Classification Assignment

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 problem statement is to create a predictive model to identify Chronic Kidney Disease (CKD) based on several parameters provided in the dataset.

3-Stages of the problem

Stage-1	Domain Selection	Machine Learning
Stage-1	Learning Selection	Supervised Learning
Stage-3	Classification / Regression	Classification

2. Tell basic info about the dataset:

Name of the data set: CKD.csv
Total no. of Columns: 25
Total no. of rows: 399

3. Mention the pre-processing method (like converting string to number – nominal data)

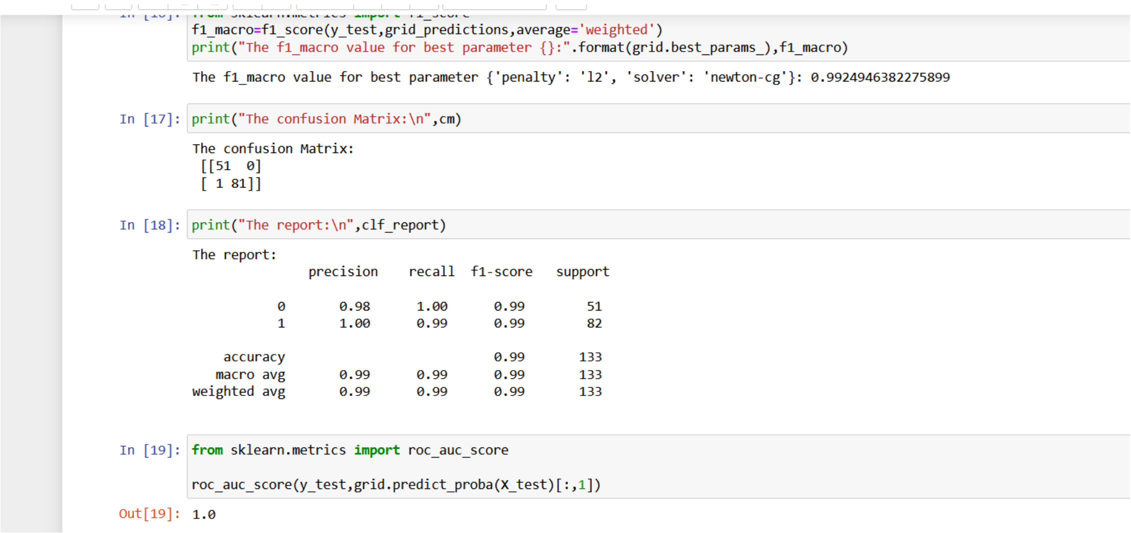
one-hot encoding categorical used for converting categorical variables into dummy / indicator variables using the code
dataset=pd.get_dummies(dataset,dtype=int,drop_first=True)

4. Develop a good model with good evaluation metric.

S. No.	Model / Evaluation Matrics	Accuracy
1	Logistic Regression - LR	0.99
2	Support Vector Machine - SVM	0.99
3	Decision Tree - DC	0.96
4	Random Forest – RF	1.00

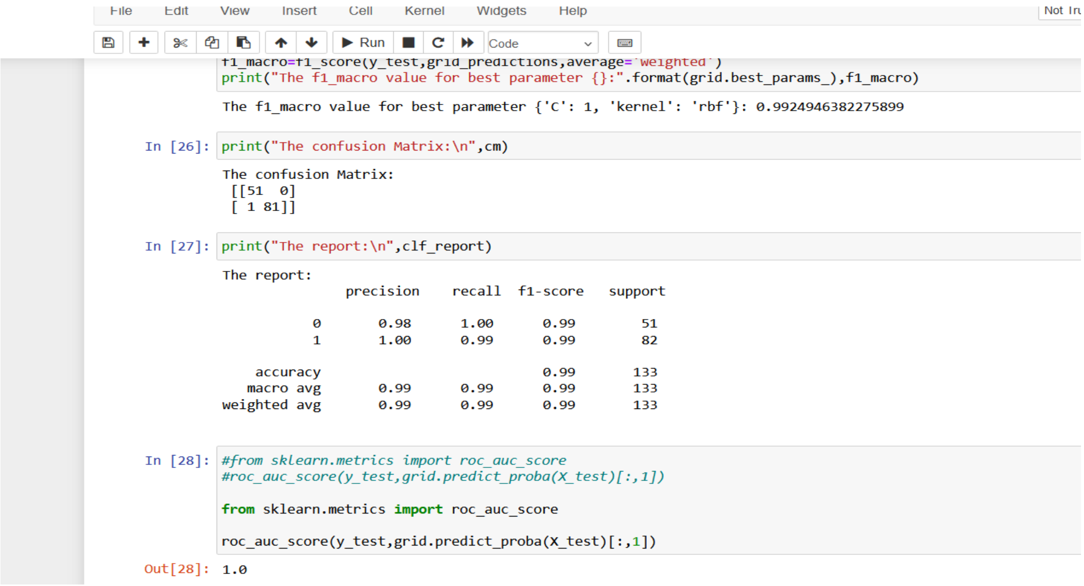
5. All the research values of each algorithm should be documented.

