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: Logistic Regression Classification

In [114]:
```
# Classification report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        28
           1       1.00      1.00      1.00        52

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80
```

In [115]:
```
# Creating a confusion matrix for training set
y_train_pred=rf.predict(X_train)
cm=confusion_matrix(y_train,y_train_pred)
cm
```

Out[115]:
```
array([[121,   1],
       [  0, 198]], dtype=int64)
```

In [116]:
```
# Accuracy score
score=round(accuracy_score(y_train,y_train_pred),3)
print("Accuracy on training set: {}".format(score))
```

```
Accuracy on training set: 0.997
```

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 8 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.

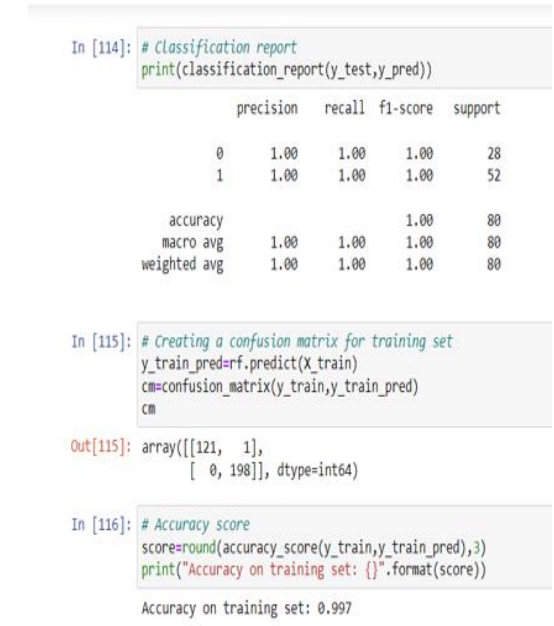
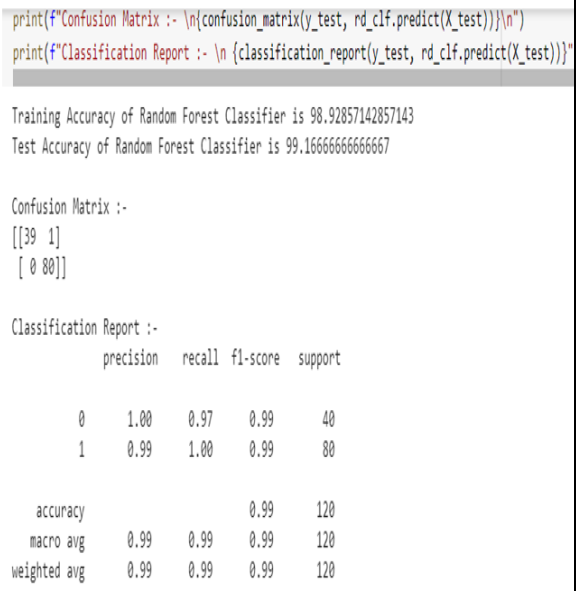
Cross validation is to use different models and identify the best:

Random Forest Classifier Model performance values:

```
print(f"Confusion Matrix :- \n{confusion_matrix(y_test, rd_clf.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, rd_clf.predict(X_test))}")
```

```
Training Accuracy of Random Forest Classifier is 98.92857142857143
Test Accuracy of Random Forest Classifier is 99.16666666666667

Confusion Matrix :-
[[39  1]
 [ 0 80]]

Classification Report :-
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        40
           1       0.99      1.00      0.99        80

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

Hence we tested with Logistic regression and Random Forest Classification wherein the accuracy of Random Forest classification is 95% compared with Logistic Regression.

Metric	Logistic Regression	Random Forest Classification
Accuracy	0.97	0.99
Other metrics		

```
In [114]: # Classification report
          print(classification_report(y_test,y_pred))

                        precision    recall  f1-score   support

                     0       1.00      1.00      1.00        28
                     1       1.00      1.00      1.00        52

              accuracy                           1.00        80
             macro avg       1.00      1.00      1.00        80
          weighted avg       1.00      1.00      1.00        80

In [115]: # Creating a confusion matrix for training set
          y_train_pred=rf.predict(X_train)
          cm=confusion_matrix(y_train,y_train_pred)
          cm

Out[115]: array([[121,   1],
                 [  0, 198]], dtype=int64)

In [116]: # Accuracy score
          score=round(accuracy_score(y_train,y_train_pred),3)
          print("Accuracy on training set: {}".format(score))

          Accuracy on training set: 0.997
```

```
print(f"Confusion Matrix :- \n{confusion_matrix(y_test, rd_clf.predict(X_test))}\n")
print(f"Classification Report :- \n {classification_report(y_test, rd_clf.predict(X_test))}"

Training Accuracy of Random Forest Classifier is 98.92857142857143
Test Accuracy of Random Forest Classifier is 99.16666666666667

Confusion Matrix :-
[[39  1]
 [ 0 80]]

Classification Report :-
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        40
           1       0.99      1.00      0.99        80

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

The above table shows that Random Forest Classification gives better results over Logistic Regression.