Chronic Kidney Disease prediction

1. Problem identification
   - Since the dataset contains numerical hence Machine Learning would optimal.
   - The dataset has Input and output hence it is fall under Supervised learning.
   - The output col has ordinal data hence it is a Classification problem.

2. Basic info of the data
   - It has 400 row and 25 cols

```
In [3]: dataset.shape

Out[3]: (399, 25)
```

3. Using KNN model the following is the result ROC AUC(0.85)

```
In [23]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

C:\Users\YAS\Anaconda3\lib\site-packages\sklearn\base.py:442: UserWarning: X does not have valid feature names
rsClassifier was fitted with feature names
  "X does not have valid feature names, but"

Out[23]: 0.8542222222222222

In [31]: result = grid.cv_results_
         print ("The best parameter {} and its score {}".format(grid.best_params_,grid.best_score_))
         cm

The best parameter {'algorithm': 'auto', 'weights': 'distance'} and its score 0.7241187572628117

Out[31]: array([[42,  3],
                [25, 50]], dtype=int64)

In [27]: print(class_report)

              precision    recall  f1-score   support

           0       0.63      0.93      0.75        45
           1       0.94      0.67      0.78        75

    accuracy                           0.77       120
   macro avg       0.79      0.80      0.77       120
weighted avg       0.82      0.77      0.77       120
```

4. Using Decision tree model the following is the result ROC AUC(0.99)

```
In [24]: re = grid.cv_results_
         print ("Best param {} and best score {}".format(grid.best_params_,grid.best_score_))

         Best param {'criterion': 'gini', 'max_features': 'sqrt', 'splitter': 'random'} and best score 0.9850656883940017

In [25]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[25]: 0.9939024390243902

In [26]: cm

Out[26]: array([[51,  0],
                [ 1, 81]], dtype=int64)

In [27]: print(class_report)

                    precision    recall  f1-score   support

                 0       0.98      1.00      0.99        51
                 1       1.00      0.99      0.99        82

          accuracy                           0.99       133
         macro avg       0.99      0.99      0.99       133
      weighted avg       0.99      0.99      0.99       133
```

5. Using Logistics Regression model the following is the result ROC AUC(100)

```
In [15]: re = grid.cv_results_
         print ("Best param {} and best score {}".format(grid.best_params_,grid.best_score_))

         Best param {'multi_class': 'multinomial', 'penalty': 'l2', 'solver': 'newton-cg'} and best score 0.9812128263915746

In [16]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[16]: 1.0

In [17]: cm

Out[17]: array([[51,  0],
                [ 0, 82]], dtype=int64)

In [18]: print(class_report)

                    precision    recall  f1-score   support

                 0       1.00      1.00      1.00        51
                 1       1.00      1.00      1.00        82

          accuracy                           1.00       133
         macro avg       1.00      1.00      1.00       133
      weighted avg       1.00      1.00      1.00       133
```

6. Using Random Forest model the following is the result ROC AUC(0.99)

```
In [37]: re = grid.cv_results_
         print ("Best param {} and best score {}".format(grid.best_params_,grid.best_score_))

         Best param {'criterion': 'gini', 'max_features': 'log2'} and best score 0.9848006070227224

In [25]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[25]: 0.9997608799617408

In [26]: cm

Out[26]: array([[51,  0],
                [ 1, 81]], dtype=int64)

In [27]: print(class_report)

                      precision    recall  f1-score   support

                   0       0.98      1.00      0.99        51
                   1       1.00      0.99      0.99        82

            accuracy                           0.99       133
           macro avg       0.99      0.99      0.99       133
        weighted avg       0.99      0.99      0.99       133
```

7. Using SVC model the following is the result ROC AUC(0.99)

```
In [25]: re = grid.cv_results_
         print ("Best param {} and best score {}".format(grid.best_params_,grid.best_score_))

         Best param {'kernel': 'linear'} and best score 0.9473330858313987

In [26]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[26]: 0.9990435198469632

In [27]: cm

Out[27]: array([[50,  1],
                [ 1, 81]], dtype=int64)

In [28]: print(class_report)

                      precision    recall  f1-score   support

                   0       0.98      0.98      0.98        51
                   1       0.99      0.99      0.99        82

            accuracy                           0.98       133
           macro avg       0.98      0.98      0.98       133
        weighted avg       0.98      0.98      0.98       133
```

Conclusion & findings

➔ Logistics Regression works 100% good for this classification problem statement. Therefore saved/deployed this model to the business community.