

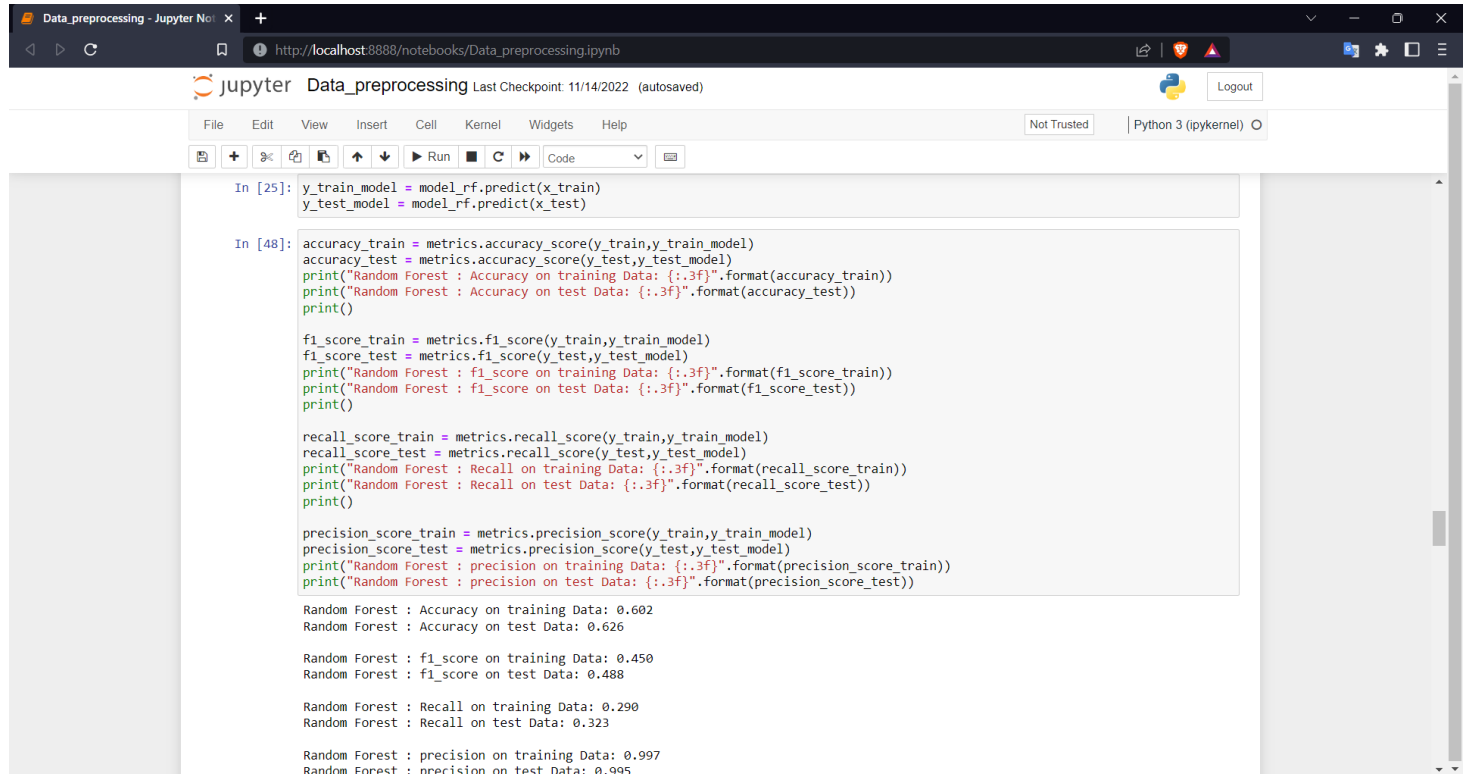
Project Development Phase
Performance Testing
Metrics

Team ID	PNT2022TMID49605
Project Name	Web phishing detection
Maximum Marks	10 Marks

Model Performance Testing:

S.No.	Parameter	Values
1.	Model Summary	As Phishing attacks are increasing these days, the need for detecting phishing website is necessary. The Random Forest algorithm is used for building the model. Thus, proposed model successfully detects the phishing and legitimate websites using machine learning algorithm.
2.	Accuracy	Training accuracy - 95% - 98% Validation accuracy - 96%

Performance:



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [25]: y_train_model = model_rf.predict(x_train)
         y_test_model = model_rf.predict(x_test)

In [48]: accuracy_train = metrics.accuracy_score(y_train,y_train_model)
         accuracy_test = metrics.accuracy_score(y_test,y_test_model)
         print("Random Forest : Accuracy on training Data: {:.3f}".format(accuracy_train))
         print("Random Forest : Accuracy on test Data: {:.3f}".format(accuracy_test))
         print()

         f1_score_train = metrics.f1_score(y_train,y_train_model)
         f1_score_test = metrics.f1_score(y_test,y_test_model)
         print("Random Forest : f1_score on training Data: {:.3f}".format(f1_score_train))
         print("Random Forest : f1_score on test Data: {:.3f}".format(f1_score_test))
         print()

         recall_score_train = metrics.recall_score(y_train,y_train_model)
         recall_score_test = metrics.recall_score(y_test,y_test_model)
         print("Random Forest : Recall on training Data: {:.3f}".format(recall_score_train))
         print("Random Forest : Recall on test Data: {:.3f}".format(recall_score_test))
         print()

         precision_score_train = metrics.precision_score(y_train,y_train_model)
         precision_score_test = metrics.precision_score(y_test,y_test_model)
         print("Random Forest : precision on training Data: {:.3f}".format(precision_score_train))
         print("Random Forest : precision on test Data: {:.3f}".format(precision_score_test))

