Project Development Phase
Model Performance Test

Date	10 November 2022
Team ID	PNT2022TMID29648
Project Name	Project – WEB PHISHING DETECTION
Maximum Marks	10 Marks

Model Performance Testing:

1.Metrices

Classification Model

Confusion Matrix, Accuracy Score & Classification Report

Logistic Regression

```
[24] from sklearn.linear_model import LogisticRegression
     log_reg = LogisticRegression()
     log_reg.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
pred3= log_reg.predict(x_test)
s3=accuracy_score(y_test,pred3)*100
results('LR',s3)
print("Accuracy score :",accuracy_score(y_test,pred3)*100)
print("Precision score :",precision_score(y_test,pred3,average="macro")*100)

print("Confustion matrix :",confusion_matrix(y_test,pred3))

from sklearn.metrics import classification_report
p=log_reg.predict(X)
print(classification_report(Y,p))
```

```
Accuracy score : 92.67299864314789
Precision score : 92.56074521739423
Confustion matrix : [[ 856   86]
 [  76 1193]]
              precision    recall  f1-score   support

          -1       0.93      0.91      0.92      4897
           1       0.93      0.95      0.94      6157

    accuracy                           0.93     11054
   macro avg       0.93      0.93      0.93     11054
weighted avg       0.93      0.93      0.93     11054
```

2.Tune The Model

Hyper Parameter Tuning

Hyper parameter tuning

```
from sklearn.linear_model import LogisticRegression
log_reg = LogisticRegression()
log_reg.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
[43] log_reg.get_params()
```

```
{'C': 1.0,
 'class_weight': None,
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 100,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': None,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

```
[46] parameters = {
        "l1_ratio":[4,5,6,7,8,9],
        "random_state":[0,1,10,42,100],
        "n_jobs":[10,20,30,40,50]
     }
```

```
[51] from sklearn.model_selection import GridSearchCV
     grid = GridSearchCV(log_reg,parameters,cv = 5,verbose=2)
```

```
grid.fit(X,Y)
```

```
[CV] END .............l1_ratio=9, n_jobs=30, random_state=100; total time=   0.4s
[CV] END .............l1_ratio=9, n_jobs=30, random_state=100; total time=   0.4s
[CV] END .............l1_ratio=9, n_jobs=30, random_state=100; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=0; total time=   1.6s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=0; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=0; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=0; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=0; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=1; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=1; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=1; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=1; total time=   1.1s
[CV] END .............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END .............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END .............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END .............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END .............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
```

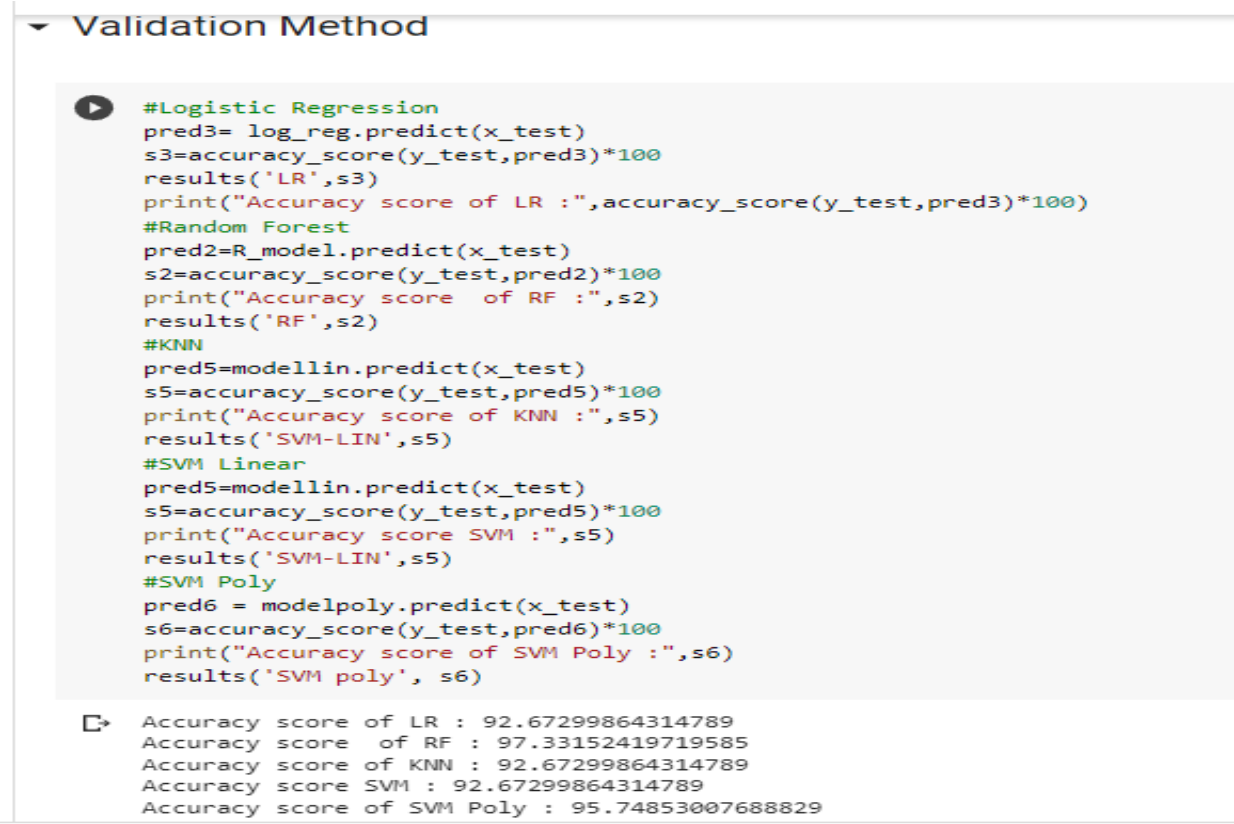
```
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=1; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=1; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=10; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=40, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=0; total time=   1.8s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=0; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=0; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=0; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=0; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
```

