Project Development Phase
Model Performance Test

Date	17 November 2022
Team ID	PNT2022TMID49652
Project Name	Project – Early Detection of Chronic Kidney Disease using Machine Learning
Maximum Marks	10 Marks

Model Performance Testing:

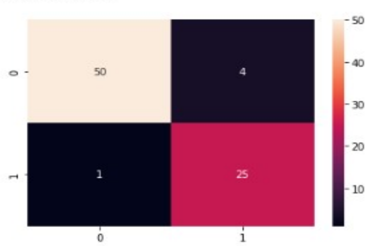
Project team shall fill the following information in model performance testing template.

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

<u>RANDOM FOREST CLASSIFICATION:</u>

In [88]:
```
#RandomForestClassifier
#displaying accuracy score of testing data
accuracy_score(y_test,y_pred) #higher accuracy
```
Out[88]: 0.9375

In [89]:
```
#confusion matrix display
sns.heatmap(confusion_matrix(y_test,y_pred),annot=True)
```
Out[89]: <AxesSubplot:>



In [90]:
```
#classification report displaying  the other performance metrics

print(classification_report(y_test,y_pred));
```
```
              precision    recall  f1-score   support

           0       0.98      0.93      0.95        54
           1       0.86      0.96      0.91        26

    accuracy                           0.94        80
   macro avg       0.92      0.94      0.93        80
weighted avg       0.94      0.94      0.94        80
```

REGRESSION MODEL FOR RANDOM FOREST:

```
In [147]: #Regression model

In [138]: import numpy as np
          import sklearn.metrics as metrics
          import matplotlib.pyplot as plt

          Y = np.array([y_test])
          Yhat = np.array([y_pred])
          X = list(range(len(Y)))

In [142]: d = Y-Yhat
          mse_f = np.mean(d**2)
          mae_f = np.mean(abs(d))
          rmse_f = np.sqrt(mse_f)
          r2_f = 1-(sum(d**2)/sum((Y-np.mean(Y))**2))

          print("Results by manual alculations")
          print("MAE:",mae_f)
          print("MSE:",mse_f)
          print("RMSE",rmse_f)
          print("R-Squared:",r2_f)

          Results by manual alculations
          MAE: 0.451875
          MSE: 0.451875
          RMSE 0.6722164829874376
          R-Squared: [[ 1.          1.          1.         ...  1.         -8.46745562
             1.        ]
           [ 1.          1.          1.         ...  1.         -8.46745562
             1.        ]
           [ 1.          1.          1.         ...  1.         -8.46745562
             1.        ]
           ...
           [ 1.          1.          1.         ...  1.         -8.46745562
             1.        ]
           [-1.19478738 -1.19478738 -1.19478738 ... -1.19478738  1.
            -1.19478738]
           [ 1.          1.          1.         ...  1.         -8.46745562
             1.        ]]
```

TUNING USING CROSS VALIDATION FOR RANDOM FOREST:

```
In [ ]: #3Tunning using cross validation

In [201]: from sklearn.model_selection import train_test_split
          # split the data with 50% in each set
          x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=2)

In [205]: model.fit(x_train, y_train)
          y_model = model.predict(x_test)

In [206]: accuracy_score(y_test, y_model)

Out[206]: 0.75

In [146]: tuned_parameters = [{'n_estimators':[7,8,9,10,11,12,13,14,15,16],'max_depth':[2,3,4,5,6,None],
                               'class_weight':[None,{0: 0.33,1:0.67},'balanced'],'random_state':[42]}]
          clf = GridSearchCV(RandomForestClassifier(), tuned_parameters, cv=10,scoring='f1')
          clf.fit(x_train, y_train)

Out[146]: GridSearchCV(cv=10, estimator=RandomForestClassifier(),
                       param_grid=[{'class_weight': [None, {0: 0.33, 1: 0.67},
                                                     'balanced'],
                                    'max_depth': [2, 3, 4, 5, 6, None],
                                    'n_estimators': [7, 8, 9, 10, 11, 12, 13, 14, 15, 16],
                                    'random_state': [42]}],
                       scoring='f1')

In [149]: print("Detailed classification report:")
          y_true, lr_pred = y_test, clf.predict(x_test)
          print(classification_report(y_true, lr_pred))

          confusion = confusion_matrix(y_test, lr_pred)
          print('Confusion Matrix:')
          print(confusion)

          Detailed classification report:
                        precision    recall  f1-score   support

                     0       1.00      0.91      0.95        54
                     1       0.84      1.00      0.91        26

              accuracy                           0.94        80
             macro avg       0.92      0.95      0.93        80
          weighted avg       0.95      0.94      0.94        80

          Confusion Matrix:
          [[49  5]
           [ 0 26]]
```

