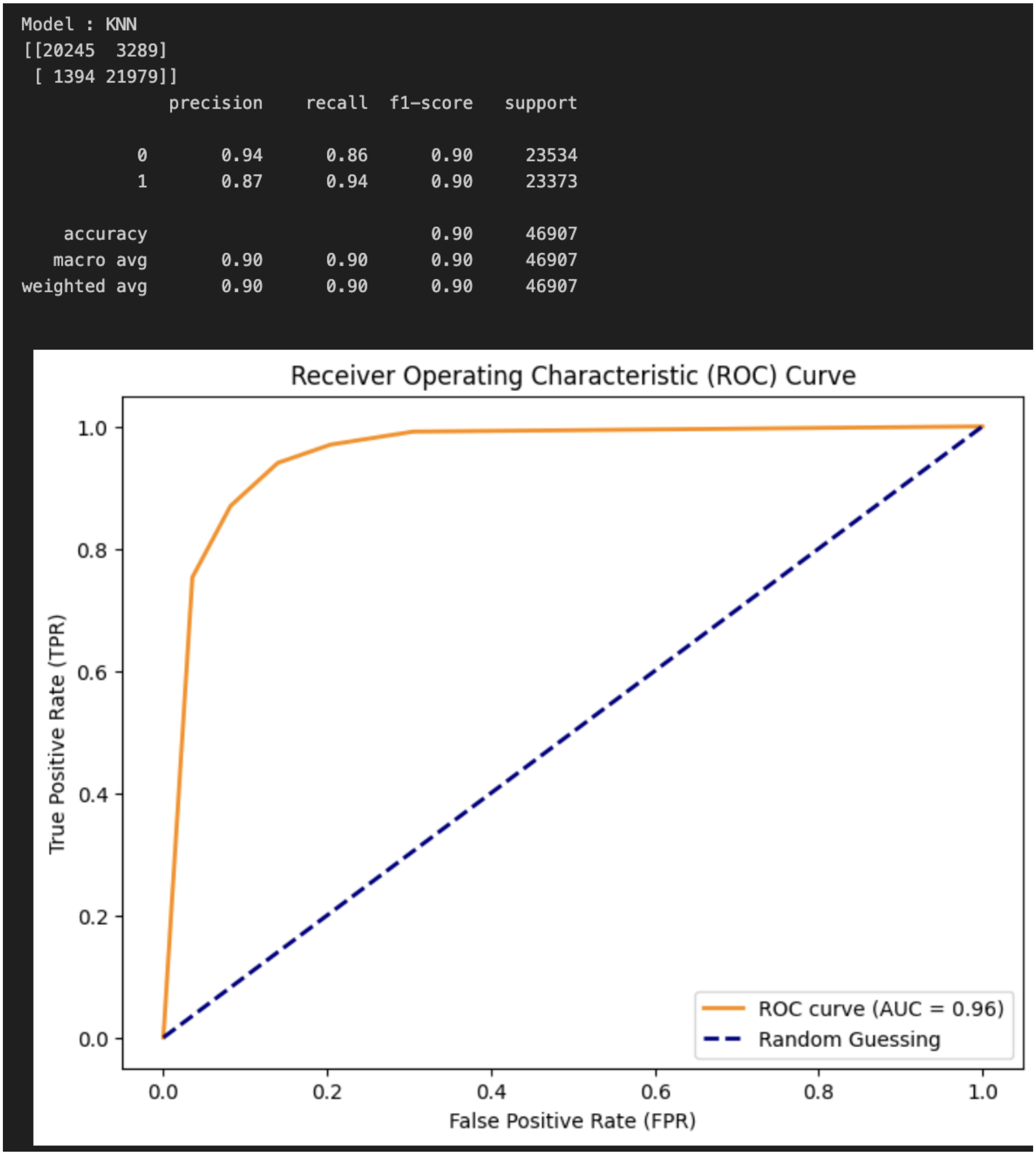


```
Model : XGBoostClassifier
[[21896  1638]
 [ 1416 21957]]
              precision    recall  f1-score   support

           0       0.94      0.93      0.93     23534
           1       0.93      0.94      0.93     23373

    accuracy                           0.93     46907
   macro avg       0.93      0.93      0.93     46907
weighted avg       0.93      0.93      0.93     46907
```

KNN



```
Model : KNN
[[20245  3289]
 [ 1394 21979]]
              precision    recall  f1-score   support

           0       0.94      0.86      0.90     23534
           1       0.87      0.94      0.90     23373

    accuracy                           0.90     46907
   macro avg       0.90      0.90      0.90     46907
weighted avg       0.90      0.90      0.90     46907
```

Conclusion

Models like Random Forest, Decision Tree, XGBoost and Kth Nearest Neighbour give the best results for the dataset. Support Vector Machine took more time complexity to run around 20 - 30 mins as it is slower with large datasets.