AI-Driven Exploration and Prediction of Company Registration Trends with the Registrar of Companies (ROC)

NAME : HARSHINI N

REGISTER NUMBER : 61772131014

MODEL:

For the model development and evaluation, the **RANDOM FOREST CLASSIFIER** is used which is a Machine Learning model.

This model combines multiple decision trees to make predictions.

It contains a number of decision trees on various subset of the given dataset and takes the average to improve the predictive accuracy of the dataset.

This model is known for its robustness , ability to handle high dimensional data and resistance to over-fitting.

This model takes less training time when compared to other algorithms.

It predicts the output with high accuracy for larger datasets too.

It can perform both classification and regression.

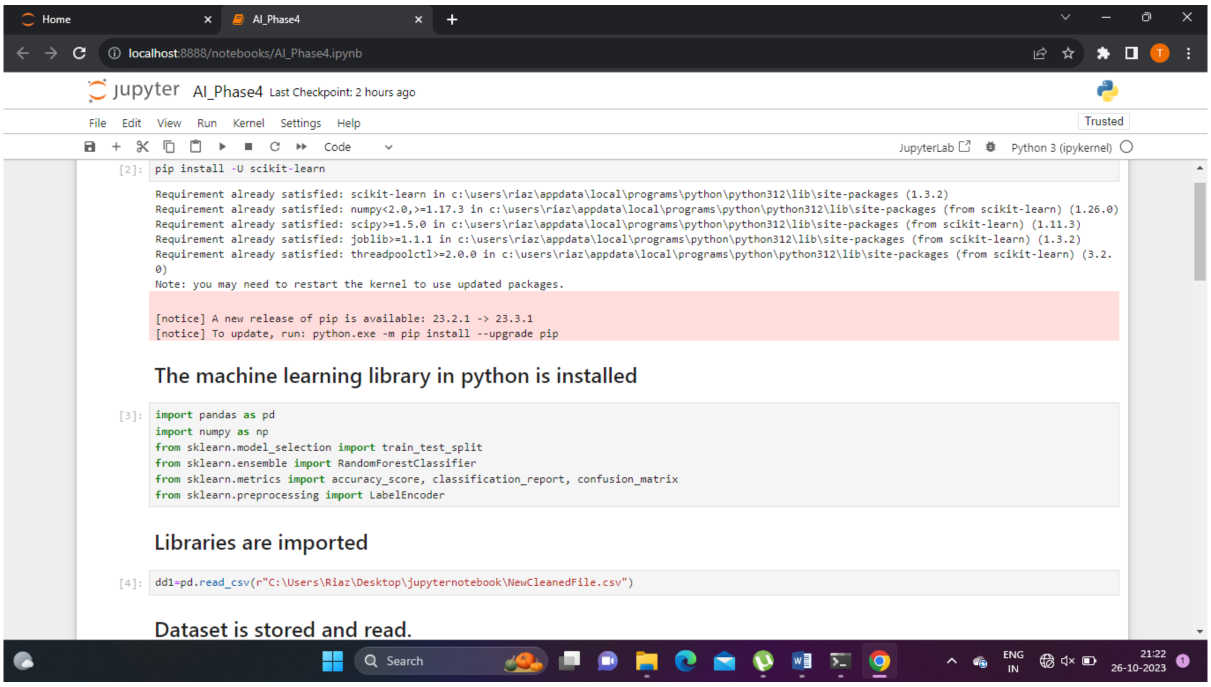
It can maintain accuracy though larger proportion of data is missing.

This model run in two phases:

- Create random forest by combining N decision trees.
- Make predictions for each tree created in the previous step.

First, the machine learning library in Python is installed and the required libraries are imported in the jupyter notebook.

