PROBLEM STATEMENT

As Data Scientists, we must develop a model to predict/classify whether the patient has chronic kidney disease or not based on the several parameters given on the Dataset.

Three stages of Problem Identification:

Stage 1- Domain Selection- Machine learning

As the input (CKD Dataset) contains mostly numerical values, we can choose the ***Machine Learning Domain***

Stage 2-Learning – Supervised Learning

As the Requirement is clear (predict patient has disease or not) we can choose ***Supervised Learning***

Stage 3 – Classification:

Prediction related whether the patient has disease or not (yes or no) so we can choose the ***Classification type***

DATASET INFORMATION:

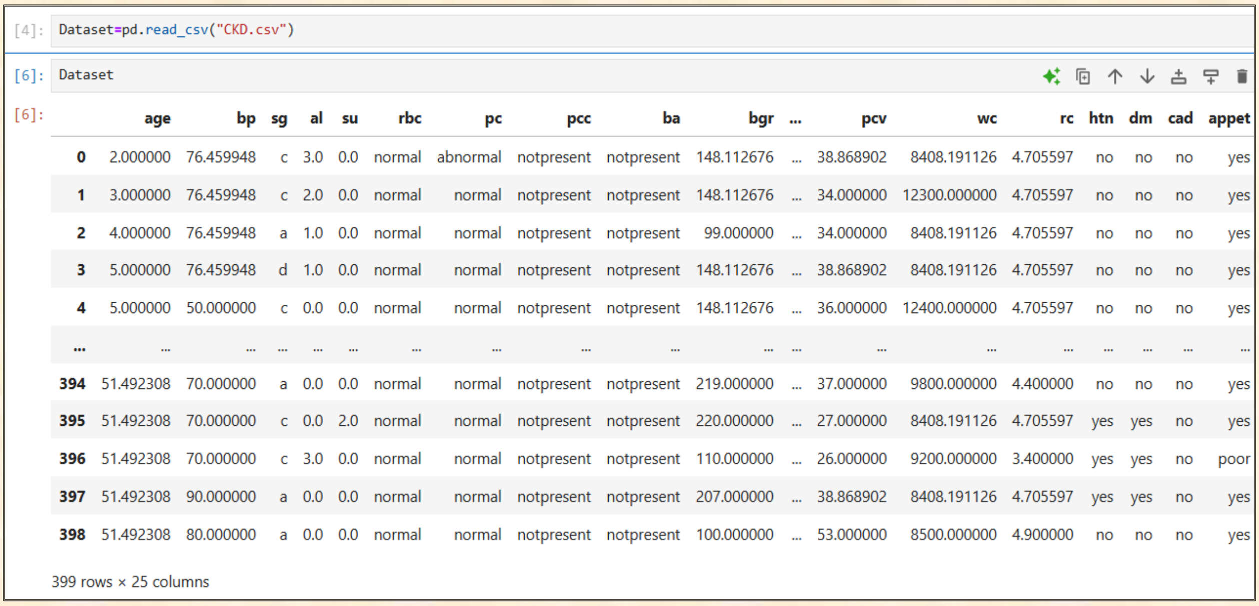
Total No of rows: 399

Total No of columns:25

Column Name: 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu' 'sc', 'sod', 'pot', 'hrmo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane', 'classification'

Input Variables: 'age', 'bp', 'sg', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu' 'sc', 'sod', 'pot', 'hrmo', 'pcv', 'wc', 'rc', 'htn', 'dm', 'cad', 'appet', 'pe', 'ane'

Output Variable: 'classification'



```
[4]: Dataset=pd.read_csv("CKD.csv")
```

```
[6]: Dataset
```

[6]:

