1 Problem Statement :

Based on the Parameters like Bp, Rbc, Pc, Pcc . etc We need to classify if the respective person has Chronic Kidney Disease (CKD)

Stage 1 - Machine Learning (as data deals with numbers)
Stage 2 - Supervised learning (requirement and inputs/outputs are clear)
Stage 3 - Classification (final output is in Categorical Data)

2 Basic information about dataset

The dataset has 399 rows and 25 columns

The Ouput Variable is Classification

3 PreProcessing :

In the dataset some of the are categorical and it is Nominal Type . So One Hot Encoding is used to convert it into numerical

4 Classification report and Confusion Matrix

Logistic Classification :

```
In [23]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, average = 'weighted')
         print(f1_macro)

         0.9924946382275899

In [24]: print('Confusion Matrix\n', cm)

         Confusion Matrix
          [[51  0]
          [ 1 81]]

In [25]: print('Classification Report\n', clf_report)

         Classification 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

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

Out[26]: 1.0
```

Best performing Parameter is {'penalty': 'l2', 'solver': 'newton-cg'}

Random Forest :

```
In [54]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, average = 'weighted')
         print(f1_macro)

         0.9849624060150376

In [55]: print('Confusion Matrix\n', cm)

         Confusion Matrix
          [[50  1]
          [ 1 81]]

In [56]: print('Classification Report\n', clf_report)

         Classification 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

In [57]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])

Out[57]: 0.9997608799617408
```

Best performing Parameter is {'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100}

Decision Tree :

```
In [43]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, average = 'weighted')
         print(f1_macro)

         0.9850141736106648

In [44]: print('Confusion Matrix\n', cm)

         Confusion Matrix
          [[51  0]
          [ 2 80]]

In [45]: print('Classification Report\n', clf_report)

         Classification Report
                       precision    recall  f1-score   support

                    0       0.96      1.00      0.98        51
                    1       1.00      0.98      0.99        82

             accuracy                           0.98       133
            macro avg       0.98      0.99      0.98       133
         weighted avg       0.99      0.98      0.99       133

In [46]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])

Out[46]: 0.9878048780487805
```

Best performing Parameter is {'criterion': 'log_loss', 'max_features': 'log2', 'splitter': 'random'}

Support Vector Machine :

```
In [40]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, average = 'weighted')
         print(f1_macro)

         0.9924946382275899

In [41]: print('Confusion Matrix\n', cm)

         Confusion Matrix
          [[51  0]
          [ 1 81]]

In [42]: print('Classification Report\n', clf_report)

         Classification 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

In [43]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])

Out[43]: 1.0
```

Best performing Parameter is {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}

Of the four model model **SVM and Logistic Classification** gave an accuracy of **0.99** and ROC Curve value as **1.0**