Classification algorithms for chronic Kidney Disease Prediction

1. Identifying the problem statement

Hospital's requirement is, management asked to create a predictive model, which will predict the Chronic Kidney Disease, based on the several parameters. The client has provided the dataset of the same.

Description:

With the given parameters such as Person's age,BP,Albumin,sugar,Blood Glucose Random,Blood urea,Creatinine,Sodium,Potassium,Haemoglobin, Packed Cell Volume,WBC Count, RBC Count,Specific gravity b,c,d,e,RBC Normal,Pus Cell Normal,Pus Cell Clumps Present,Bacteria,Hypertension, Diabetes,Coronary artery disease,Apetite,Peda Edema,Anaemia
We need to predict, whether person has Chronic Kidney disease or not

Stage 1: Domain Selection
Since the input is in number format, **Machine Learning** can be used to solve this problem
Stage 2: Learning Selection
In the given data set, input and output are well defined, so **Supervised Learning** can be used to solve this problem
Stage 3: Type of algorithm: **Classification**

2. Basic Information about the given dataset

Dataset name: CKD.csv
The given Chronic kidney disease dataset contains 24 input columns & 1 output column

Input column names:

Age (type: numerical)
Blood Pressure (type: numerical)
Specific gravity (type: categorical-nominal)
Albumin(type: numerical)
Sugar(type: numerical)
Red Blood Cells(type: categorical-nominal)
Pus Cells(type: categorical-nominal)
Pus Cell Clumps(type: categorical-nominal)
Bacteria(type: categorical-nominal)
Blood Glucose random(type: numerical)
Blood Urea(type: numerical)

Serum Creatinine(type: numerical)
Sodium(type: numerical)
Potassium(type: numerical)
Haemoglobin(type: numerical)
Packed Cell Volume(type: numerical)
Red Blood Cell Count(type: numerical)
White Blood Cell Count(type: numerical)
Hypertension(type: categorical-nominal)
Diabetes Mellitus(type: categorical-nominal)
Coronary artery Disease(type: categorical-nominal)
Appetite(type: categorical-nominal)
Peda Edema(type: categorical-nominal)
Anaemia(type: categorical-nominal)

Output column name:

Charges (type: categorical-nominal)

Total number of rows present: 399
Total number of columns present: 25

3. **Pre-processing method**

- **One Hot Encoding**
- **Standardization**

There are 11 categorical(nominal) columns namely Specific gravity, RBC, Pus Cells, Pus Cells clumps, Bacteria, Hypertension, Diabetes Mellitus, Coronary Artery disease, Appetite, Peda Adema, Anaemia in the data set. So, in order to apply classification algorithms, we convert those categorical columns into numerical columns using **One Hot Encoding** technique

Most of the columns have different number of digits, especially few columns contains single digit(1's and 0's) after applying One hot Encoding. So, to make all columns into same scale, we apply **Standardization** technique.

4. We develop algorithms for Classification – SVM, Decision tree, Random Forest and Logistic Regression – Obtain classification report, Receiver Operating characteristics Area under Curve score of various models.

5. We documented the Classification report, Receiver Operating Characteristics Area Under Curve scores for various classification models

- **Support Vector Machine - Classification**

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

Classification Report:

```
#Printing the Classification report
print("The report:\n",clf_report)
```

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

          no       0.98      1.00      0.99        51
         yes       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
```

- The percentage of correct classification of persons with chronic Kidney disease and persons without chronic kidney disease to the total number of persons available in the test set is 99% **(Accuracy)**
- The percentage of correct classification of people having kidney disease to the sum of correctly classified as people having kidney disease and wrongly classified as people having kidney disease in the test Set is 1.0 **(Precision)**

  The percentage of correct classification of people not having kidney disease to the sum of correctly classified as people not having kidney disease and wrongly classified as people not having kidney disease in the test set is 0.98 **(Precision)**
- The percentage of correct classification of people having kidney disease to the total count of people having kidney disease in the test set is 0.99 **(Recall)**

  The percentage of correct classification of people not having kidney disease to the total count of people not having kidney disease in the test set is 1.0 **(Recall)**
- The overall performance of people having chronic kidney disease is 0.99 **(F1 Measure)**

The overall performance of people not having chronic kidney dise ase is 0.99 **(F1-Measure)**

Receiver Operating Characteristic - Area Under Curve score: **1.0**

Best Parameters as part of SVM Classification model:

{'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}

F1 score calculated for best parameters is: **0.9924**

- **Decision Tree - Classification**

The confusion Matrix:
[[49 2]
[6 76]]

Classification Report:

```
#Printing the Classification report
print("The report:\n",clf_report)
```

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

          no       0.89      0.96      0.92        51
         yes       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
```

Receiver Operating Characteristic - Area Under Curve score: **0.9438**

Best Parameters as part of Decision Tree Classification model:

{'criterion': 'gini', 'max_features': 'log2', 'splitter': 'random'}

F1 score calculated for best parameters is: **0.9402**

- **Random Forest - Classification**

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

Classification Report:

```
#Printing the Classification report
print("The report:\n",clf_report)
```

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

          no       0.98      1.00      0.99        51
         yes       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
```

Receiver Operating Characteristic - Area Under Curve score: **1.0**

Best Parameters as part of Random Forest Classification model:

{'criterion': 'entropy', 'max_features': 'log2', 'n_estimators': 100}

F1 score calculated for best parameters is: **0.9924**

- **Logistic Regression - Classification**

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

Classification Report:

```
#Printing the Classification report
print("The report:\n",clf_report)
```

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

          no       0.98      1.00      0.99        51
         yes       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
```

Receiver Operating Characteristic - Area Under Curve score: **1.0**

Best Parameters as part of Logistic Regression (Classification) model:

{'penalty': 'l2', 'solver': 'newton-cg'}

F1 score calculated for best parameters is: **0.9924**

6. Final Model:

Except Decision tree model, other models such as Support Vector machi ne, Random Forest and Logistic Regression gives good results.
Therefore the final model can be considered to be either of 3 models - SVM/Logistic Regression/Random Forest classifications

Classification model comparison:

Classification Model	Accuracy	Precision	Recall	f1-Score	ROC_AUC Score
Support Vector Machine	99%	0.99	0.99	0.99	1
Random forest	99%	0.99	0.99	0.99	1
Logistic Regression	99%	0.99	0.99	0.99	1

Note: The macro average has been calculated for Precision, Recall and f1 -Score values and mentioned in the table above.

Best Parameters:
SVM: {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'}
Random Forest: {'criterion': 'entropy', 'max_features': 'log2', 'n_estimators': 100}
Logistic Regression: {'penalty': 'l2', 'solver': 'newton-cg'}