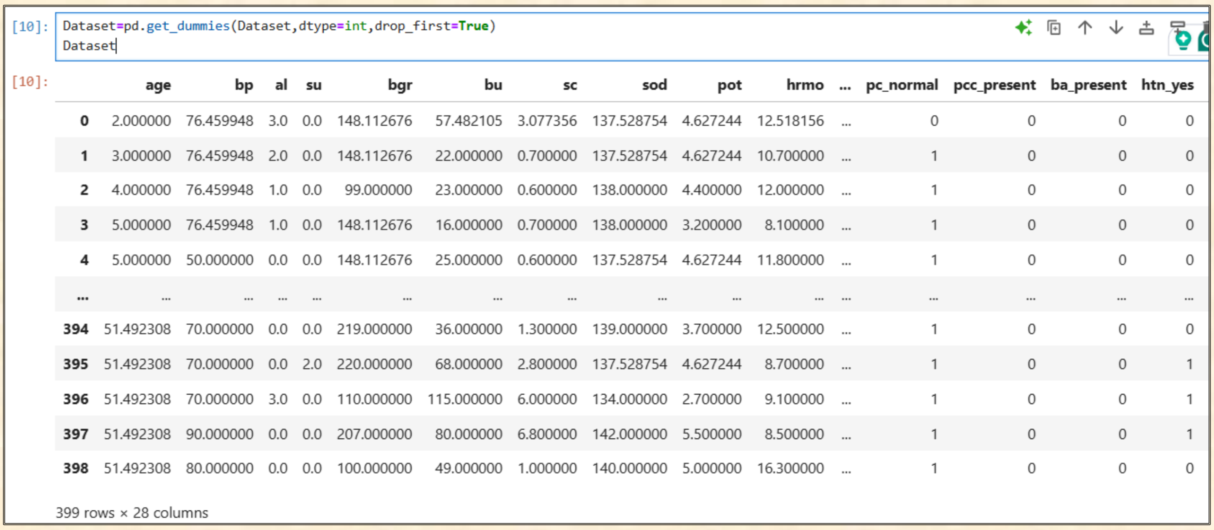
	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet
0	2.000000	76.459948	c	3.0	0.0	normal	abnormal	notpresent	notpresent	148.112676	...	38.868902	8408.191126	4.705597	no	no	no	yes
1	3.000000	76.459948	c	2.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	34.000000	12300.000000	4.705597	no	no	no	yes
2	4.000000	76.459948	a	1.0	0.0	normal	normal	notpresent	notpresent	99.000000	...	34.000000	8408.191126	4.705597	no	no	no	yes
3	5.000000	76.459948	d	1.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	38.868902	8408.191126	4.705597	no	no	no	yes
4	5.000000	50.000000	c	0.0	0.0	normal	normal	notpresent	notpresent	148.112676	...	36.000000	12400.000000	4.705597	no	no	no	yes
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
394	51.492308	70.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	219.000000	...	37.000000	9800.000000	4.400000	no	no	no	yes
395	51.492308	70.000000	c	0.0	2.0	normal	normal	notpresent	notpresent	220.000000	...	27.000000	8408.191126	4.705597	yes	yes	no	yes
396	51.492308	70.000000	c	3.0	0.0	normal	normal	notpresent	notpresent	110.000000	...	26.000000	9200.000000	3.400000	yes	yes	no	poor
397	51.492308	90.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	207.000000	...	38.868902	8408.191126	4.705597	yes	yes	no	yes
398	51.492308	80.000000	a	0.0	0.0	normal	normal	notpresent	notpresent	100.000000	...	53.000000	8500.000000	4.900000	no	no	no	yes

399 rows × 25 columns

Preprocessing Method:

Since the Dataset has many categorical columns like rbc, pc, pcc, ba, bu,..etc, it should be converted to numbers

Here the categorical data is nominal, so one hot encoding method is used to convert the categorical data (string) to numerical data (numbers 1 or 0)

