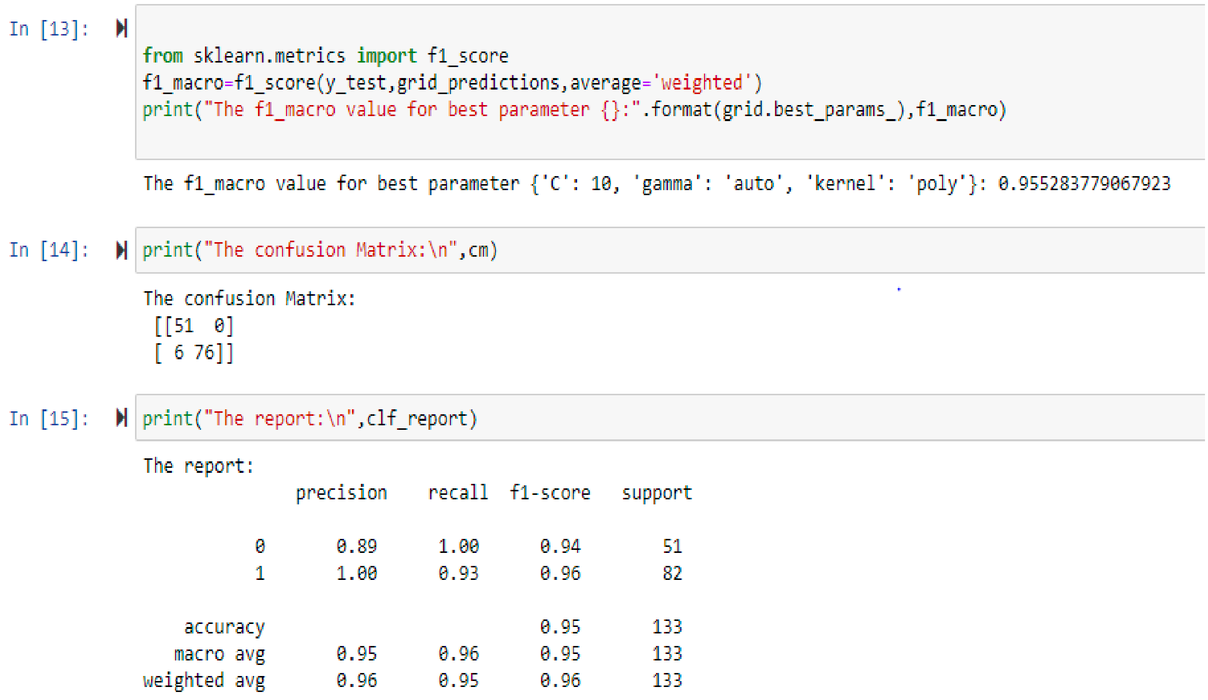
ASSIGNMENT- CLASSIFICATION

PROBLEM STATEMENT:

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.

SVM



```
In [13]:  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'}: 0.955283779067923

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

          The confusion Matrix:
           [[51  0]
           [ 6 76]]

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

          The report:
                        precision    recall  f1-score   support

                     0       0.89      1.00      0.94        51
                     1       1.00      0.93      0.96        82

              accuracy                           0.95       133
             macro avg       0.95      0.96      0.95       133
          weighted avg       0.96      0.95      0.96       133
```

