Classification Assignment - CKD

Problem Statement or Requirement:

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.

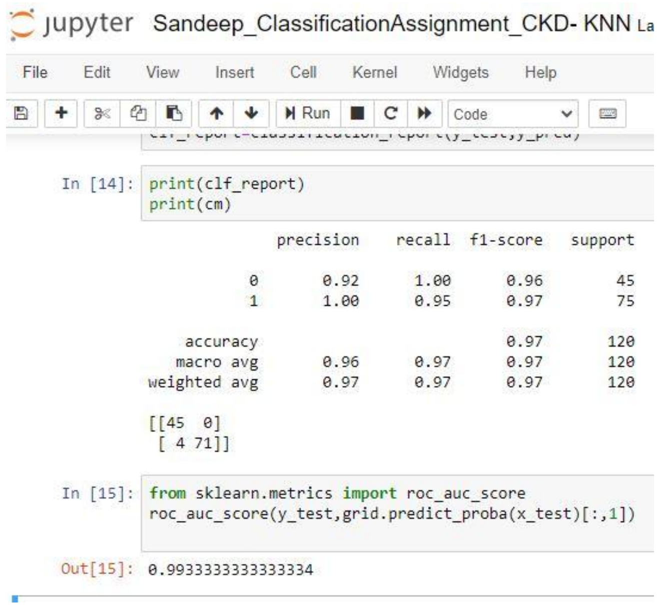
1.) Identify your problem statement

The Given problem is a classification under supervised learning, since problem statement and data are labeled properly including target variable.

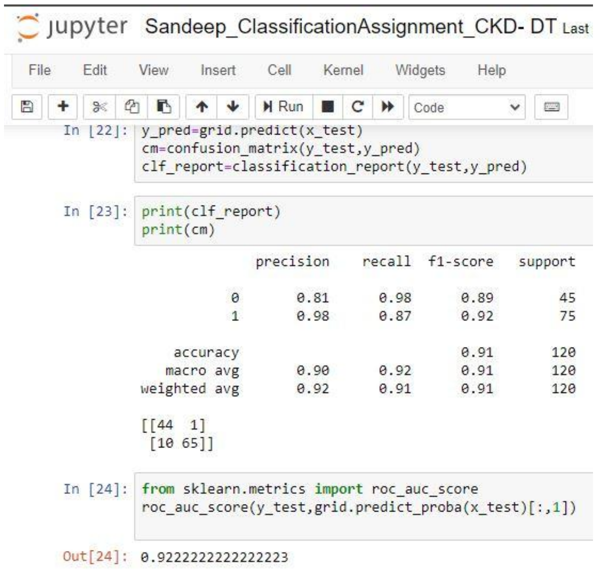
2.) Tell basic info about the dataset (Total number of rows, columns)
- In data set we have 399 rows and 25 columns including categorical data column
- We have 24 input indulging categorical Colum (11 column) and one categorical output Column

3.) Mention the pre-processing method if you're doing any (like converting string to number – nominal data)
- We need to convert the nominal data using one hot encoding method
- We need to use standardization technique to minimize the difference between the values

4.) All the research values of each algorithm should be documented. (You can make tabulation or screenshot of the results.)

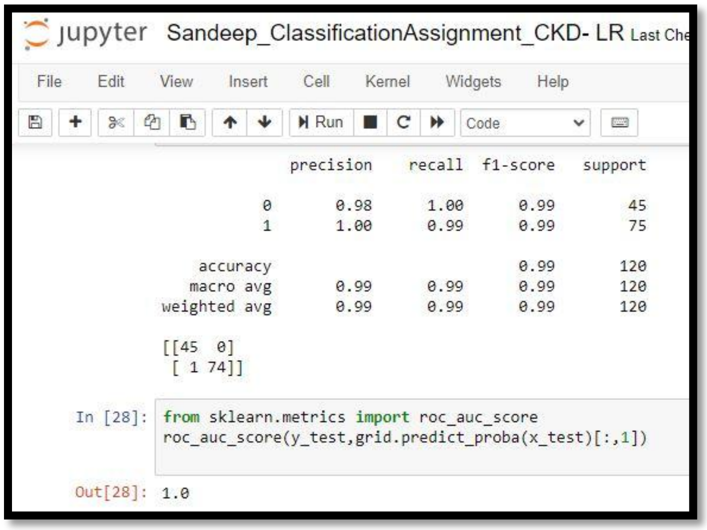

jupyter Sandeep_ClassificationAssignment_CKD- KNN La
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
In [14]: print(clf_report)
         print(cm)

                  precision    recall  f1-score   support

               0       0.92      1.00      0.96        45
               1       1.00      0.95      0.97        75

        accuracy                           0.97       120
       macro avg       0.96      0.97      0.97       120
    weighted avg       0.97      0.97      0.97       120