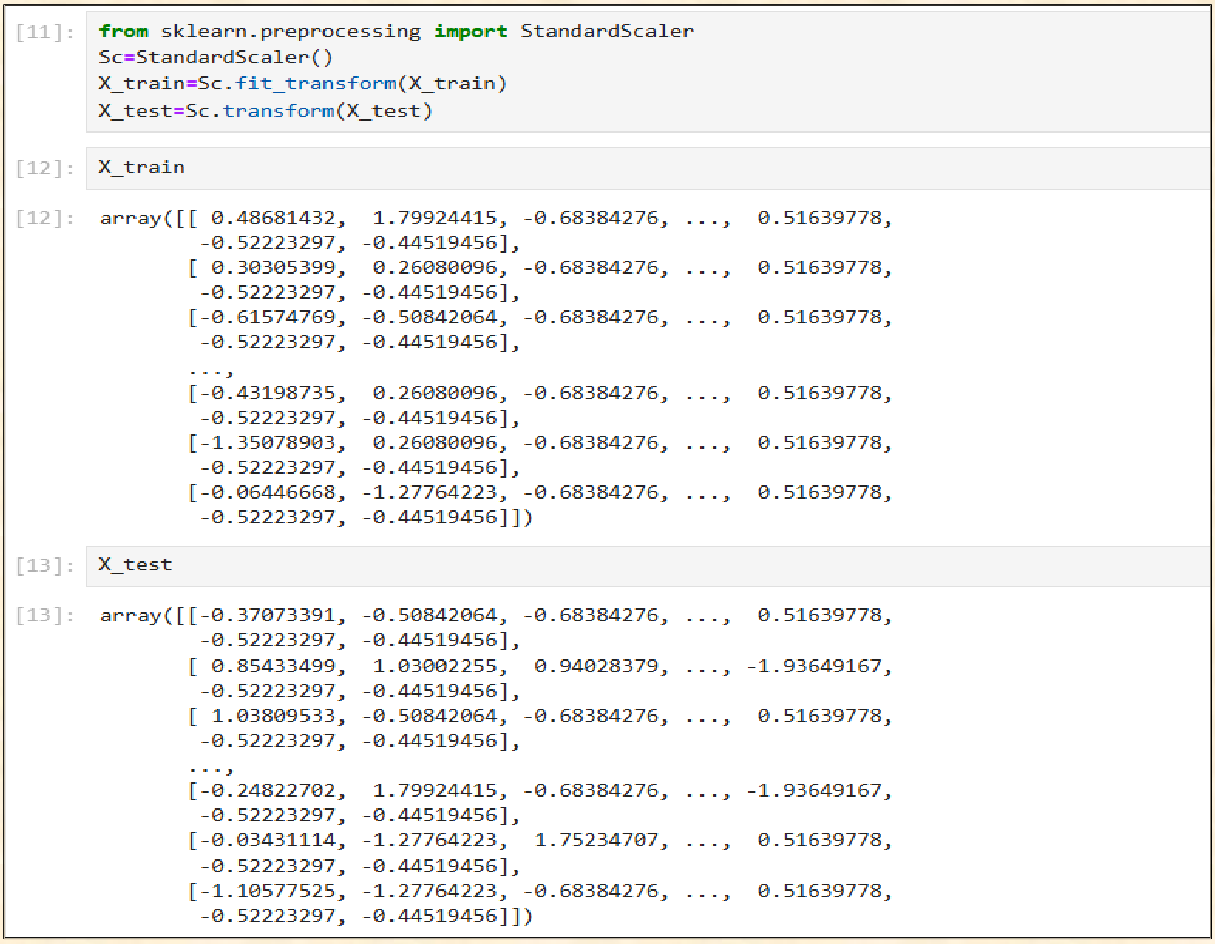


```
[10]: Dataset=pd.get_dummies(Dataset,dtype=int,drop_first=True)
      Dataset
```

[10]:

	age	bp	al	su	bgr	bu	sc	sod	pot	hrmo	...	pc_normal	pcc_present	ba_present	htn_yes
0	2.000000	76.459948	3.0	0.0	148.112676	57.482105	3.077356	137.528754	4.627244	12.518156	...	0	0	0	0
1	3.000000	76.459948	2.0	0.0	148.112676	22.000000	0.700000	137.528754	4.627244	10.700000	...	1	0	0	0
2	4.000000	76.459948	1.0	0.0	99.000000	23.000000	0.600000	138.000000	4.400000	12.000000	...	1	0	0	0
3	5.000000	76.459948	1.0	0.0	148.112676	16.000000	0.700000	138.000000	3.200000	8.100000	...	1	0	0	0
4	5.000000	50.000000	0.0	0.0	148.112676	25.000000	0.600000	137.528754	4.627244	11.800000	...	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
394	51.492308	70.000000	0.0	0.0	219.000000	36.000000	1.300000	139.000000	3.700000	12.500000	...	1	0	0	0
395	51.492308	70.000000	0.0	2.0	220.000000	68.000000	2.800000	137.528754	4.627244	8.700000	...	1	0	0	1
396	51.492308	70.000000	3.0	0.0	110.000000	115.000000	6.000000	134.000000	2.700000	9.100000	...	1	0	0	1
397	51.492308	90.000000	0.0	0.0	207.000000	80.000000	6.800000	142.000000	5.500000	8.500000	...	1	0	0	1
398	51.492308	80.000000	0.0	0.0	100.000000	49.000000	1.000000	140.000000	5.000000	16.300000	...	1	0	0	0