1. Screen Shot of – Logistic Regression



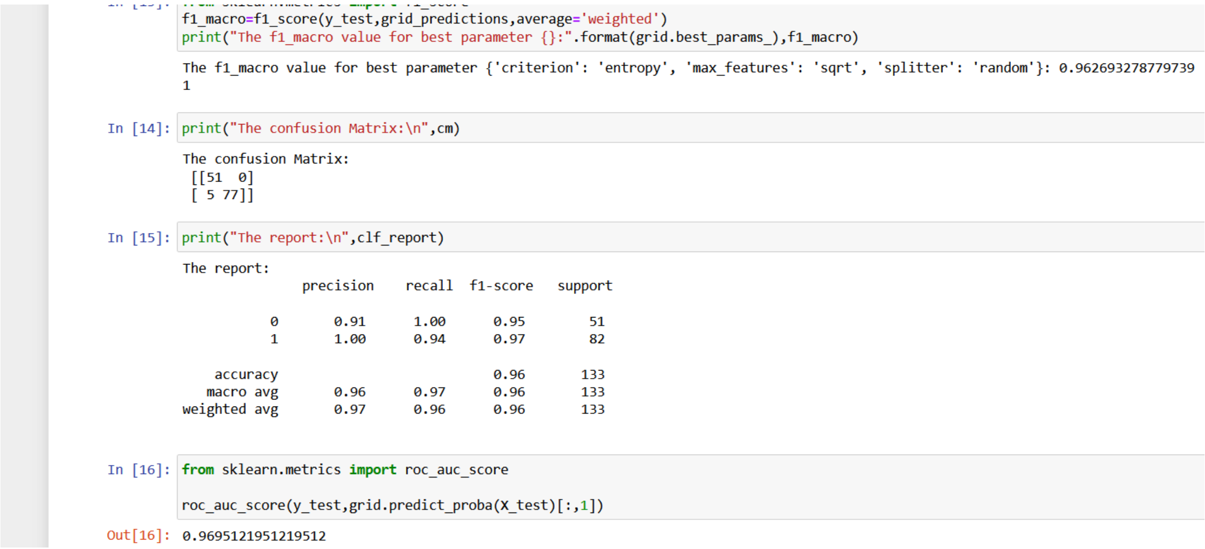
```
In [16]: from sklearn.metrics import f1_score
         f1_macro=f1_score(y_test,grid_predictions,average='weighted')
         print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

         The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 0.9924946382275899

In [17]: print("The confusion Matrix:\n",cm)

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

In [18]: 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 [19]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

Out[19]: 1.0
```