[[45  0]
 [ 4 71]]

In [15]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[15]: 0.9933333333333334
```

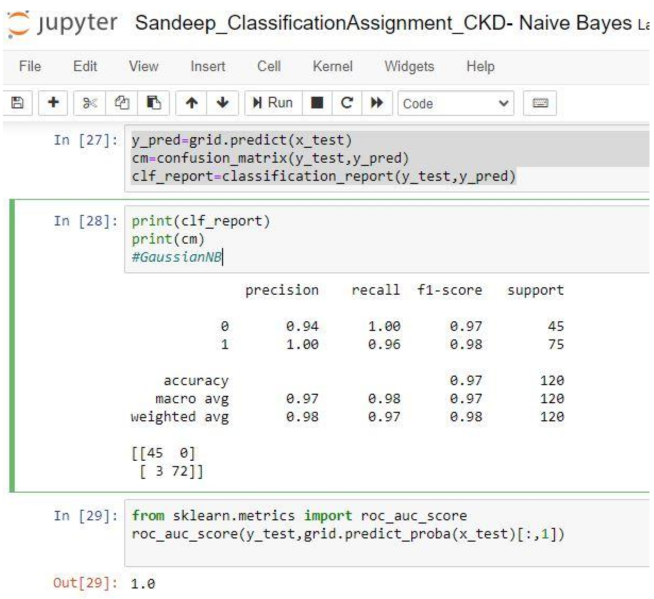

jupyter Sandeep_ClassificationAssignment_CKD- DT Last
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
In [22]: y_pred=grid.predict(x_test)
         cm=confusion_matrix(y_test,y_pred)
         clf_report=classification_report(y_test,y_pred)

In [23]: print(clf_report)
         print(cm)

                  precision    recall  f1-score   support

               0       0.81      0.98      0.89        45
               1       0.98      0.87      0.92        75

        accuracy                           0.91       120
       macro avg       0.90      0.92      0.91       120
    weighted avg       0.92      0.91      0.91       120

[[44  1]
 [10 65]]

In [24]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[24]: 0.9222222222222223
```

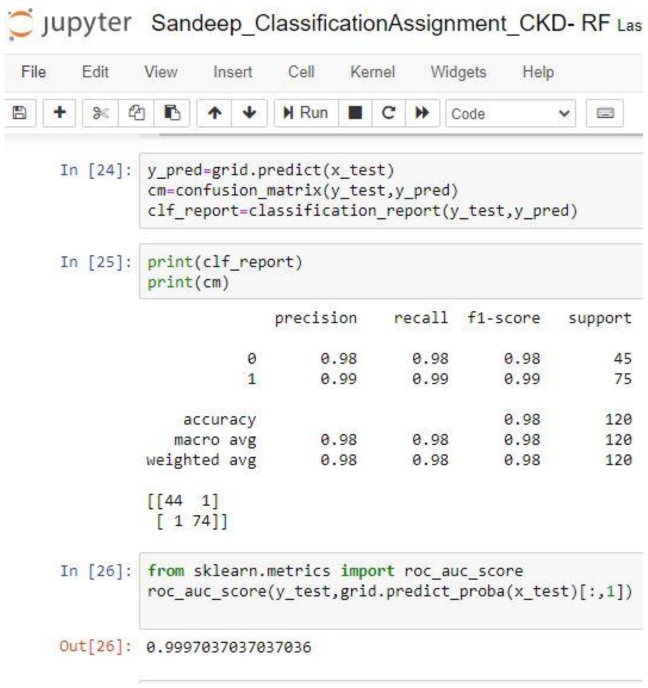
Classification Assignment - CKD


jupyter Sandeep_ClassificationAssignment_CKD- LR Last Che
File Edit View Insert Cell Kernel Widgets Help
Run Code