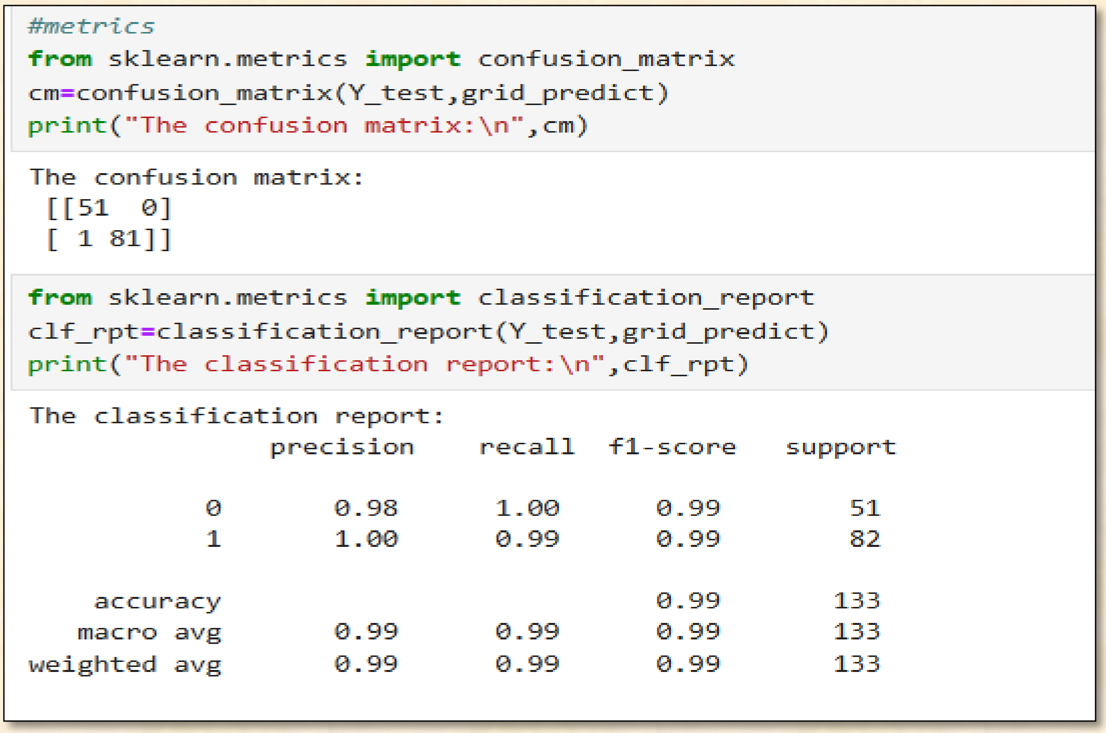
399 rows × 28 columns



```
[11]: from sklearn.preprocessing import StandardScaler
      Sc=StandardScaler()
      X_train=Sc.fit_transform(X_train)
      X_test=Sc.transform(X_test)

[12]: X_train

[12]: array([[ 0.48681432,  1.79924415, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456],
             [ 0.30305399,  0.26080096, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456],
             [-0.61574769, -0.50842064, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456],
             ...,
             [-0.43198735,  0.26080096, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456],
             [-1.35078903,  0.26080096, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456],
             [-0.06446668, -1.27764223, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456]])

[13]: X_test

[13]: array([[-0.37073391, -0.50842064, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456],
             [ 0.85433499,  1.03002255,  0.94028379, ..., -1.93649167,
              -0.52223297, -0.44519456],
             [ 1.03809533, -0.50842064, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456],
             ...,
             [-0.24822702,  1.79924415, -0.68384276, ..., -1.93649167,
              -0.52223297, -0.44519456],
             [-0.03431114, -1.27764223,  1.75234707, ...,  0.51639778,
              -0.52223297, -0.44519456],
             [-1.10577525, -1.27764223, -0.68384276, ...,  0.51639778,
              -0.52223297, -0.44519456]])
```

1. SUPPORT VECTOR MACHINE CLASSIFIER

Confusion matrix and classification report of Support Vector Machine model is given below



