Problem Statement:

A client's requirement is, he wants to predict the chronic kidney disease (CKD)based on several parameters. The Client has provided the dataset of the same. we must develop a model which will predict the CKD.

1.Identify your problem statement:

Stage 1: I identify that the given problem statement comes under **Machine Learning - Domain.** Clients provide datasets for prediction, the datasets are major in numerical values, so the machine learning domain is suitable for this problem.

Stage 2: Learning Selection

Based on the dataset and requirement from client is clear ,so it comes under '**Supervised Learning'**

Stage 3: Based on the dataset its comes under **Classification**

2. basic info about the dataset:

The dataset has clear inputs and output for the model creation.
The dataset contains **25** columns and **339** rows. The output column is 'Classsification'. Except classification column other columns are input.

3.Pre-processing method of data:

Except some columns all other columns are in numerical values.so i convert the categorical data into numerical values . It is Nominal data so i used One hot encoding by get_dummies using pandas in python library.

4.Developed Models:

Here i using a machine learning algorithms for classification to develop the varies model for this problems mentioned below:

- **Support Vector Machine**
- **Decision Tree**
- **Random Forest**
- **KNN**

- **Logistic Regression**
- **Guassain Naive bayes**
- **Multimonial Naive bayes**
- **Bernoulliu Naive bayes**
- **Categorical Naive bayes**
- **Complement Naive bayes**

5. Evaluation metric results of all the Models:

Logistic Regression Classifier:

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

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

           0       1.00      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
1.0
```

Parameters Used: 'C': 10, 'dual': False, 'penalty': 'l2', 'solver': 'lbfgs'

Support Vector Machine:

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

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

           0       1.00      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
1.0
```

Parameters Used: 'C': 10, 'gamma': 'scale', 'kernel': 'poly'

Decision Tree:

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

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

           0       1.00      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
1.0
```

Parameters Used:'criterion': 'gini', 'max_features': 'log2', 'splitter': 'random'

RandomForest:

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

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

           0       1.00      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
1.0
```

Parameters Used:'criterion': 'entropy', 'max_features': 'log2', 'n_estimators': 50

KNN(KNearest Neighbors):

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

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

           0       0.95      1.00      0.97       150
           1       1.00      0.97      0.98       249

    accuracy                           0.98       399
   macro avg       0.97      0.98      0.98       399
weighted avg       0.98      0.98      0.98       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.9998527443105756
```

Parameters Used:'algorithm': 'auto', 'metric': 'minkowski', 'n_neighbors': 5, 'p': 2, 'weights': 'uniform'

Gaussian Naive Bayes:

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

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

           0       0.96      0.99      0.98       150
           1       1.00      0.98      0.99       249

    accuracy                           0.98       399
   macro avg       0.98      0.98      0.98       399
weighted avg       0.98      0.98      0.98       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.9997590361445783
```

Parameters Used:'priors': None, 'var_smoothing': 1e-08

Multinomial Naive Bayes:

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

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

           0       0.73      0.97      0.83       150
           1       0.97      0.78      0.87       249

    accuracy                           0.85       399
   macro avg       0.85      0.87      0.85       399
weighted avg       0.88      0.85      0.85       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.9499866131191431
```

Parameters Used:'priors': 'alpha': 1.0, 'fit_prior': True, 'force_alpha': True

Bernaulli's Naive Bayes:

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

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

           0       0.97      0.99      0.98       150
           1       1.00      0.98      0.99       249

    accuracy                           0.99       399
   macro avg       0.98      0.99      0.99       399
weighted avg       0.99      0.99      0.99       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.999437751004016
```

Parameters Used:'priors':'alpha': 1.0, 'binarize': 0.0, 'fit_prior': True, 'force_alpha': True

Categorical Naive Bayes:

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

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

           0       0.99      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.9999196787148594
```

Parameters Used:'alpha': 1.0, 'fit_prior': True, 'force_alpha': True

Complement Naive Bayes:

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

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

           0       0.72      0.97      0.83       150
           1       0.97      0.78      0.86       249

    accuracy                           0.85       399
   macro avg       0.85      0.87      0.84       399
weighted avg       0.88      0.85      0.85       399
```

```
from sklearn.metrics import roc_auc_score

roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.9499866131191431
```

Parameters Used:'alpha': 'alpha': 1.0, 'fit_prior': True, 'force_alpha': True, 'norm': False

6.Final Model:

The final model I choose for this problem (CKD prediction) is **Logistic regression classifier** algorithm based model .Because it gives a high Accuracy value & ROC_AUC value. So I created the deployment phase for the Logistic Regression Classifier Model. The parameters used are **'C': 10, 'dual': False, 'penalty': 'l2', 'solver': 'lbfgs'** and the Accuracy value is **1.0 and ROC_AUC value is 1.0**.