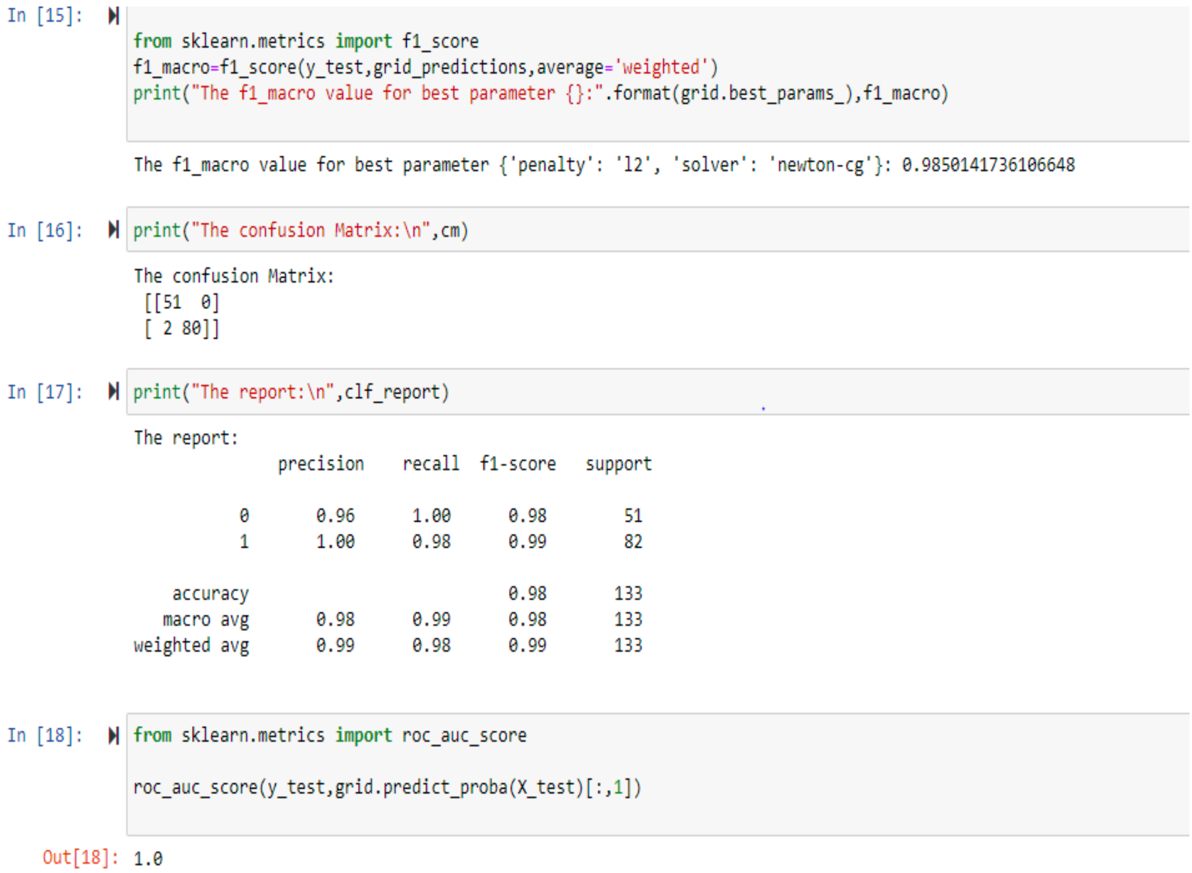
C Parameter	**Gamma**	**Kernel**	**F1 score**
10	**auto**	rbf	0.96
		poly	**0.95**
		sigmoid	0.95
		linear	0.94
	scale	rbf	0.96
		poly	0.96
		sigmoid	0.95

		linear	0.94
100	auto	rbf	0.94
		poly	0.95
		sigmoid	0.94
		linear	0.92
	scale	rbf	0.94
		poly	0.96
		sigmoid	0.94
		linear	0.92
1000	auto	rbf	0.94
		poly	0.91
		sigmoid	0.93
		linear	0.92
	scale	rbf	0.94
		poly	0.91
		sigmoid	0.93
		linear	0.92
2000	auto	rbf	0.94
		poly	0.91
		sigmoid	0.93
		linear	0.92
	scale	rbf	0.94
		poly	0.91
		sigmoid	0.93
		linear	0.92
3000	auto	rbf	0.94
		poly	0.91
		sigmoid	0.93
		linear	0.92
	scale	rbf	0.94
		poly	0.91
		sigmoid	0.93
		linear	0.92

In SVM Classification, when the parameter C=10, gamma=auto, kernel=poly, the F1 score is **0.95**.

