Project Development Phase
Model Performance Test

Date	19 November 2022
Team ID	PNT2022TMID46390
Project Name	Project – Early Detection of Chronic Kidney 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	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuracy Score- & Classification Report -	See Below
2.	Tune the Model	Hyper-parameter Tuning - Validation Method -	See Below

1. Metrics

Model: Gradient Boost Classification

```
# GradientBoostingClassifier:
from sklearn.ensemble import GradientBoostingClassifier
GradientBoost = GradientBoostingClassifier()
GradientBoost = GradientBoost.fit(X_train,y_train)

# Predictions:
y_pred = GradientBoost.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy: 0.975
[[55  3]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        58
           1       0.95      1.00      0.98        62

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```

2. Tune the Model

Hyper parameter Tuning:

- The number of features is important and should be tuned in random forest classification.
- Initially all parameters in the data set are taken as independent values to arrive at the dependent decision of Chronic Kidney Disease or No Chronic Kidney Disease.
- But the result was not accurate so used only 10 more correlated values as independent values to arrive at the dependent decision of Chronic Kidney Disease or not.

Validation Method:

It involves **partitioning the training data set into subsets, where one subset is held out to test the performance of the model**. This data set is called the validation data set.

Cross validation is to use different models and identify the best:

Random Forest Classifier Model performance values:

```
# Importing Performance Metrics:
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
# RandomForestClassifier:
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(X_train,y_train)

# Predictions:
y_pred = RandomForest.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy: 0.975
[[55  3]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        58
           1       0.95      1.00      0.98        62

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```

Hence we tested with Gradient Boost Classifier and Random Forest Classification wherein the accuracy of Random Forest classification is 95% compared with Gradient Boost Classifier.

<table>
<tr><th>Metric</th><th>Gradient Boost Classifier</th><th>Random Forest Classification</th></tr>
<tr><td>Accuracy</td><td>0.97</td><td>0.99</td></tr>
<tr><td>Other metrics</td><td>

```
# GradientBoostingClassifier:
from sklearn.ensemble import GradientBoostingClassifier
GradientBoost = GradientBoostingClassifier()
GradientBoost = GradientBoost.fit(X_train,y_train)

# Predictions:
y_pred = GradientBoost.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy: 0.975
[[55  3]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        58
           1       0.95      1.00      0.98        62

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```

</td><td>

```
# Importing Performance Metrics:
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
# RandomForestClassifier:
from sklearn.ensemble import RandomForestClassifier
RandomForest = RandomForestClassifier()
RandomForest = RandomForest.fit(X_train,y_train)

# Predictions:
y_pred = RandomForest.predict(X_test)

# Performance:
print('Accuracy:', accuracy_score(y_test,y_pred))
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
```

```
Accuracy: 0.975
[[55  3]
 [ 0 62]]
              precision    recall  f1-score   support

           0       1.00      0.95      0.97        58
           1       0.95      1.00      0.98        62

    accuracy                           0.97       120
   macro avg       0.98      0.97      0.97       120
weighted avg       0.98      0.97      0.97       120
```

</td></tr>
</table>

The above table shows that Random Forest Classification gives better results over Gradient Boost Classifier.