2. Screen Shot of SVM



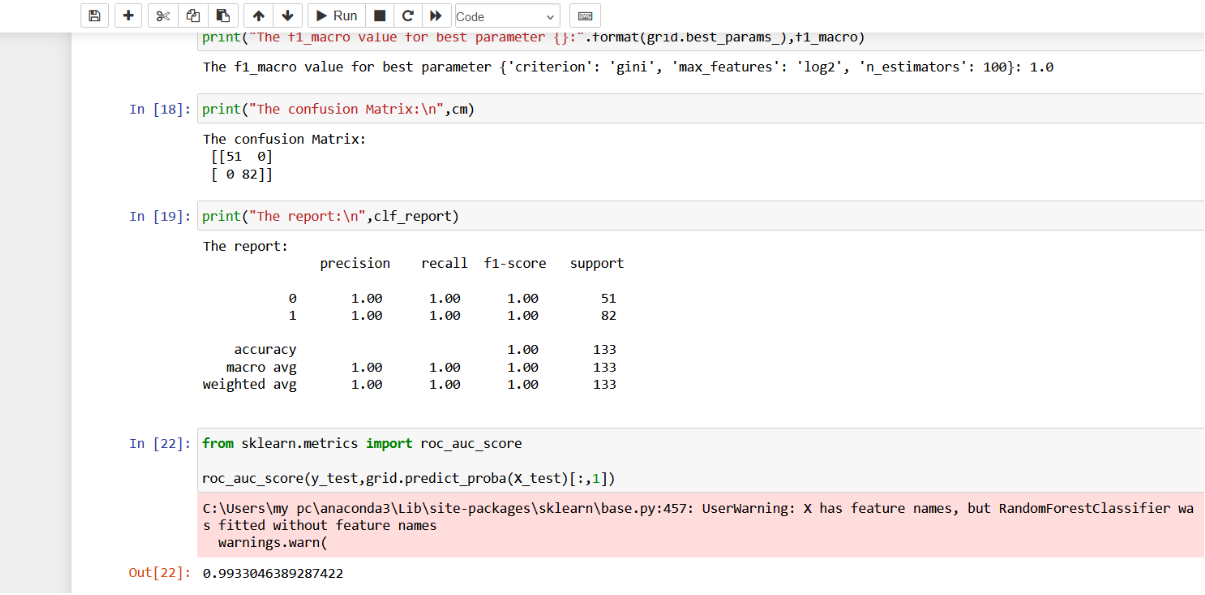
```
         f1_macro=f1_score(y_test,grid_predictions,average='weighted')
         print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

         The f1_macro value for best parameter {'C': 1, 'kernel': 'rbf'}: 0.9924946382275899

In [26]: print("The confusion Matrix:\n",cm)

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

In [27]: 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 [28]: #from sklearn.metrics import roc_auc_score
         #roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

         from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

Out[28]: 1.0
```

3. Screen Shot of DC



```
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': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'}: 0.962693278779739
1

In [14]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[51  0]
 [ 5 77]]

In [15]: print("The report:\n",clf_report)

The report:
               precision    recall  f1-score   support

           0       0.91      1.00      0.95        51
           1       1.00      0.94      0.97        82

    accuracy                           0.96       133
   macro avg       0.96      0.97      0.96       133
weighted avg       0.97      0.96      0.96       133


In [16]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

Out[16]: 0.9695121951219512
```

4. Screen Shot of RF



```
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': 'log2', 'n_estimators': 100}: 1.0

In [18]: print("The confusion Matrix:\n",cm)

The confusion Matrix:
 [[51  0]
 [ 0 82]]

In [19]: print("The report:\n",clf_report)

The report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        51
           1       1.00      1.00      1.00        82

    accuracy                           1.00       133
   macro avg       1.00      1.00      1.00       133
weighted avg       1.00      1.00      1.00       133


In [22]: from sklearn.metrics import roc_auc_score

         roc_auc_score(y_test,grid.predict_proba(X_test)[:,1])

C:\Users\my pc\anaconda3\Lib\site-packages\sklearn\base.py:457: UserWarning: X has feature names, but RandomForestClassifier wa
s fitted without feature names
  warnings.warn(

Out[22]: 0.9933046389287422
```

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

Given the evaluation metrics, all models seem to perform well, with high accuracy scores. However, the Random Forest model stands out with a perfect accuracy score of 1.00, indicating that it has correctly classified all instances in the dataset.

Final Model Selection: Random Forest (RF)

Overall, based on its high accuracy the Random Forest model is chosen as the final model for predicting Chronic Kidney Disease.