```
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=1; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=10; total time=   0.4s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=42; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=100; total time=   1.1s
[CV] END ..............l1_ratio=9, n_jobs=50, random_state=100; total time=   1.1s
GridSearchCV(cv=5, estimator=LogisticRegression(),
             param_grid={'l1_ratio': [4, 5, 6, 7, 8, 9],
                         'n_jobs': [10, 20, 30, 40, 50],
                         'random_state': [0, 1, 10, 42, 100]},
             verbose=2)
```

```
[54] print(grid.best_params_)
     print(grid.best_estimator_)
     print(grid.best_score_)
```

```
{'l1_ratio': 4, 'n_jobs': 10, 'random_state': 0}
LogisticRegression(l1_ratio=4, n_jobs=10, random_state=0)
0.9225618104459195
```

Validation Method



Validation Method

```
#Logistic Regression
pred3= log_reg.predict(x_test)
s3=accuracy_score(y_test,pred3)*100
results('LR',s3)
print("Accuracy score of LR :",accuracy_score(y_test,pred3)*100)
#Random Forest
pred2=R_model.predict(x_test)
s2=accuracy_score(y_test,pred2)*100
print("Accuracy score  of RF :",s2)
results('RF',s2)
#KNN
pred5=modellin.predict(x_test)
s5=accuracy_score(y_test,pred5)*100
print("Accuracy score of KNN :",s5)
results('SVM-LIN',s5)
#SVM Linear
pred5=modellin.predict(x_test)
s5=accuracy_score(y_test,pred5)*100
print("Accuracy score SVM :",s5)
results('SVM-LIN',s5)
#SVM Poly
pred6 = modelpoly.predict(x_test)
s6=accuracy_score(y_test,pred6)*100
print("Accuracy score of SVM Poly :",s6)
results('SVM poly', s6)
```

```
Accuracy score of LR : 92.67299864314789
Accuracy score  of RF : 97.33152419719585
Accuracy score of KNN : 92.67299864314789
Accuracy score SVM : 92.67299864314789
Accuracy score of SVM Poly : 95.74853007688829
```