Classification Assignment

problem Statement:

The task is to develop a predictive model to identify the presence of Chronic Kidney Disease (CKD) based on a set of medical parameters. The goal is to help the hospital management by accurately predicting CKD using the provided dataset.

Basics details of the dataset:

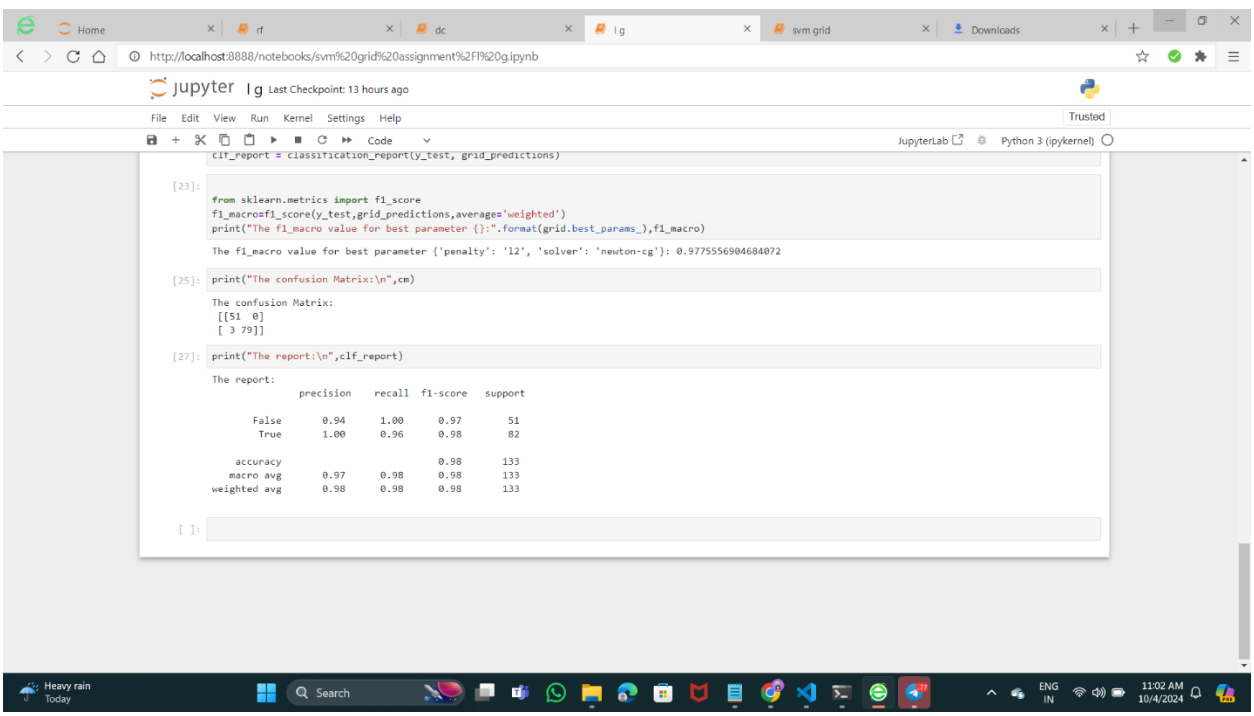
It contains 399 rows and 25 cols.

indep=dataset[['age','bp','al','su','bgr','bu','sc','sc','sod','pot','hrmo','pc_normal','pcc_present','ba_present','htn_yes','dm_yes','cad_yes','appet_yes','pe_yes','ane_yes']]

dep=dataset['classification_yes']