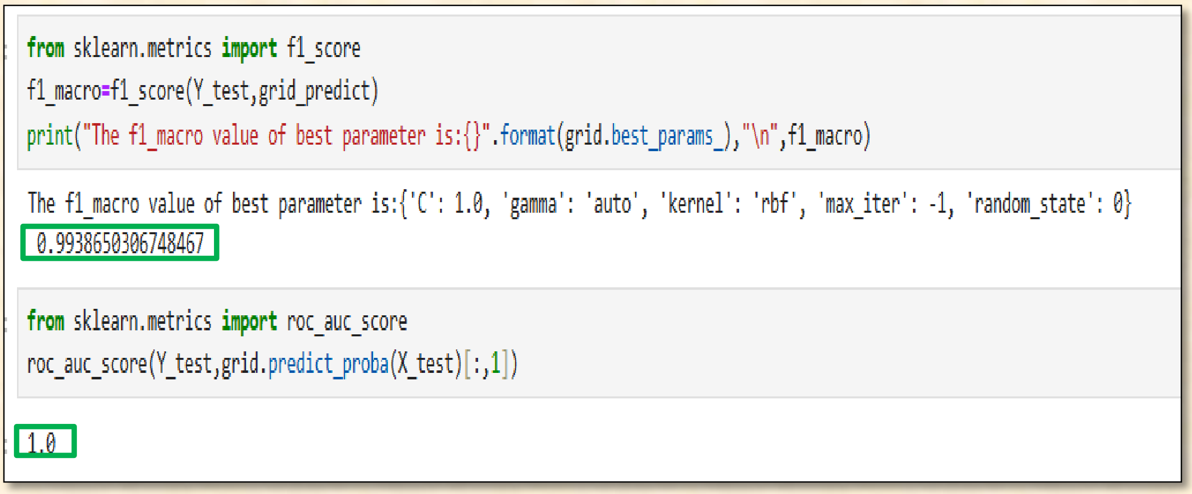
```
#metrics
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predict)
print("The confusion matrix:\n",cm)
```

```
The confusion matrix:
 [[51  0]
 [ 1 81]]
```

```
from sklearn.metrics import classification_report
clf_rpt=classification_report(Y_test,grid_predict)
print("The classification report:\n",clf_rpt)
```

```
The classification 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
```

f1_macro and roc_auc_score of Support Vector Machine model is given below



```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,grid_predict)
print("The f1_macro value of best parameter is:{}".format(grid.best_params_),"\n",f1_macro)
```

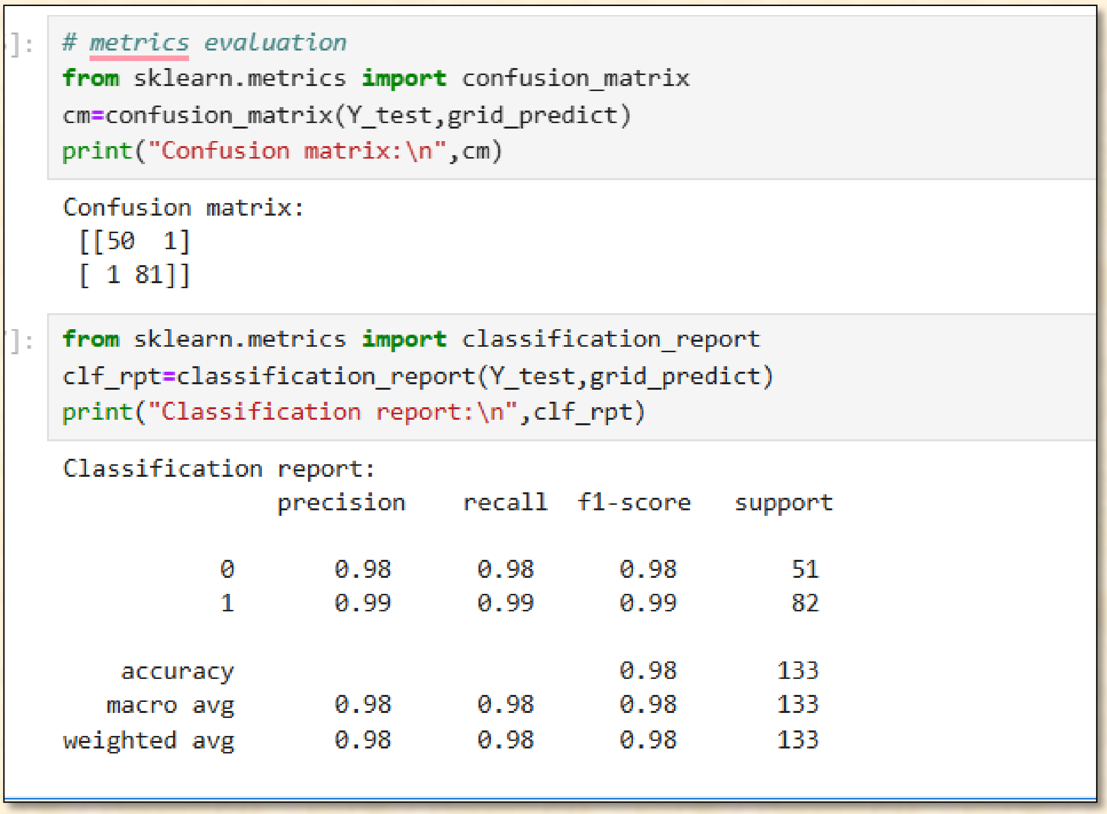
```
The f1_macro value of best parameter is:{'C': 1.0, 'gamma': 'auto', 'kernel': 'rbf', 'max_iter': -1, 'random_state': 0}
 0.9938650306748467
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