LOGISTIC CLASSIFICATION

penalty	solver	F1 score
l2	newton-cg	0.98
	lbfgs	0.96
	liblinear	0.95
	saga	0.95

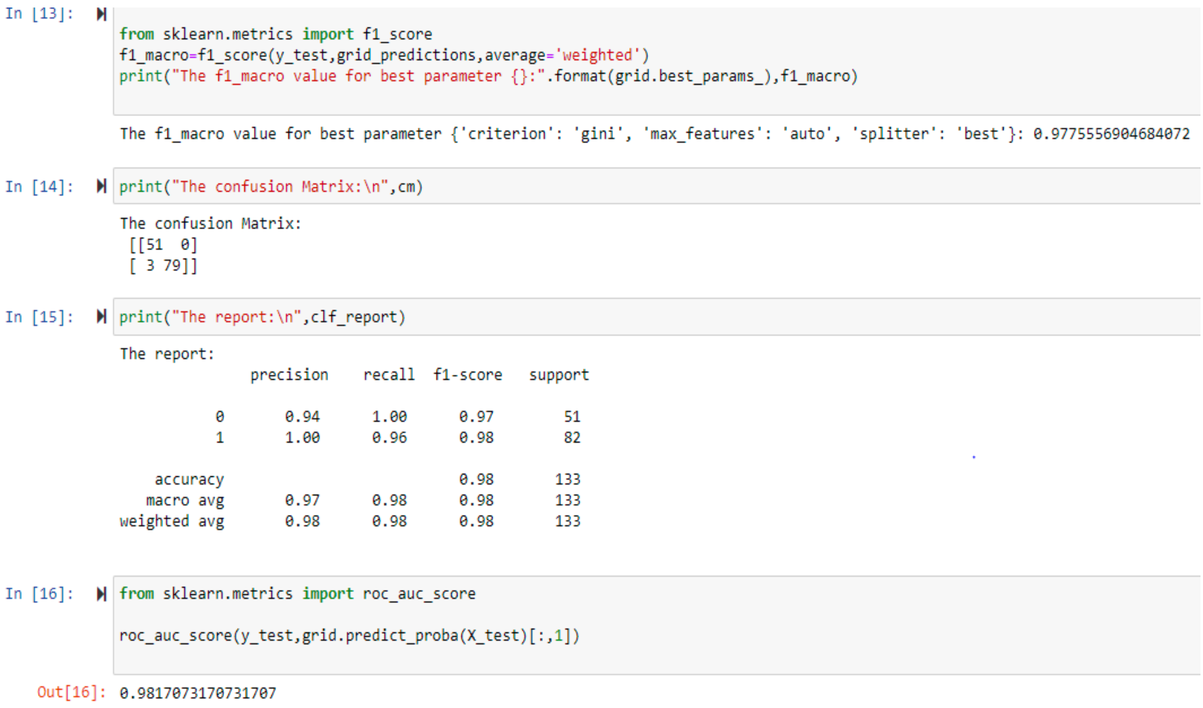


```
In [15]:  from sklearn.metrics import f1_score
          f1_macro=f1_score(y_test,grid_predictions,average='weighted')
          print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

          The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 0.9850141736106648

In [16]:  print("The confusion Matrix:\n",cm)

          The confusion Matrix:
           [[51  0]
           [ 2 80]]

In [17]:  print("The report:\n",clf_report)

          The report:
                        precision    recall  f1-score   support

                     0       0.96      1.00      0.98        51
                     1       1.00      0.98      0.99        82

              accuracy                           0.98       133
             macro avg       0.98      0.99      0.98       133
          weighted avg       0.99      0.98      0.99       133


In [18]:  from sklearn.metrics import roc_auc_score

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

 Out[18]: 1.0
```

In Logistic Classification, when the parameter penalty=l2, solver=newton-cg, the F1 score is **0.98**.