The data set is read as a CSV file and stored in the jupyter notebook.



```
[2]: pip install -U scikit-learn

Requirement already satisfied: scikit-learn in c:\users\riaz\appdata\local\programs\python\python312\lib\site-packages (1.3.2)
Requirement already satisfied: numpy<2.0,>=1.17.3 in c:\users\riaz\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.26.0)
Requirement already satisfied: scipy>=1.5.0 in c:\users\riaz\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.11.3)
Requirement already satisfied: joblib>=1.1.1 in c:\users\riaz\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\riaz\appdata\local\programs\python\python312\lib\site-packages (from scikit-learn) (3.2.
0)
Note: you may need to restart the kernel to use updated packages.

[notice] A new release of pip is available: 23.2.1 -> 23.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

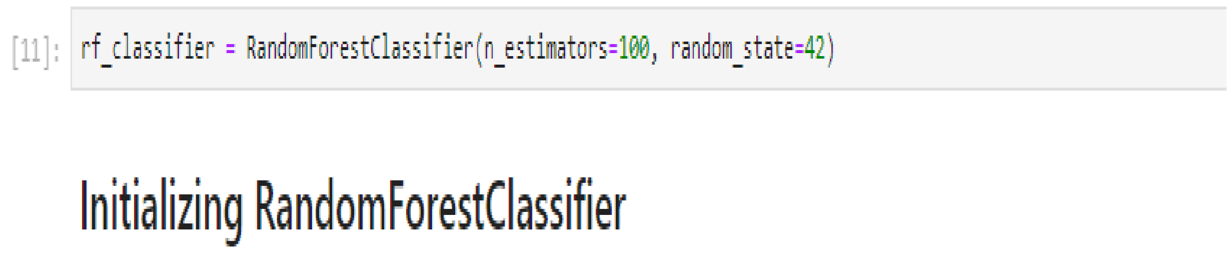
The machine learning library in python is installed

```
[3]: import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
     from sklearn.preprocessing import LabelEncoder
```

Libraries are imported

```
[4]: dd1=pd.read_csv(r"C:\Users\Riaz\Desktop\jupyternotebook\NewCleanedFile.csv")
```

Dataset is stored and read.



```
[11]: rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
```

Initializing RandomForestClassifier

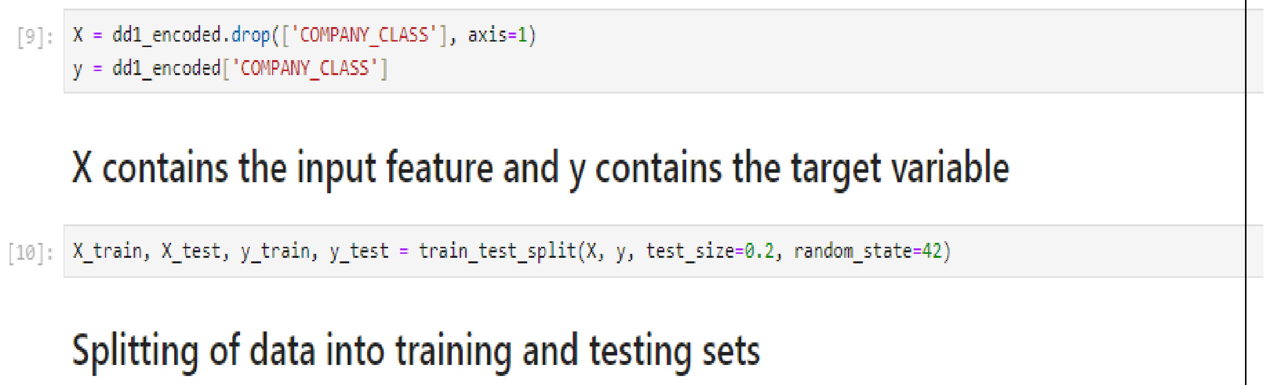
The RandomForestClassifier is initialized.

SPLITTING AND TRAINING:

The input feature and target variable is chosen from the dataset and stored in X and y respectively.

The data is split into two parts:

- Training set
- Testing set



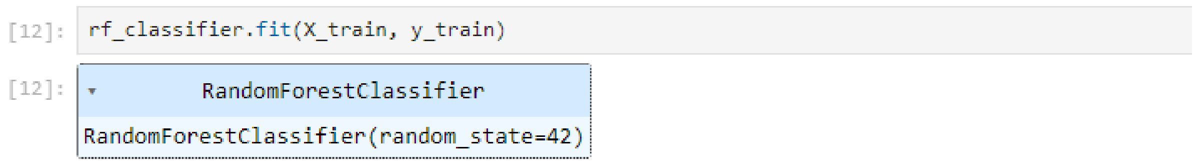
```
[9]:  X = dd1_encoded.drop(['COMPANY_CLASS'], axis=1)
      y = dd1_encoded['COMPANY_CLASS']
```

X contains the input feature and y contains the target variable

```
[10]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Splitting of data into training and testing sets

Training is done for the Random Forest Classifier using the training data.



```
[12]: rf_classifier.fit(X_train, y_train)

[12]: ▾      RandomForestClassifier
      RandomForestClassifier(random_state=42)
```

Trains the RandomForestClassifier

The training set is used to train the Random Forest model, while the testing set is used to evaluate the performance.