```
1.0
```

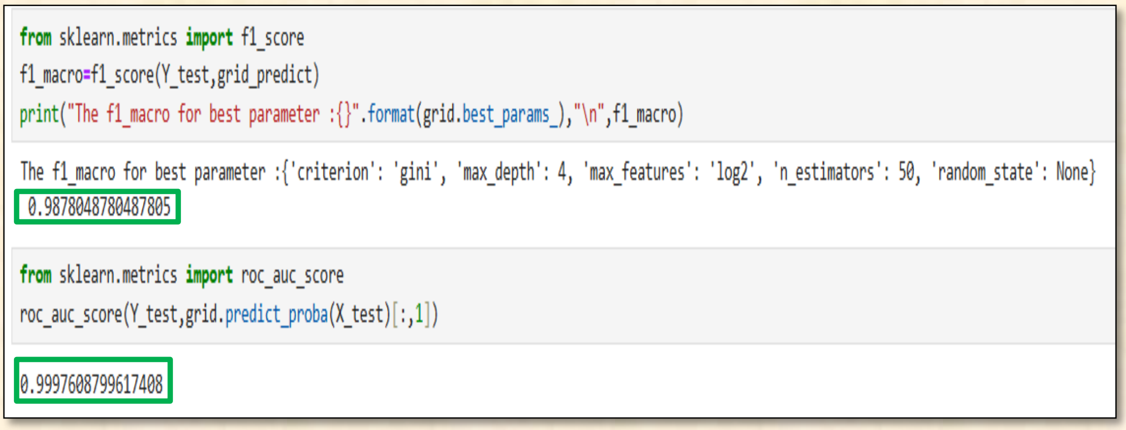
2. RANDOM FOREST CLASSIFIER:

Confusion matrix and classification report of RandomForestTree is given below



```
]: # metrics evaluation
   from sklearn.metrics import confusion_matrix
   cm=confusion_matrix(Y_test,grid_predict)
   print("Confusion matrix:\n",cm)

   Confusion matrix:
    [[50  1]
    [ 1 81]]

]: from sklearn.metrics import classification_report
   clf_rpt=classification_report(Y_test,grid_predict)
   print("Classification report:\n",clf_rpt)

   Classification report:
                  precision    recall  f1-score   support

              0       0.98      0.98      0.98        51
              1       0.99      0.99      0.99        82