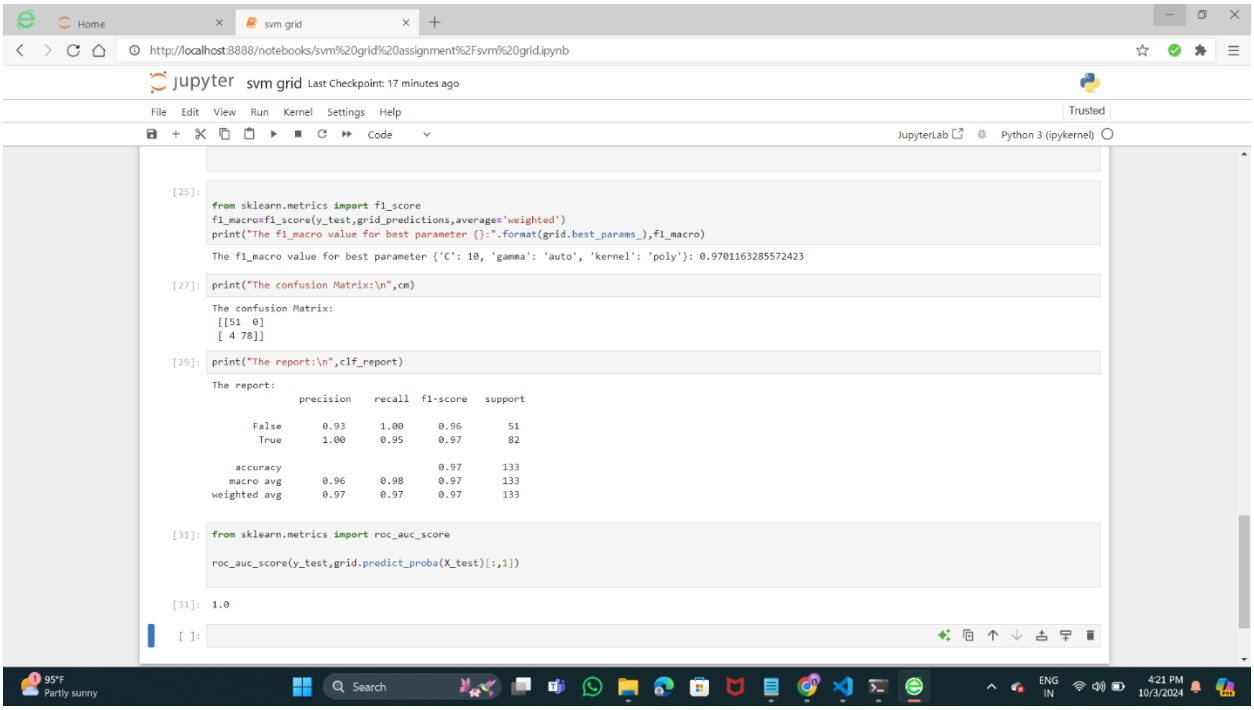
These are the input and output.

Developed a good model :



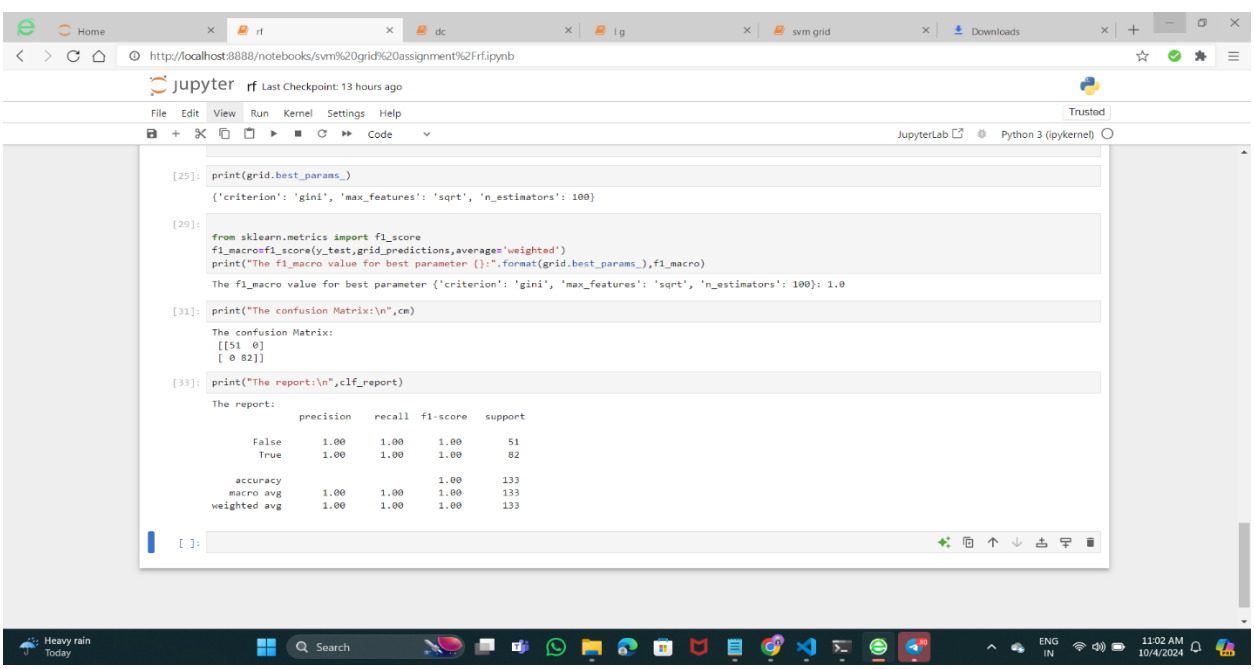
I worked on Logistic regression classification, SVM classification, Dc and random forest classification. Here Logistic regression classification gives us the best accuracy and best performance. So, after working on various algorithm I decided that Logistic regression classification gives me good result. It gives a accuracy of 0.98. So, it’s a good model.

