Problem Statement & Solution

1.Identifying the problem - “Predicting Chronic Kidney Disease (CKD)”

Stage of Problem Identification

- 1. Machin Learning
- 2. Supervised Learning
- 3. Classification

2. Basic info about the dataset.

- X = age, bp, al, su, bgr, bu, sc, sod, pot, hrmo, pcv, wc, rc, sg_b, sg_c, sg_d, sg_e, rbc_normal, pc_normal, pcc_present, ba_present, htn_yes, dm_yes, cad_yes, appet_yes, pe_yes, ane_yes
- Y = classification_yes
- The total number of rows = 399
- The total number of columns = 28 (including the output column)

3. Pre-Processing method

The dataset I received from the client includes both

- numerical values
- categorical values

That categorical data column is nominal, meaning it does not possess any inherent ranking or order.

#To address this, I utilized one-hot encoding along with the “drop_first=True” function to minimize duplications within the columns.

4. To Develop a good model. Below listed algorithm used.

- SVM
- Decision Tree
- Random Forest
- Logistics Regression
- Ridge Classifier

5. All the research values

1. Support Vector Machine

Confusion Matric report and Roc score

```
[17]: 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

[18]: from sklearn.metrics import roc_auc_score

      roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

[18]: 1.0
```

- **2. Decision Tree:**

Confusion Matric report and Roc score

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

     The report:
                   precision    recall  f1-score   support

                0       0.91      0.96      0.93        51
                1       0.97      0.94      0.96        82

         accuracy                           0.95       133
        macro avg       0.94      0.95      0.94       133
     weighted avg       0.95      0.95      0.95       133

16]: from sklearn.metrics import roc_auc_score

     roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

16]: 0.9499043519846964
```

- **3. Random Forest:**

Confusion Matric report and Roc score

```
15]: 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

16]: from sklearn.metrics import roc_auc_score

     roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

16]: 1.0
```

- **4. Logistic Regression:**

Confusion Matric report and Roc score

```
15]: 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

16]: from sklearn.metrics import roc_auc_score

     roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

16]: 1.0
```

- **5. Ridge Classifier:**

Confusion Matric report and Roc score

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

The report:
                precision    recall  f1-score   support

           0       0.94      1.00      0.97        51
           1       1.00      0.96      0.98        82

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

```
[18]: from sklearn.metrics import roc_auc_score

decision_values = grid.decision_function(X_test)

roc_auc = roc_auc_score(y_test, decision_values)
print(roc_auc)

0.998804399808704
```

6. Final saved model.

After reviewing the confusion matrix report and the ROC score, the **SVM** model was found to be the best option for deployment. Although **Logistic Regression** and **Random Forest** produced similar outcomes, I chose one of them for the deployment.