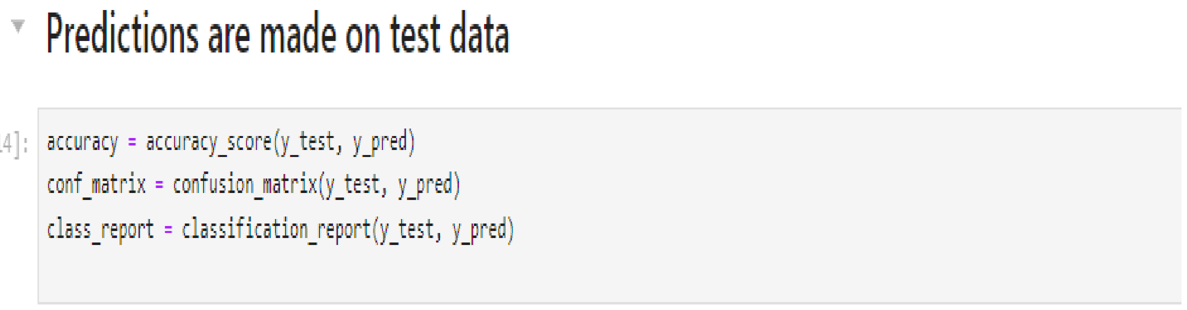
Once the model is trained, the testing set is used to evaluate the performance using various metrics like accuracy, precision, recall.

EVALUATION:

Evaluating the Random Forest model involves assessing its performance to understand how well it generalizes to unseen data.

PREDICTION:

Since the model is fitted into the training set, now we can predict the test result. For prediction, we create a new prediction vector ‘y_pred’.

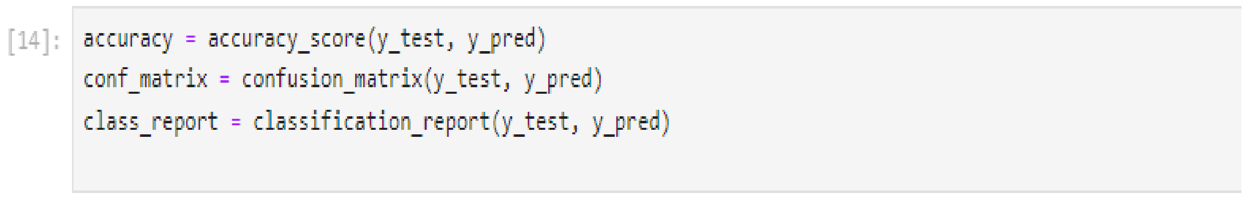


Predictions are made on test data

```
[14]: accuracy = accuracy_score(y_test, y_pred)
      conf_matrix = confusion_matrix(y_test, y_pred)
      class_report = classification_report(y_test, y_pred)
```

ACCURACY:

It measures the proportion of correctly classified instances in the test set.



```
[14]: accuracy = accuracy_score(y_test, y_pred)
      conf_matrix = confusion_matrix(y_test, y_pred)
      class_report = classification_report(y_test, y_pred)
```

