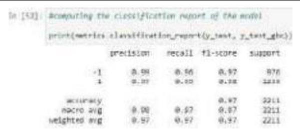
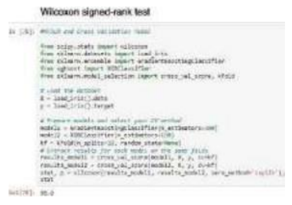
Project Development Phase
Model Performance Test

DATE	17-11-2002
Team ID	PNT2022TMID28852
TEAM MEMBERS	1. SWARNA RAJINI 2. B.ANANTHA KUMAR 3. CH.GIREESH BABU 4. D.SANTHOSH
PROJECT NAME	WEB PHISHING DETECTION
Marks	4 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Metrics	Classification Model: Gradient Boosting Classification Accuray Score- 97.4%	
2.	Tune the Model	Hyperparameter Tuning - 97% Validation Method – KFOLD & Cross Validation Method	

```
In [53]: #computing the classification report of the model

         print(metrics.classification_report(y_test, y_test_gbc))

                      precision    recall  f1-score   support

                  -1       0.99      0.96      0.97       976
                   1       [illegible]  [illegible]  [illegible]  [illegible]

            accuracy                           0.97      2211
           macro avg       0.98      0.97      0.97      2211
        weighted avg       0.97      0.97      0.97      2211
```

Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation Model

         from scipy.stats import wilcoxon
         from sklearn.datasets import load_iris
         from sklearn.ensemble import GradientBoostingClassifier
         from xgboost import XGBClassifier
         from sklearn.model_selection import cross_val_score, KFold

         # Load the dataset
         X = load_iris().data
         y = load_iris().target

         # Prepare models and select your CV method
         model1 = GradientBoostingClassifier(n_estimators=100)
         model2 = XGBClassifier(n_estimators=100)
         kf = KFold(n_splits=20, random_state=None)
         # Extract results for each model on the same folds
         results_model1 = cross_val_score(model1, X, y, cv=kf)
         results_model2 = cross_val_score(model2, X, y, cv=kf)
         stat, p = wilcoxon(results_model1, results_model2, zero_method='zsplit');
         stat

Out[78]: 95.0
```

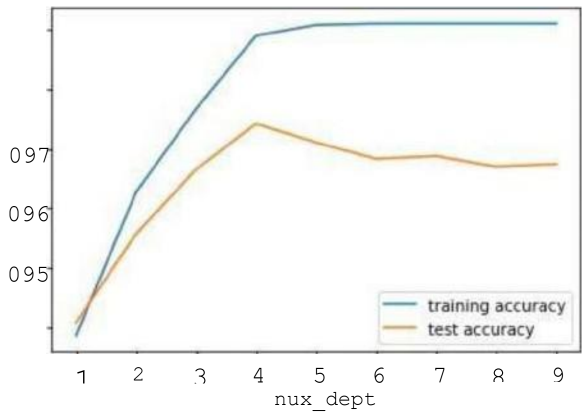
1. **METRICS:**

CLASSIFICATION REPORT:

```
  In [52] : #computing the classification
        report of the model
        print(metrics.classification_report(y
        _test, y_test_gbc))
                  precisionrecall f1- support
                                  score
```

```
                -1     e. 99   e. 96   e .97      976
                 1      e.97    e.99    0.98     1235

          accuracy                       e.97     2211
         macro avg     e. 98    e."     0.97     2211
    weighted           e. 97   e. 97    0.97     2211
    avg
```

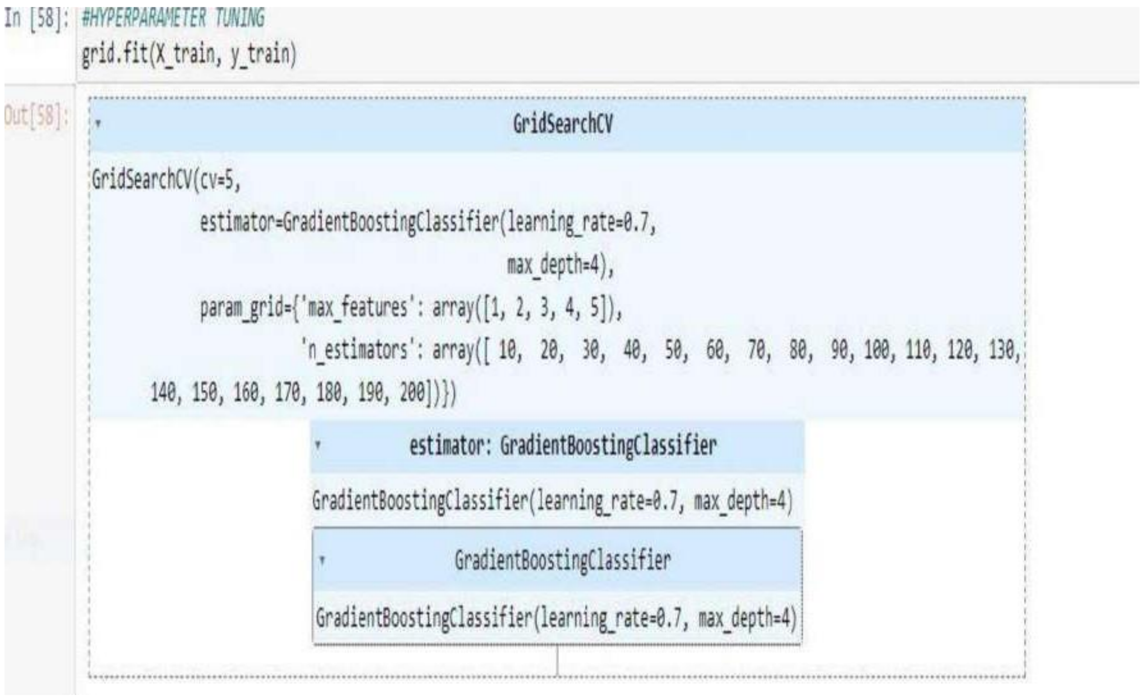
PERFORMANCE :