       accuracy                           0.98       133
      macro avg       0.98      0.98      0.98       133
   weighted avg       0.98      0.98      0.98       133
```

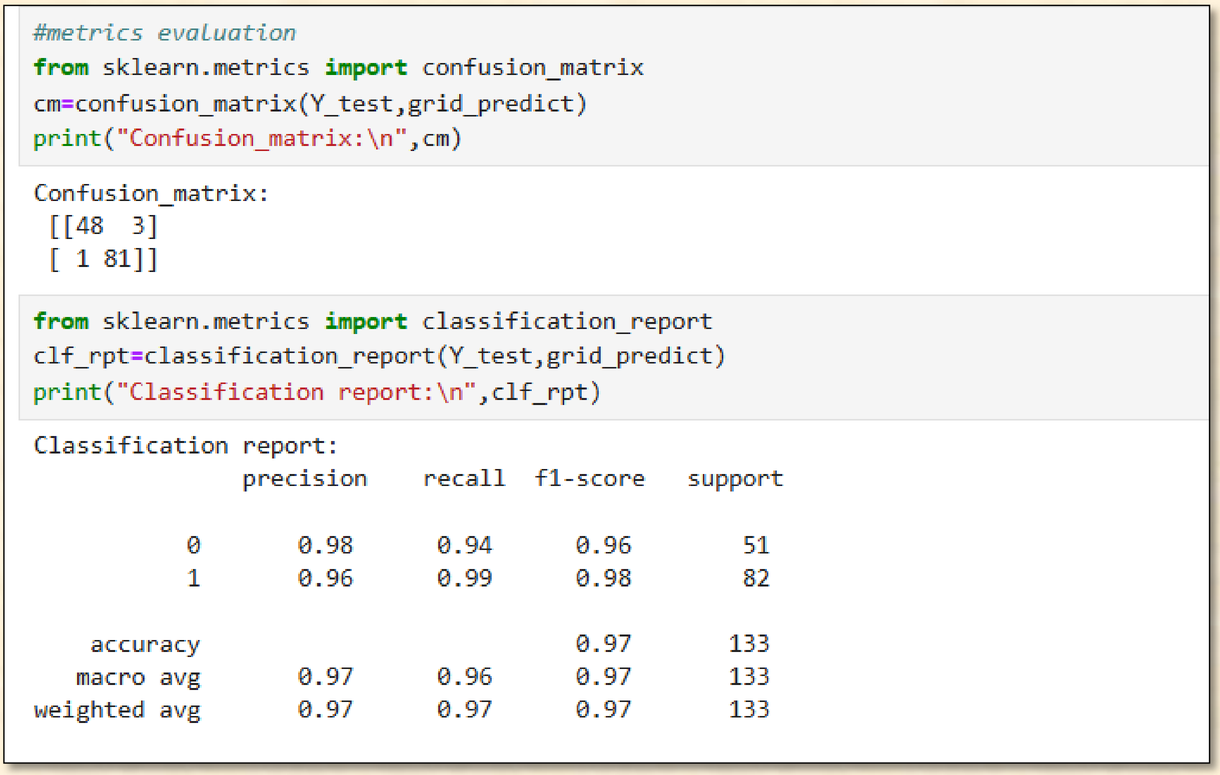
f1_macro and roc_auc_score of RandomForestTree model is given below



```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,grid_predict)
print("The f1_macro for best parameter :{}".format(grid.best_params_),"\n",f1_macro)

The f1_macro for best parameter :{'criterion': 'gini', 'max_depth': 4, 'max_features': 'log2', 'n_estimators': 50, 'random_state': None}
 0.9878048780487805

from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])

0.9997608799617408
```

3. DECISION TREE CLASSIFIER:

Confusion matrix and classification report of DecisionTree is given below



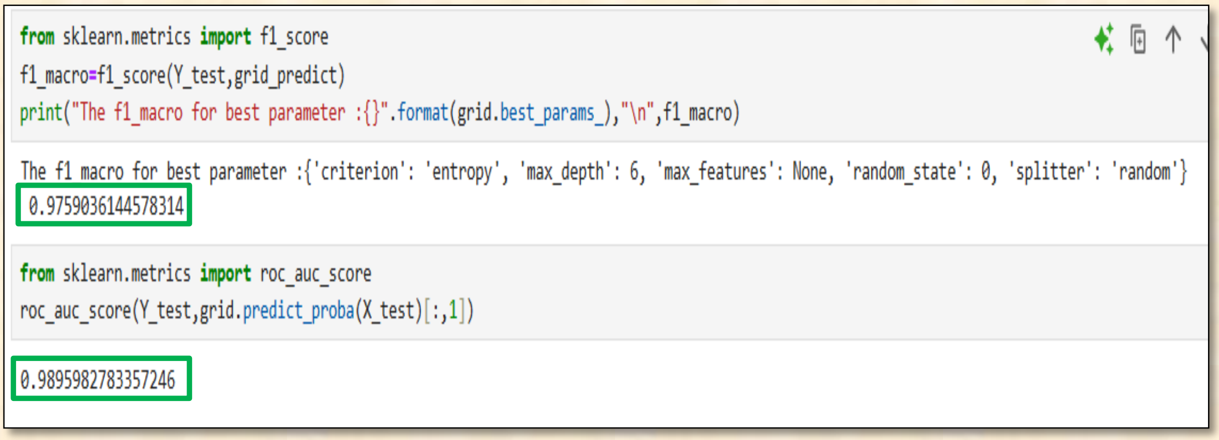
```
#metrics evaluation
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predict)
print("Confusion_matrix:\n",cm)
```

