1) Problem Identification

Stage 1 : Domain Selection

Input – Number

Domain – Machine Learning

Stage 2 : Learning Selection

Supervised

Stage 3 : Classification or Regression

Here in the dataset the output label contains **Categorical values**. So it falls under **Classification**.

2) Basic Information about Dataset

File Name	: **CKD.csv**
Total number of Rows	: **399**
Total number of Columns	: **25**
Input Columns	: **24 ['age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu', 'sc', 'sod', 'pot', 'hrmo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane']**
Output Columns	: **1 [classification]**

3) Pre processing Method

Here in the dataset we have **twelve columns** as **nominal data.**

[sg,bc,pc,pcc,ba,htn,dm,cad,appet,pe,ane,classification].

4) Machine Learning Algorithms

For this section refer created python files with many models

5) Confusion Matrix and Classification Report

A. Logistic Grid – Classification

In [14]:
```
 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': 'sag'} 0.9924946382275899
```

In [15]:
```
print('The Confusion Matrix:\n',cm)
```

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

In [16]:
```
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
```

In [17]:
```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])
```

Out[17]: 1.0

B.Support Vector Machine Grid – Classification

In [13]:
```
 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': 'auto', 'kernel': 'sigmoid'} 0.9924946382275899
```

In [14]:
```
print('The Confusion Matrix:\n',cm)
```

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

In [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
```

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

Out[16]: 1.0

C.Decision Tree Grid – Classification

In [13]:
```
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': 'auto', 'splitter': 'best'} 0.9402326571144843
```

In [14]:
```
print('The Confusion Matrix:\n',cm)
```

```
The Confusion Matrix:
 [[49  2]
 [ 6 76]]
```

In [15]:
```
print('The Report:\n',clf_report)
```

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

           0       0.89      0.96      0.92        51
           1       0.97      0.93      0.95        82

    accuracy                           0.94       133
   macro avg       0.93      0.94      0.94       133
weighted avg       0.94      0.94      0.94       133
```

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

Out[16]:
```
0.9438067910090866
```

D.Random Forest Grid – Classification

In [13]:
```
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': 'sqrt', 'n_estimators': 100} 0.9849624060150376
```

In [14]:
```
print('The Confusion Matrix:\n',cm)
```

```
The Confusion Matrix:
 [[50  1]
 [ 1 81]]
```

In [15]:
```
print('The Report:\n',clf_report)
```

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

Out[16]:
```
0.9997608799617408
```

6) Final Model (Overall Performance-Accuracy)

Logistic Grid Classification	**-**	**0.99**
Support Vector Machine Grid Classification	**-**	**0.99**
Decision Tree Grid Classification	-	0.94
Random Forest Grid Classification	-	0.98

So from the above analysis we can get the best accuracy with **Logistic Grid Classification** and **Support Vector Machine Grid Classification algorithms .**So we can choose any one of them for the **final model.**