Problem Statement or Requirement:

A requirement from the Hospital, Management asked us to create a predictive model which will predict the Chronic Kidney Disease (CKD) based on the several parameters. The Client has provided the dataset of the same.

1. **Identifying The Problem Statement**:

*Machine Learning
*Supervised Learning
*Classification

2. ***Total Number Of Rows :** 399 rows
***Total Number Of Columns :** 28 columns

3. **Preprocessed Method:**
*Converted String Variables Into Numerical Value

4. **Good Model:**
*Logistic Regression **(1.0)**

5. ***Random Forest:**

```
[16]: from sklearn.metrics import f1_score
      f1_macro=f1_score(y_test,grid_predictions,average='weighted')
      print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

      The f1_macro value for best parameter {'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100}: 0.9924946382275899

[17]: print("The confusion Matrix:\n",cm)

      The confusion Matrix:
       [[51  0]
       [ 1 81]]

[18]: print("The report:\n",clf_report)

      The 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
```

***Decision Tree:**

[16]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'log2', 'splitter': 'random'}: 0.9924946382275899
```

[17]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[51  0]
 [ 1 81]]
```

[18]:
```
print("The report:\n",clf_report)
```

```
The 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
```

***Support Vector Machine:**

[25]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}: 0.9774002964206194
```

[17]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[49  2]
 [ 1 81]]
```

[18]:
```
print("The report:\n",clf_report)
```

```
The report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97        51
           1       0.98      0.99      0.98        82

    accuracy                           0.98       133
   macro avg       0.98      0.97      0.98       133
weighted avg       0.98      0.98      0.98       133
```

***Logistic Regression:**

[17]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 0.9924946382275899
```

[18]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[51  0]
 [ 1 81]]
```

[19]:
```
print("The report:\n",clf_report)
```

```
The 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
```

***K-Nearest Neighbors:**

[19]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'algorithm': 'auto', 'leaf_size': 20, 'n_neighbors': 9, 'p': 1, 'weights': 'distance'}: 0.7922855082912762
```

[20]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[46  5]
 [23 59]]
```

[21]:
```
print("The report:\n",clf_report)
```

```
The report:
              precision    recall  f1-score   support

           0       0.67      0.90      0.77        51
           1       0.92      0.72      0.81        82

    accuracy                           0.79       133
   macro avg       0.79      0.81      0.79       133
weighted avg       0.82      0.79      0.79       133
```

***Naive Bayes:**

[18]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test,grid_predictions,average='weighted')
print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)
```

```
The f1_macro value for best parameter {'alpha': 0.1, 'fit_prior': True}: 0.8516325059223402
```

[19]:
```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[50  1]
 [19 63]]
```

[20]:
```
print("The report:\n",clf_report)
```

```
The report:
              precision    recall  f1-score   support

           0       0.72      0.98      0.83        51
           1       0.98      0.77      0.86        82

    accuracy                           0.85       133
   macro avg       0.85      0.87      0.85       133
weighted avg       0.88      0.85      0.85       133
```

6. **Final model:**
   *Logistic Regression
   *Cause it gives (1.0) as result