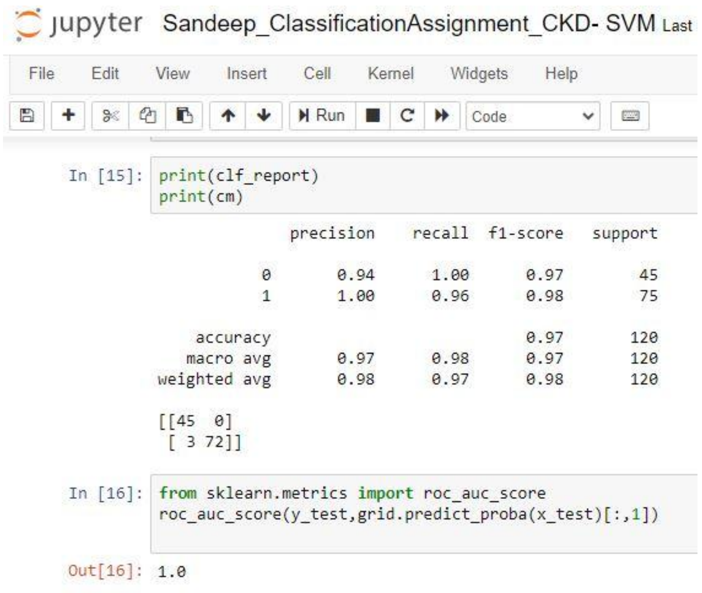
```
              precision    recall  f1-score   support

           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

[[45  0]
 [ 1 74]]

In [28]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[28]: 1.0
```


jupyter Sandeep_ClassificationAssignment_CKD- Naive Bayes La
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
In [27]: y_pred=grid.predict(x_test)
         cm=confusion_matrix(y_test,y_pred)
         clf_report=classification_report(y_test,y_pred)

In [28]: print(clf_report)
         print(cm)
         #GaussianNB

              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

[[45  0]
 [ 3 72]]

In [29]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[29]: 1.0
```

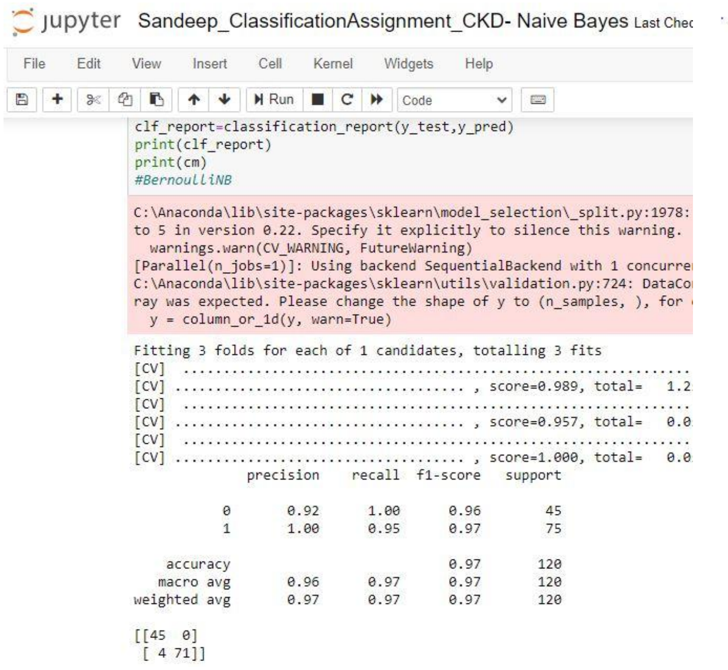

jupyter Sandeep_ClassificationAssignment_CKD- RF Las
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
In [24]: y_pred=grid.predict(x_test)
         cm=confusion_matrix(y_test,y_pred)
         clf_report=classification_report(y_test,y_pred)

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

              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

[[44  1]
 [ 1 74]]

In [26]: from sklearn.metrics import roc_auc_score
         roc_auc_score(y_test,grid.predict_proba(x_test)[:,1])

Out[26]: 0.9997037037037036
```


jupyter Sandeep_ClassificationAssignment_CKD- SVM Last
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
In [15]: print(clf_report)
         print(cm)

              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

[[45  0]
 [ 3 72]]

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

Out[16]: 1.0
```

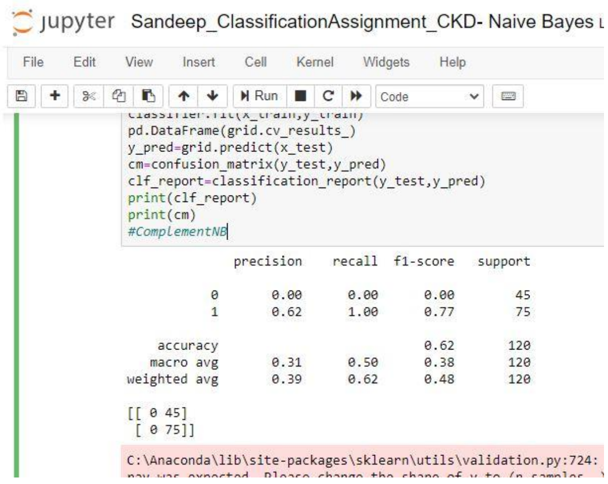
Classification Assignment - CKD


jupyter Sandeep_ClassificationAssignment_CKD- Naive Bayes Last Chec
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
clf_report=classification_report(y_test,y_pred)
print(clf_report)
print(cm)
#BernoulliNB
```

```
C:\Anaconda\lib\site-packages\sklearn\model_selection\_split.py:1978:
to 5 in version 0.22. Specify it explicitly to silence this warning.
  warnings.warn(CV_WARNING, FutureWarning)
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurre
C:\Anaconda\lib\site-packages\sklearn\utils\validation.py:724: DataCo
ray was expected. Please change the shape of y to (n_samples, ), for
  y = column_or_1d(y, warn=True)
```

