CLASSIFICATION-ASSIGNMENT

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. Identify your problem statement.

   To predict the Chronic Kidney Disease using the given parameters.

   3 Stages:
   Stage 1: Machine Learning
   Stage 2: Supervised
   Stage 3: Classification

2. Tell basic info about the dataset (Total number of rows, columns)

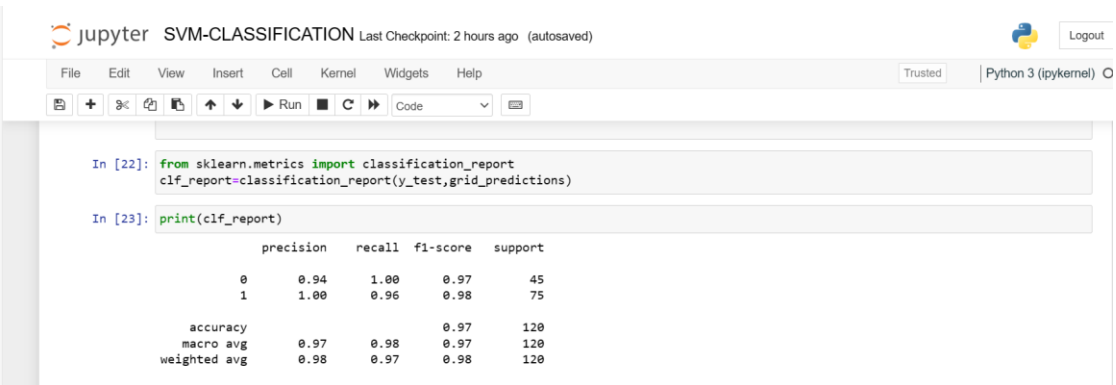
   399 rows × 25 columns
   Total number of rows: 399
   Total number of columns: 25

3. Mention the pre-processing method if you're doing any (like converting string to number – nominal data).

   10 Columns is in categorical data. So we have to convert into nominal data as a pre-processing work.

4. Develop a good model with good evaluation metric. You can use any machine learning algorithm; you can create many models. Finally, you have to come up with final model.
5. All the research values of each algorithm should be documented. (You can make tabulation or screenshot of the results

SVM:



Random Forest:

jupyter Random Forest-Classification Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
In [16]: from sklearn.metrics import confusion_matrix
         cm=confusion_matrix(y_test,grid_predictions)

In [17]: from sklearn.metrics import classification_report
         clf_report=classification_report(y_test,grid_predictions)

In [18]: print(clf_report)

              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.99      0.99        75

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120
```

Decision Tree:

jupyter Decision Tree-Classification Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
In [16]: from sklearn.metrics import confusion_matrix
         cm=confusion_matrix(y_test,grid_predictions)

In [17]: from sklearn.metrics import classification_report
         clf_report=classification_report(y_test,grid_predictions)

In [18]: print(clf_report)

              precision    recall  f1-score   support

           0       0.98      0.98      0.98        45
           1       0.99      0.99      0.99        75

    accuracy                           0.98       120
   macro avg       0.98      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

Logistic Regression:

jupyter Logistic Regression-classification Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
In [22]: from sklearn.metrics import classification_report
         clf_report=classification_report(y_test,grid_predictions)

In [23]: print(clf_report)

              precision    recall  f1-score   support

           0       0.98      1.00      0.99        45
           1       1.00      0.99      0.99        75

    accuracy                           0.99       120
   macro avg       0.99      0.99      0.99       120
weighted avg       0.99      0.99      0.99       120

In [24]: table=pd.DataFrame.from_dict(re)

In [25]: table
```

KNN:

jupyter KNN-Classification Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
cm=confusion_matrix(y_test,grid_predictions)
```

In [17]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,grid_predictions)
```

In [18]:
```
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.90      1.00      0.95        45
           1       1.00      0.93      0.97        75

    accuracy                           0.96       120
   macro avg       0.95      0.97      0.96       120
weighted avg       0.96      0.96      0.96       120
```

In [19]:
```
table=pd.DataFrame.from_dict(re)
```

Naives Bayes:

Gaussian Naïve Bayes:

jupyter Gaussian Naives Bayes- Classification Last Checkpoint: 42 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [20]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

In [21]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,y_pred)
```

In [22]:
```
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        45
           1       1.00      0.96      0.98        75

    accuracy                           0.97       120
   macro avg       0.97      0.98      0.97       120
weighted avg       0.98      0.97      0.98       120
```

Bernoulli Naïve Bayes:

jupyter Bernoulli Naives Bayes- Classification Last Checkpoint: 13 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

In [23]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,y_pred)
```

In [24]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(y_test,y_pred)
```

In [25]:
```
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97        45
           1       1.00      0.96      0.98        75

    accuracy                           0.97       120
   macro avg       0.97      0.98      0.97       120
weighted avg       0.98      0.97      0.98       120
```

6. Mention your final model, justify why u have chosen the same.

I have chosen Random Forest algorithm with the accuracy of 0.99 and the macro_average of 0.99.

It will provide good performance and prediction for the problem statement.