

## Project Development Phase Model Performance Test

Date	17th November 2022
Team ID	PNT2022TMID38399
Project Name	Project – Detecting Parkinson’s Disease using Machine Learning
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	<b>Classification Model:</b> Confusion Matrix, F1 Score, Accuracy Score & Classification Report	<p>The screenshot shows a Jupyter Notebook with the following code and output:</p> <pre> # Confusion matrix cm = confusion_matrix(y_test, y_pred) print(cm)  # F1 score f1 = f1_score(y_test, y_pred) print(f1)  # Classification report report = classification_report(y_test, y_pred) print(report) </pre> <p>The output displays the Confusion Matrix, F1 Score, and a detailed Classification Report for the model.</p>
2.	Tune the Model	Data mining - XGBoost Classifier	<p>The screenshot shows a Jupyter Notebook with the following code and output:</p> <pre> # XGBoost Classifier - Supervised Machine Learning xgb = xgb.XGBClassifier() xgb.fit(X_train, y_train)  # Assessing the model using metrics y_pred = xgb.predict(X_test) cm = confusion_matrix(y_test, y_pred) print(cm)  # Confusion matrix cm = confusion_matrix(y_test, y_pred) print(cm) </pre> <p>The output displays the results of the XGBoost Classifier training and evaluation, including the Confusion Matrix and performance metrics.</p>

## Metrics Parameter screensho

### ▼ XGBClassification - Supervised Machine Learning

```
✓ [31] Model_XG = XGBClassifier(random_state=0)
0s   Model_XG.fit(x_train,y_train)

XGBClassifier()
```

### ▼ Assessing the model using metrics

```
✓ [32] y_predict = Model_XG.predict(x_test)
0s   print(accuracy_score(y_test,y_predict)*100)

98.30508474576271
```

Hence by reducing the overfitting using XGBoost Classifier, we are getting accuracy\_score of **98.30%** for the model

### ▼ Confusion metrics

```
✓ [33] from sklearn.metrics import confusion_matrix
0s   ypre = Classification_model.predict(x_test)
     ypre = (ypre>0.5)
     confusion_matrix(y_test,ypre)

array([[18,  6],
       [ 6, 29]])
```

## Tune the model Parameter screenshot

```
[32] y_predict = Model_XG.predict(x_test)
     print(accuracy_score(y_test,y_predict)*100)
```

```
98.30508474576271
```

Hence by reducing the overfitting using XGBoost Classifier, we are getting accuracy\_score of **98.30%** for the model

### ▼ Confusion metrics

```
[33] from sklearn.metrics import confusion_matrix
     ypre = Classification_model.predict(x_test)
     ypre = (ypre>0.5)
     confusion_matrix(y_test,ypre)
```

```
array([[18,  6],
       [ 6, 29]])
```

### ▼ F1 score

```
[34] from sklearn.metrics import f1_score
     Variation_score = f1_score(y_test, Model_XG.predict(x_test), average='binary')
     print(Variation_score/0.01)
```

```
98.59154929577464
```

### ▼ Classification report

```
[35] from sklearn import metrics
     from sklearn.metrics import classification_report
     print("\n Classification report for Model  %s:\n%s\n" % (Model_XG, metrics.classification_report(y_test, y_pred)))
```

```
Classification report for Model  XGBClassifier():
precision    recall  f1-score   support
```