Accuracy of the model's prediction

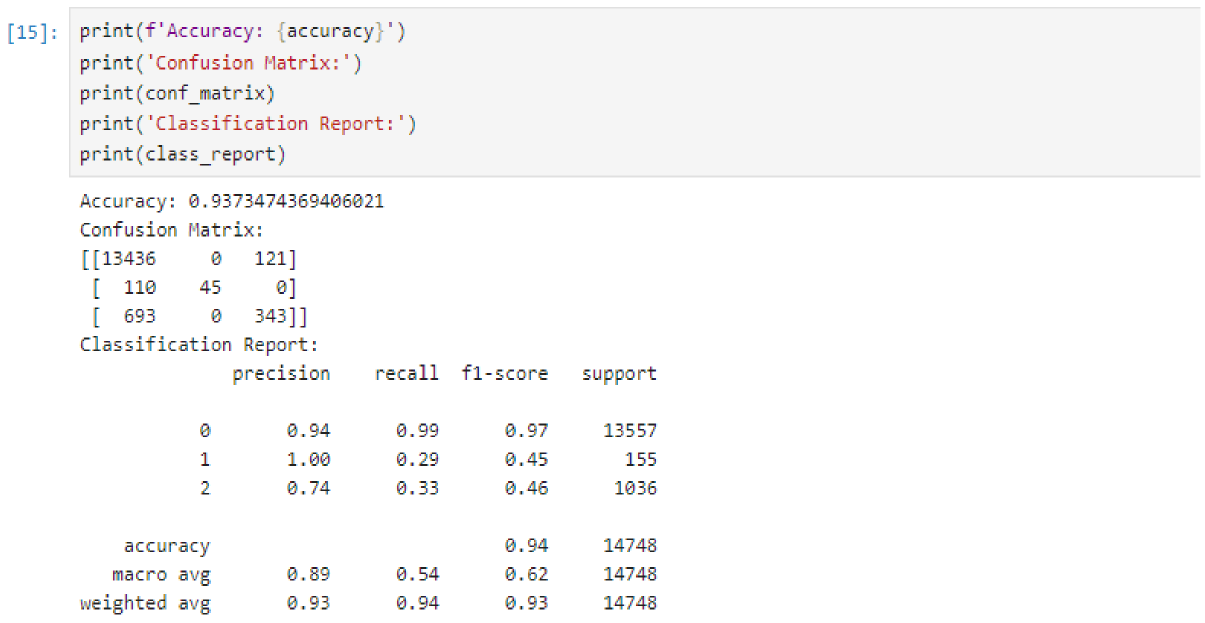
The accuracy can be assessed by using the above code.

In this code, the accuracy is calculated by using ‘accuracy_score’ by comparing the true target values ‘y_test’ with predicted values ‘y_pred’.

The result would be a decimal value between 0 and 1.

The confusion matrix and the classification report can also be assessed by using Random Forest Classifier Model.

Confusion matrix is used to determine the correct and incorrect predictions.



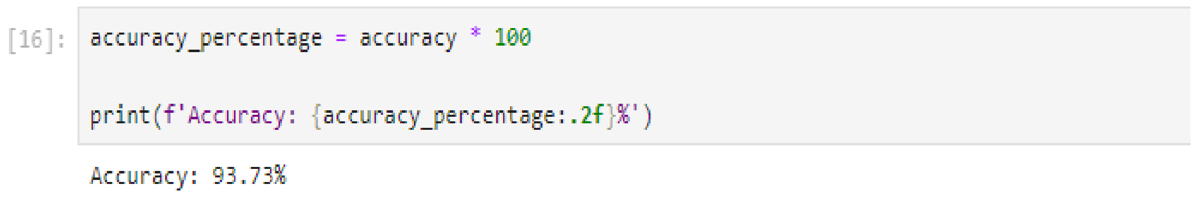
```
[15]: print(f'Accuracy: {accuracy}')
      print('Confusion Matrix:')
      print(conf_matrix)
      print('Classification Report:')
      print(class_report)
```

```
Accuracy: 0.9373474369406021
Confusion Matrix:
[[13436     0   121]
 [  110    45     0]
 [  693     0   343]]
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97     13557
           1       1.00      0.29      0.45       155
           2       0.74      0.33      0.46      1036

    accuracy                           0.94     14748
   macro avg       0.89      0.54      0.62     14748
weighted avg       0.93      0.94      0.93     14748
```

Prints the accuracy,confusion matrix,classification report



```
[16]: accuracy_percentage = accuracy * 100

      print(f'Accuracy: {accuracy_percentage:.2f}%')
```

```
Accuracy: 93.73%
```

Calculates accuracy in percentage and prints it

The accuracy is printed in percentage.

Accuracy achieved is 93.73%.