All the researched values of algorithm are documented here.



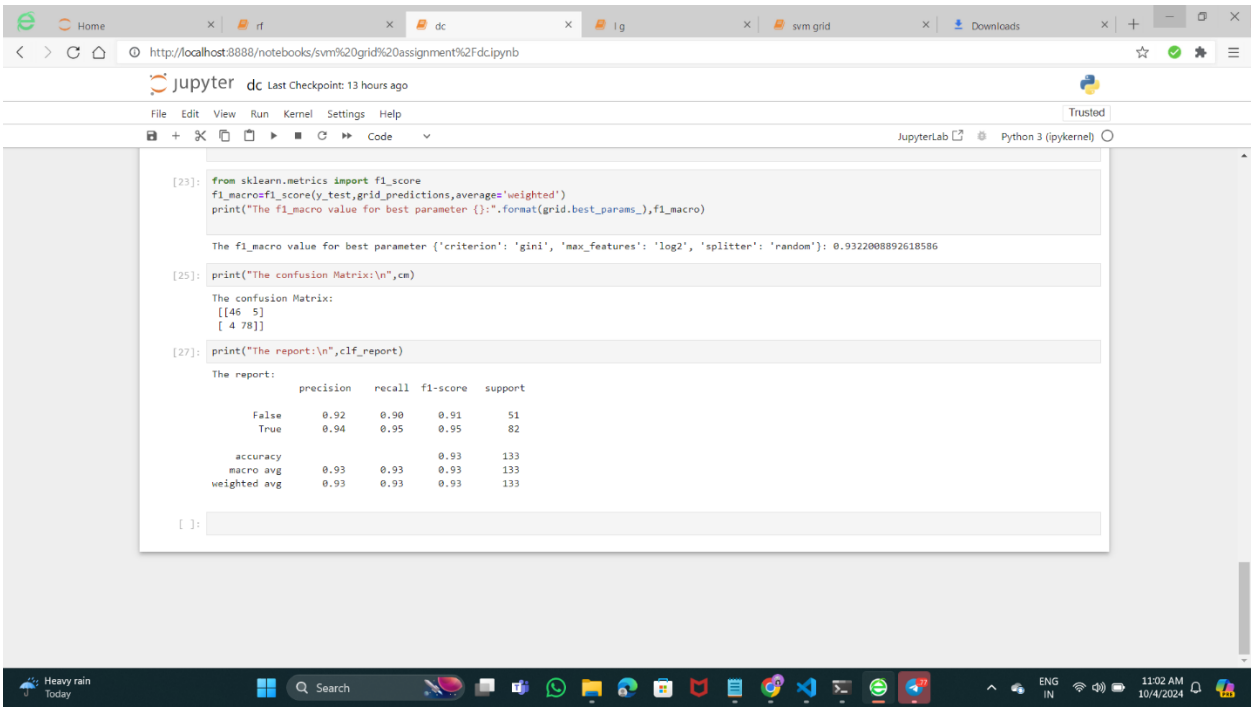
```
[25]: from sklearn.metrics import f1_score
      f1_macro=f1_score(y_test,grid_predictions,average='weighted')
      print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

      The f1_macro value for best parameter {'C': 10, 'gamma': 'auto', 'kernel': 'poly'}: 0.9701163285572423

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

      The confusion Matrix:
       [[51  0]
       [ 4 78]]

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

      The report:
                    precision    recall  f1-score   support

            False       0.93      1.00      0.96        51
             True       1.00      0.95      0.97        82

         accuracy                           0.97       133
        macro avg       0.96      0.98      0.97       133
     weighted avg       0.97      0.97      0.97       133

[31]: from sklearn.metrics import roc_auc_score

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

[31]: 1.0
```