DECISION TREE

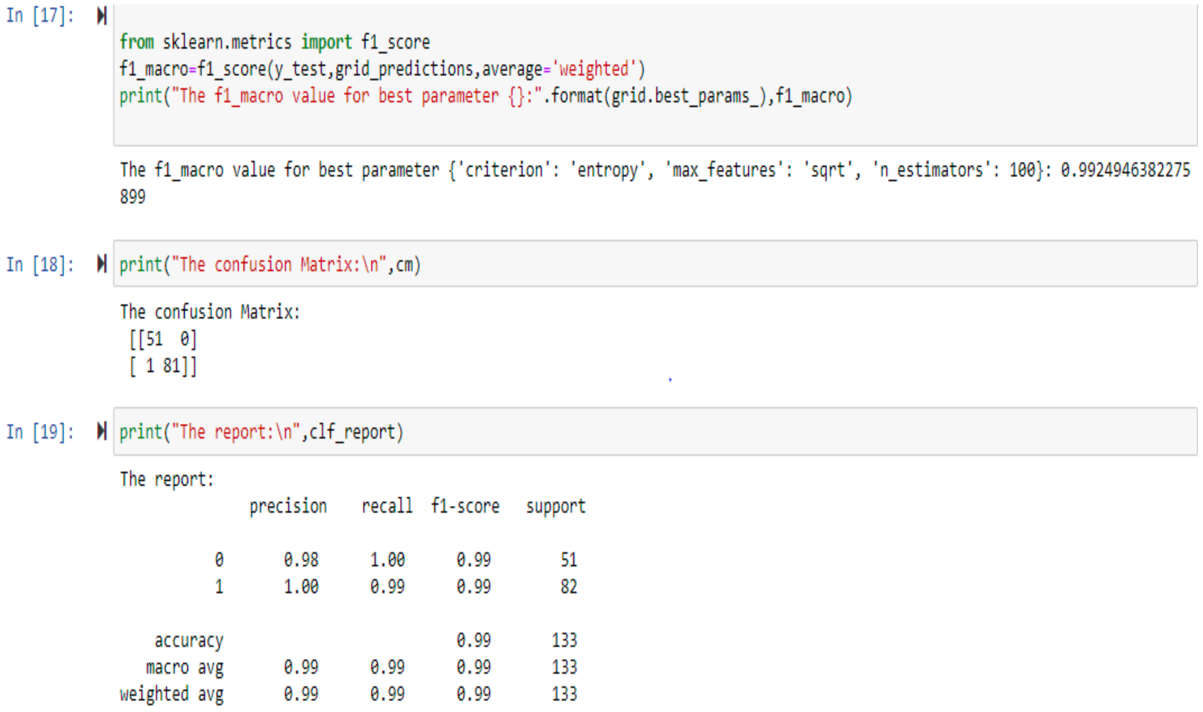
Criterion	Max_features	Splitter	F1 score
gini	auto	best	0.97
		random	0.94
	sqrt	best	0.95
		random	0.94
	log2	best	0.92
		random	0.94
entropy	auto	best	0.92
		random	0.93
	sqrt	best	0.92
		random	0.93
	log2	best	0.92
		random	0.94



```
In [13]:  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': 'auto', 'splitter': 'best'}: 0.9775556904684072

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

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

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

           The report:
                          precision    recall  f1-score   support

                      0       0.94      1.00      0.97        51
                      1       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

In [16]:  from sklearn.metrics import roc_auc_score

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

  Out[16]: 0.9817073170731707
```

In Decision Tree Classification, when the parameter criterion=gini, max_features=auto, splitter=best, the F1 score is **0.97**.

RANDOM FOREST



```
In [17]:  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': 'entropy', 'max_features': 'sqrt', 'n_estimators': 100}: 0.9924946382275
          899

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

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

In [19]:  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
```

Criterion	Max_features	Splitter	F1 score
gini	auto	10	0.95
		100	0.98
	sqrt	10	0.96
		100	0.98
	log2	10	0.96
		100	0.98
entropy	auto	10	0.95
		100	0.98
	sqrt	10	0.97
		100	**0.99**
	log2	10	0.97
		100	0.97

In Random Forest Classification, when the parameter criterion=entropy, max_features=sqrt, splitter=100, the F1 score is **0.99**.

FINAL MODEL:

RANDOM FOREST is the best model to predict the person does have chronic kidney disease(CKD).

RANDOM FOREST = 0.99