LOGISTIC REGRESSION:

```
In [95]: LogisticRegression()
Out[95]: LogisticRegression()

In [96]: y_pred1=model1.predict(x_test)

In [97]: y_pred1
Out[97]: array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
                0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
                0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
                0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0])

In [98]: #Model Evaluation

In [99]: accuracy_score(y_test,y_pred1)
Out[99]: 0.9125

In [100]: sns.heatmap(confusion_matrix(y_test,y_pred1),annot=True)
Out[100]: <AxesSubplot:>
```

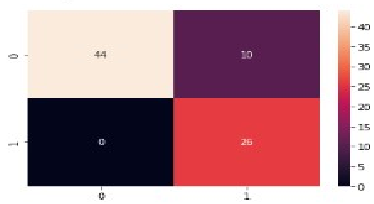

```
In [101]: print(classification_report(y_test,y_pred1))

              precision    recall  f1-score   support

           0       1.00      0.87      0.93        54
           1       0.79      1.00      0.88        26

    accuracy                           0.91        80
   macro avg       0.89      0.94      0.91        80
weighted avg       0.93      0.91      0.91        80
```

REGRESSION MODEL FOR LOGISTIC REGRESSION:

```
In [247]: import numpy as np
          import sklearn.metrics as metrics
          import matplotlib.pyplot as plt

          Y = np.array([(y_test)])
          Yhat = np.array([y_pred1])
          X = list(range(len(Y)))

In [248]: d = Y-Yhat
          mse_f = np.mean(d**2)
          mae_f = np.mean(abs(d))
          rmse_f = np.sqrt(mse_f)
          r2_f = 1-(sum(d**2)/sum((Y-np.mean(Y))**2))

          print("Results by manual alculations")
          print("MAE:",mae_f)
          print("MSE:",mse_f)
          print("RMSE",rmse_f)
          print("R-Squared:",r2_f)

          Results by manual alculations
          MAE: 0.469375
          MSE: 0.469375
          RMSE 0.6851094803022361
          R-Squared: [[ 1.          1.          1.          ... -8.46745562 -8.46745562
             1.        ]
           [ 1.          1.          1.          ... -8.46745562 -8.46745562
             1.        ]
           [ 1.          1.          1.          ... -8.46745562 -8.46745562
             1.        ]
           ...
           [ 1.          1.          1.          ... -8.46745562 -8.46745562
             1.        ]
           [-1.19478738 -1.19478738 -1.19478738 ...  1.          1.
            -1.19478738]
           [ 1.          1.          1.          ... -8.46745562 -8.46745562
             1.        ]]
```

GAUSSIAN NB:

```
         model2.fit(x_train,y_train)
Out[102]: GaussianNB()

In [103]: GaussianNB()
Out[103]: GaussianNB()

In [104]: y_pred3=model2.predict(x_test)
          y_pred3
Out[104]: array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
                 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0,
                 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0,
                 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0])

In [105]: accuracy_score(y_test,y_pred3)
Out[105]: 0.875

In [106]: sns.heatmap(confusion_matrix(y_test,y_pred3),annot=True)
Out[106]: <AxesSubplot:>
```



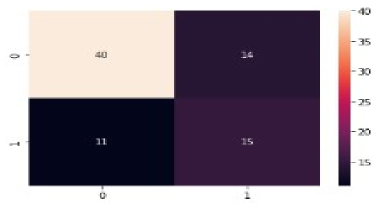

```
In [107]: print(classification_report(y_test,y_pred3))

              precision    recall  f1-score   support

           0       1.00      0.81      0.90        54
           1       0.72      1.00      0.84        26

    accuracy                           0.88        80
   macro avg       0.86      0.91      0.87        80
weighted avg       0.91      0.88      0.88        80
```

REGRESSION MODEL FOR GAUSSIAN NB:

```
In [255]: import numpy as np
          import sklearn.metrics as metrics
          import matplotlib.pyplot as plt

          Y = np.array([(y_test)])
          Yhat = np.array([y_pred3])
          X = list(range(len(Y)))

In [256]: d = Y-Yhat
          mse_f = np.mean(d**2)
          mae_f = np.mean(abs(d))
          rmse_f = np.sqrt(mse_f)
          r2_f = 1-(sum(d**2)/sum((Y-np.mean(Y))**2))

          print("Results by manual alculations")
          print("MAE:",mae_f)
          print("MSE:",mse_f)
          print("RMSE",rmse_f)
          print("R-Squared:",r2_f)

   Results by manual alculations
   MAE: 0.4825
   MSE: 0.4825
   RMSE 0.6946221994724903
   R-Squared: [[ 1.          1.          1.         ... -8.46745562 -8.46745562
      1.        ]
    [ 1.          1.          1.         ... -8.46745562 -8.46745562
      1.        ]
    [ 1.          1.          1.         ... -8.46745562 -8.46745562
      1.        ]
    ...
    [ 1.          1.          1.         ... -8.46745562 -8.46745562
      1.        ]
    [-1.19478738 -1.19478738 -1.19478738 ...  1.          1.
     -1.19478738]
    [ 1.          1.          1.         ... -8.46745562 -8.46745562
      1.        ]]
```