worked on svm model



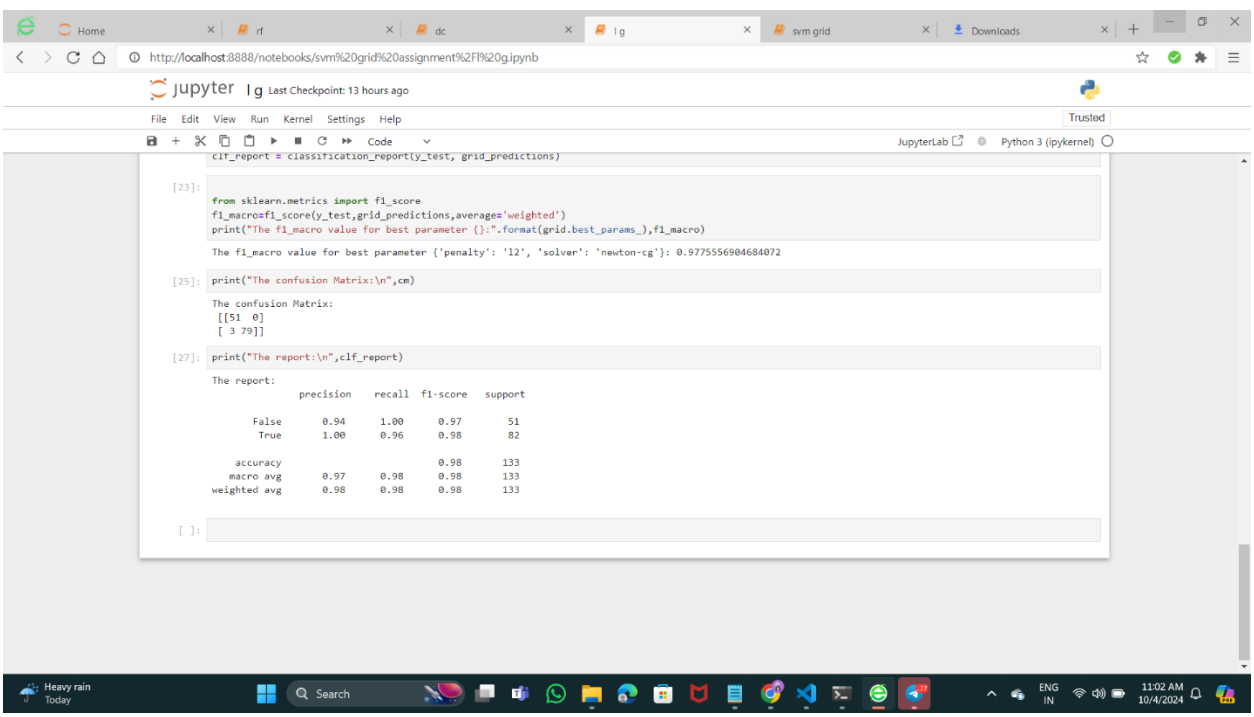
```
[25]: print(grid.best_params_)

      {'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 100}

[29]: 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}: 1.0

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

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

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

      The report:
                    precision    recall  f1-score   support

            False       1.00      1.00      1.00        51
             True       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
```

Worked on random forest classification



```
[23]: 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': 'log2', 'splitter': 'random'}: 0.9322008892618586

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

The confusion Matrix:
 [[46  5]
 [ 4 78]]

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

The report:
              precision    recall  f1-score   support

       False       0.92      0.90      0.91        51
        True       0.94      0.95      0.95        82

    accuracy                           0.93       133
   macro avg       0.93      0.93      0.93       133
weighted avg       0.93      0.93      0.93       133

[ ]:
```

Worked on dc



```
clf_report = classification_report(y_test, grid_predictions)

[23]: 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'}: 0.9775556904684072

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

The confusion Matrix:
 [[51  0]
 [ 3 79]]

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

The report:
              precision    recall  f1-score   support

       False       0.94      1.00      0.97        51
        True       1.00      0.96      0.98        82

    accuracy                           0.98       133
   macro avg       0.97      0.98      0.98       133
weighted avg       0.98      0.98      0.98       133

[ ]:
```

worked on logistic regression classification

Final model

Model	Accuracy	Precision (False)	Precision (True)	Recall (False)	Recall (True)	F1-score (False)	F1-score (True)
Random Forest	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SVM	0.97	0.93	1.00	1.00	0.95	0.96	0.97
Decision Tree	0.83	0.82	0.84	0.88	0.83	0.85	0.83
Logistic Regression	**0.98**	0.84	1.00	1.00	0.88	0.91	0.93

Logistic Regression achieved the highest overall accuracy of 0.98, indicating strong performance in classifying the target variable. logistic regression is recommended as the best model for this particular problem.