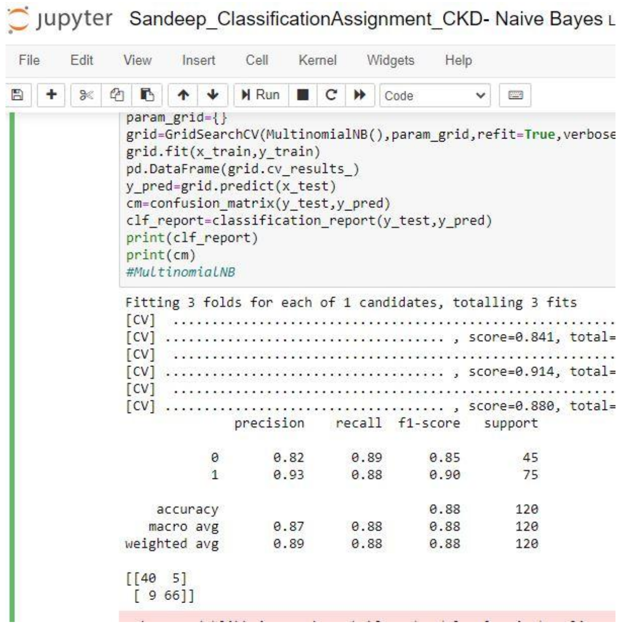
```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV]  ................................................................
[CV] .................................... , score=0.989, total=   1.2
[CV]  ................................................................
[CV] .................................... , score=0.957, total=   0.0
[CV]  ................................................................
[CV] .................................... , score=1.000, total=   0.0
              precision    recall  f1-score   support

           0       0.92      1.00      0.96        45
           1       1.00      0.95      0.97        75

    accuracy                           0.97       120
   macro avg       0.96      0.97      0.97       120
weighted avg       0.97      0.97      0.97       120

[[45  0]
 [ 4 71]]
```


jupyter Sandeep_ClassificationAssignment_CKD- Naive Bayes
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
pd.DataFrame(grid.cv_results_)
y_pred=grid.predict(x_test)
cm=confusion_matrix(y_test,y_pred)
clf_report=classification_report(y_test,y_pred)
print(clf_report)
print(cm)
#ComplementNB
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        45
           1       0.62      1.00      0.77        75

    accuracy                           0.62       120
   macro avg       0.31      0.50      0.38       120
weighted avg       0.39      0.62      0.48       120

[[ 0 45]
 [ 0 75]]
```

```
C:\Anaconda\lib\site-packages\sklearn\utils\validation.py:724:
```


jupyter Sandeep_ClassificationAssignment_CKD- Naive Bayes
File Edit View Insert Cell Kernel Widgets Help
Run Code

```
param_grid={}
grid=GridSearchCV(MultinomialNB(),param_grid,refit=True,verbose
grid.fit(x_train,y_train)
pd.DataFrame(grid.cv_results_)
y_pred=grid.predict(x_test)
cm=confusion_matrix(y_test,y_pred)
clf_report=classification_report(y_test,y_pred)
print(clf_report)
print(cm)
#MultinomialNB
```

```
Fitting 3 folds for each of 1 candidates, totalling 3 fits
[CV]  ................................................................
[CV] .................................... , score=0.841, total=
[CV]  ................................................................
[CV] .................................... , score=0.914, total=
[CV]  ................................................................
[CV] ..................................... , score=0.880, total=
              precision    recall  f1-score   support

           0       0.82      0.89      0.85        45
           1       0.93      0.88      0.90        75

    accuracy                           0.88       120
   macro avg       0.87      0.88      0.88       120
weighted avg       0.89      0.88      0.88       120

[[40  5]
 [ 9 66]]
```

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

I have choose **Logistic Regression** as the best model it gave us best accuracy and it's ROC value is one.