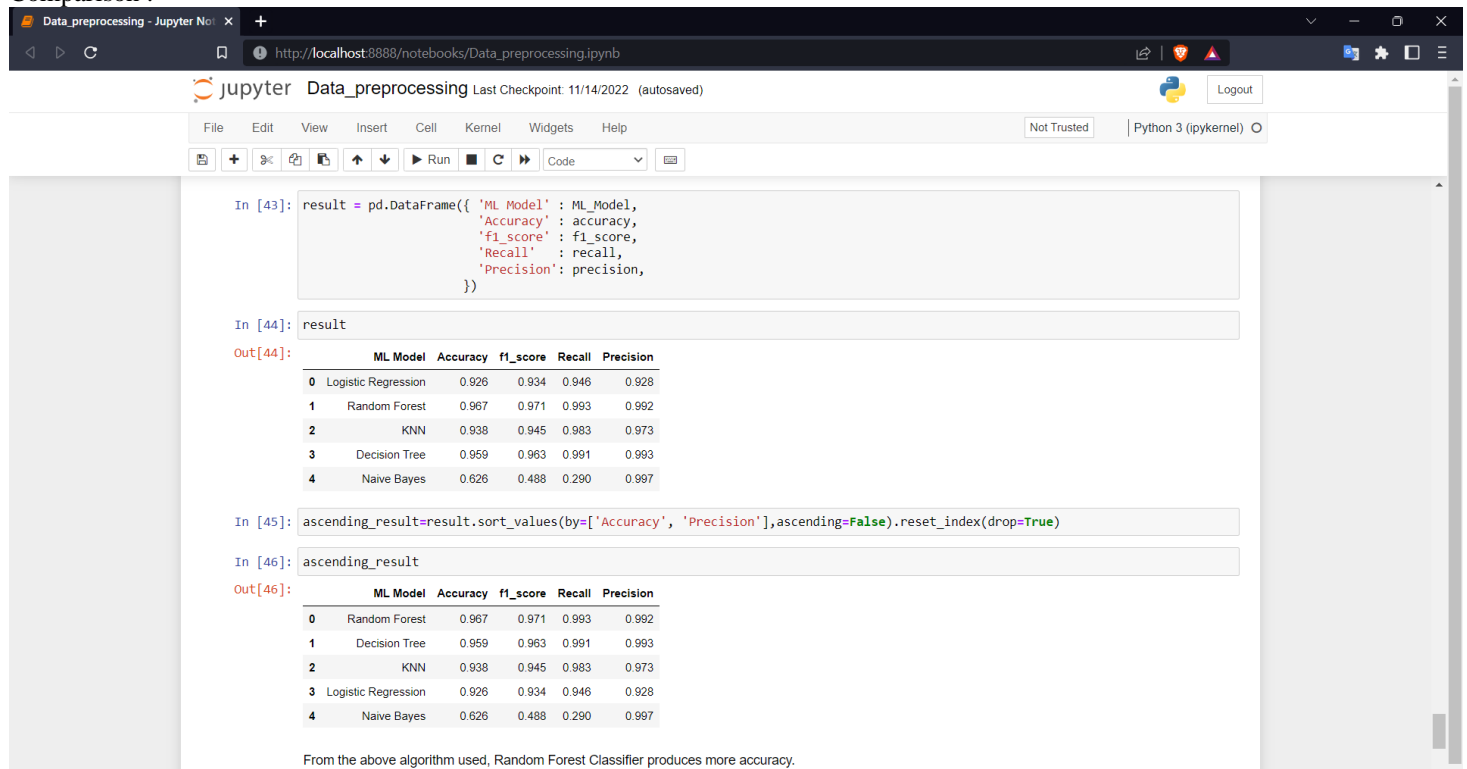
Random Forest : Accuracy on training Data: 0.602
Random Forest : Accuracy on test Data: 0.626

Random Forest : f1_score on training Data: 0.450
Random Forest : f1_score on test Data: 0.488

Random Forest : Recall on training Data: 0.290
Random Forest : Recall on test Data: 0.323

Random Forest : precision on training Data: 0.997
Random Forest : precision on test Data: 0.995
```

Comparison :



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [43]: result = pd.DataFrame({ 'ML Model' : ML_Model,
                                'Accuracy' : accuracy,
                                'f1_score' : f1_score,
                                'Recall' : recall,
                                'Precision': precision,
                                })

In [44]: result

Out[44]:
```

	ML Model	Accuracy	f1_score	Recall	Precision
0	Logistic Regression	0.926	0.934	0.946	0.928
1	Random Forest	0.967	0.971	0.993	0.992
2	KNN	0.938	0.945	0.983	0.973
3	Decision Tree	0.959	0.963	0.991	0.993
4	Naive Bayes	0.626	0.488	0.290	0.997

```
In [45]: ascending_result=result.sort_values(by=['Accuracy', 'Precision'],ascending=False).reset_index(drop=True)

In [46]: ascending_result

Out[46]:
```

	ML Model	Accuracy	f1_score	Recall	Precision
0	Random Forest	0.967	0.971	0.993	0.992
1	Decision Tree	0.959	0.963	0.991	0.993
2	KNN	0.938	0.945	0.983	0.973
3	Logistic Regression	0.926	0.934	0.946	0.928
4	Naive Bayes	0.626	0.488	0.290	0.997

From the above algorithm used, Random Forest Classifier produces more accuracy.