SUPPORT VECTOR MACHINE:

```
In [108]: #4th model based on Support vector machine

          from sklearn.svm import SVC
          model3=SVC()
          model3.fit(x_train,y_train)

Out[108]: SVC()

In [109]: y_pred4=model3.predict(x_test)
          y_pred4

Out[109]: array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0,
                 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0,
                 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
                 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0])

In [207]: accuracy_score(y_test,y_pred4) #Least accuracy

Out[207]: 0.6875

In [111]: sns.heatmap(confusion_matrix(y_test,y_pred4),annot=True)

Out[111]: <AxesSubplot:>
```




```
In [112]: print(classification_report(y_test,y_pred4))

                        precision    recall  f1-score   support

                   0       0.78      0.74      0.76        54
                   1       0.52      0.58      0.55        26

            accuracy                           0.69        80
           macro avg       0.65      0.66      0.65        80
        weighted avg       0.70      0.69      0.69        80
```

REGRESSION MODEL FOR SVM:

```
In [262]: import numpy as np
          import sklearn.metrics as metrics
          import matplotlib.pyplot as plt

          Y = np.array([(y_test)])
          Yhat = np.array([y_pred4])
          X = list(range(len(Y)))

In [263]: d = Y-Yhat
          mse_f = np.mean(d**2)
          mae_f = np.mean(abs(d))
          rmse_f = np.sqrt(mse_f)
          r2_f = 1-(sum(d**2)/sum((Y-np.mean(Y))**2))

          print("Results by manual alculations")
          print("MAE:",mae_f)
          print("MSE:",mse_f)
          print("RMSE",rmse_f)
          print("R-Squared:",r2_f)

    Results by manual alculations
    MAE: 0.451875
    MSE: 0.451875
    RMSE 0.6722164829874376
    R-Squared: [[ 1.          1.          1.          ... -8.46745562 -8.46745562
       1.        ]
     [ 1.          1.          1.          ... -8.46745562 -8.46745562
       1.        ]
     [ 1.          1.          1.          ... -8.46745562 -8.46745562
       1.        ]
     ...
     [ 1.          1.          1.          ... -8.46745562 -8.46745562
       1.        ]
     [-1.19478738 -1.19478738 -1.19478738 ...  1.          1.
      -1.19478738]
     [ 1.          1.          1.          ... -8.46745562 -8.46745562
       1.        ]]
```

DECISION TREE CLASSIFIER:

```
In [113]: #5th model is based on Decision tree

          from sklearn.tree import DecisionTreeClassifier
          model4=DecisionTreeClassifier(criterion='entropy')
          model4.fit(x_train,y_train)

Out[113]: DecisionTreeClassifier(criterion='entropy')

In [114]: y_pred5=model4.predict(x_test)
          y_pred5

Out[114]: array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
                 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
                 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0,
                 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0])

In [115]: accuracy_score(y_test,y_pred5) #has same high accuracy as Random Forest

Out[115]: 0.8875

In [116]: sns.heatmap(confusion_matrix(y_test,y_pred5),annot=True)

Out[116]: <AxesSubplot:>
```


```
In [117]: print(classification_report(y_test,y_pred5))

                        precision    recall  f1-score   support

                     0       0.91      0.93      0.92        54
                     1       0.84      0.81      0.82        26

              accuracy                           0.89        80
             macro avg       0.87      0.87      0.87        80
          weighted avg       0.89      0.89      0.89        80
```

REGRESSION MODEL FOR DECISION TREE:

```
          import matplotlib.pyplot as plt

          Y = np.array([(y_test)])
          Yhat = np.array([y_pred5])
          X = list(range(len(Y)))

In [270]: d = Y-Yhat
          mse_f = np.mean(d**2)
          mae_f = np.mean(abs(d))
          rmse_f = np.sqrt(mse_f)
          r2_f = 1-(sum(d**2)/sum((Y-np.mean(Y))**2))

          print("Results by manual alculations")
          print("MAE:",mae_f)
          print("MSE:",mse_f)
          print("RMSE",rmse_f)
          print("R-Squared:",r2_f)

            Results by manual alculations
            MAE: 0.43875
            MSE: 0.43875
            RMSE 0.6623820649745885
            R-Squared: [[ 1.          1.          1.         ... 1.         -8.46745562
               1.        ]
             [ 1.          1.          1.         ... 1.         -8.46745562
               1.        ]
             [ 1.          1.          1.         ... 1.         -8.46745562
               1.        ]
             ...
             [ 1.          1.          1.         ... 1.         -8.46745562
               1.        ]
             [-1.19478738 -1.19478738 -1.19478738 ... -1.19478738  1.
              -1.19478738]
             [ 1.          1.          1.         ... 1.         -8.46745562
               1.        ]]
```