Project Development Phase
Model Performance Test

Date	13 November 2022
Team ID	PNT2022TMID26146
Project Name	Project – Web Phishing Detection
Maximum Marks	10 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- 97.1%
			`#computing the classification report of the model`
			`print(metrics.classification_report(y_test, y_test_gbc))`
			precision recall f1-score support
			-1 0.98 0.95 0.97 956
			1 0.96 0.99 0.97 1255
			accuracy 0.97 2211
			macro avg 0.97 0.97 0.97 2211
			weighted avg 0.97 0.97 0.97 2211

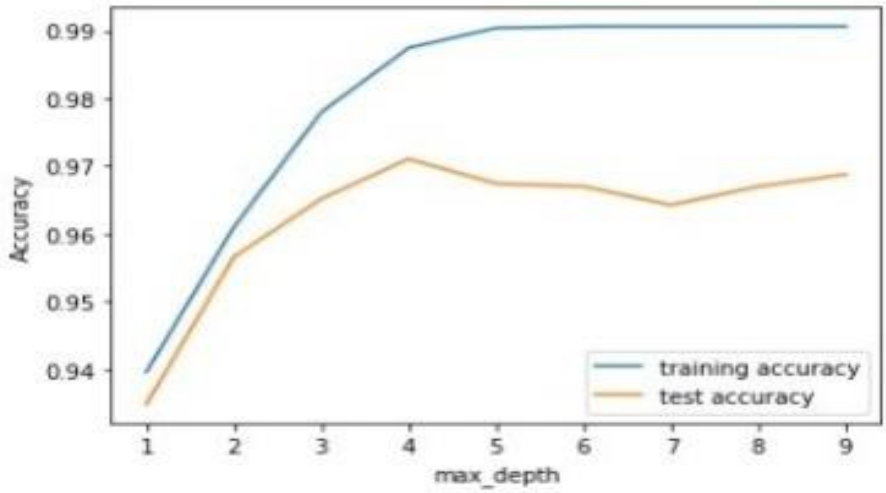
2.	Tune the Model	Hyperparameter Tuning - 97%
		Validation Method – KFOLD &
		Cross Validation Method
			Wilcoxon signed-rank test
			In [78]:
			`#KFOLD and Cross Validation Model`
			`from scipy.stats import wilcoxon`
			`from sklearn.datasets import load_iris`
			`from sklearn.ensemble import GradientBoostingClassifier`
			`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:

```
#computing the classification report of the model
print(metrics.classification_report(y_test, y_test_gbc))
```

	precision	recall	f1-score	support
-1	0.98	0.95	0.97	956
1	0.96	0.99	0.97	1255
accuracy			0.97	2211
macro avg	0.97	0.97	0.97	2211
weighted avg	0.97	0.97	0.97	2211

PERFORMANCE :



	ML Model	Accuracy	f1_score	Recall	Precision
0	Logistic Regression	0.924	0.933	0.947	0.927
1	K-Nearest Neighbors	0.953	0.959	0.990	0.989
2	Support Vector Machine	0.957	0.963	0.982	0.966
3	Decision Tree	0.958	0.963	0.992	0.991
4	Random Forest	0.965	0.970	0.995	0.987
5	Gradient Boosting Classifier	0.971	0.975	0.992	0.985

1. TUNE THE MODEL – HYPERPARAMETER TUNING

```
# fit the model
gbc.fit(X_train,y_train)
```

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

```
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)
```

VALIDATION METHODS: KFOLD & Cross Folding

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

5x2CV combined F test

In [89]:
```
from mlxtend.evaluate import combined_ftest_5x2cv
from sklearn.tree import DecisionTreeClassifier, ExtraTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from mlxtend.data import iris_data

# Prepare data and clfs
X, y = iris_data()
clf1 = GradientBoostingClassifier()
clf2 = DecisionTreeClassifier()

# Calculate p-value
f, p = combined_ftest_5x2cv(estimator1=clf1,
                            estimator2=clf2,
                            X=X, y=y,
                            random_seed=1)

print('f-value:', f)
print('p-value:', p)
```

```
f-value: 1.727272727272733
p-value: 0.2840135734291782
```