```
Out 33 J :   ML Model Accuracy fl _score Recall Precision
         ________________________________________________________________

          0 Gradient Boosting       0.974   0.977  0.994  0.986
               Classifier
           CatBoost Classifier      0.972   0.975  0.994  0.989

          2         Random Forest    0.969   0.972  0.992  0.991

          3       Support Vector    0.964   0.968  0.980  0.965
          Machine
          4         Decision Tree    0.958   0.962  0.991  0.993
          5  K-Nearest Neighbors     0.956   0.961  0.991  0.989

          6  Logistic Regression    0.934   0.941  0.943  0.927
                                            0454
                     Naive Bayes     0.605          0292   0.997
                      Classifier
          8   XGBoost Classifier    0.548   0.548  0.993  0.984
```

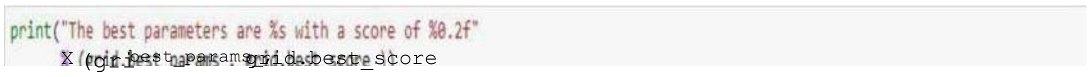
9	Multi-layer Perceptron	0.543	0.543	0.989	0983

2. TUNE THE MODEL - HYPERPARAMETER TUNING



```
In [58]: #HYPERPARAMETER TUNING
         grid.fit(X_train, y_train)

Out[58]:
GridSearchCV

GridSearchCV(cv=5,
             estimator=GradientBoostingClassifier(learning_rate=0.7,
                                                  max_depth=4),
             param_grid={'max_features': array([1, 2, 3, 4, 5]),
                         'n_estimators': array([ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120, 130,
       140, 150, 160, 170, 180, 190, 200])})

estimator: GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)

GradientBoostingClassifier
GradientBoostingClassifier(learning_rate=0.7, max_depth=4)
```

```
In [59]:
```



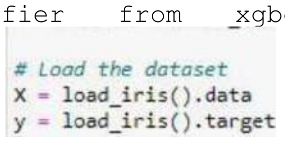
```
print("The best parameters are %s with a score of %0.2f"
      % (grid.best_params_, grid.best_score_))
```

```
The best parneters are {'max_features': 5, 'n_estimators': 20) with a score of 0.97
```

VALIDATION METHODS: KFOLD & Cross Folding

Wilcoxon signed-rank test

```
In [78]: #KFOLD and Cross Validation

          from scipy . stats import wilcoxon from
          sklearn. datasets import load_iris from
          sklearn        .ensemble        import
     GradientBoostingC1assifier   from   _xgboost
```



```
# Load the dataset
X = load_iris().data
y = load_iris().target
```

```
     import  X6BC1assifier  from
     sklearn .model_selection import cross_val_score,
     KF01d

     #    modell  Prepare models and select your CV method
     — =  mode12
          kf g
                    Extract results for each model on the same
                  folds   results_modell   =   X,   y,   cv=kf)
          stat,   results_mode12 — cross_va1_score(mode12, X,
     y,   stat    cv=kf) p = results_mode12, zsplit• );

outt78J:          9S.ø
```

5x2CV combined F test

```
In   [891:  from  mlxtend.  evaluate  import  combined
      ftest_5x2cv   from   sklearn.   tree   import
      DecisionTreeCIassifier, ExtraTreeCIassifier from
      sklearn.ensemble                         import
      GradientBoostingC1as5ifier   from   mlxtend.data
      import iris_data # Prepare data and c Ifs

      clfl
      GradientBoostingC1assifier(
      ) clf2 •
      DecisionTreeCIassifier()

      # CaLcuLate p-vaLue f, p
      cortined estimator2=c1f2,


      j print(
         f) print(
      •p-value: ' ,
      p)
      f-value: 1.727272727272733
      p-value: 0.284ø135734291782
```