```
Confusion_matrix:
 [[48  3]
 [ 1 81]]
```

```
from sklearn.metrics import classification_report
clf_rpt=classification_report(Y_test,grid_predict)
print("Classification report:\n",clf_rpt)
```

```
Classification report:
               precision    recall  f1-score   support

           0       0.98      0.94      0.96        51
           1       0.96      0.99      0.98        82

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

f1_macro and roc_auc_score of DecisionTree model is given below



```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,grid_predict)
print("The f1_macro for best parameter :{}".format(grid.best_params_),"\n",f1_macro)
```

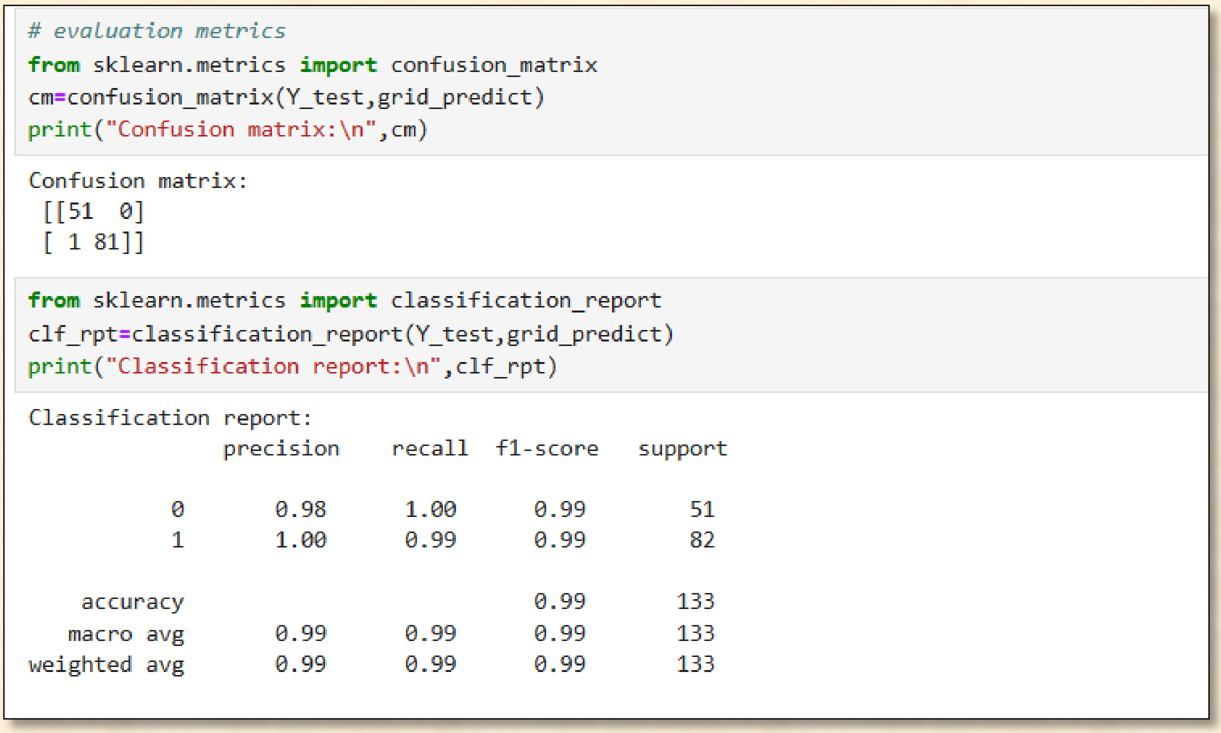
```
The f1_macro for best parameter :{'criterion': 'entropy', 'max_depth': 6, 'max_features': None, 'random_state': 0, 'splitter': 'random'}
0.9759036144578314
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

```
0.9895982783357246
```

4. LOGISTIC REGRESSION

Confusion matrix and classification report of Logistic Regression is given below



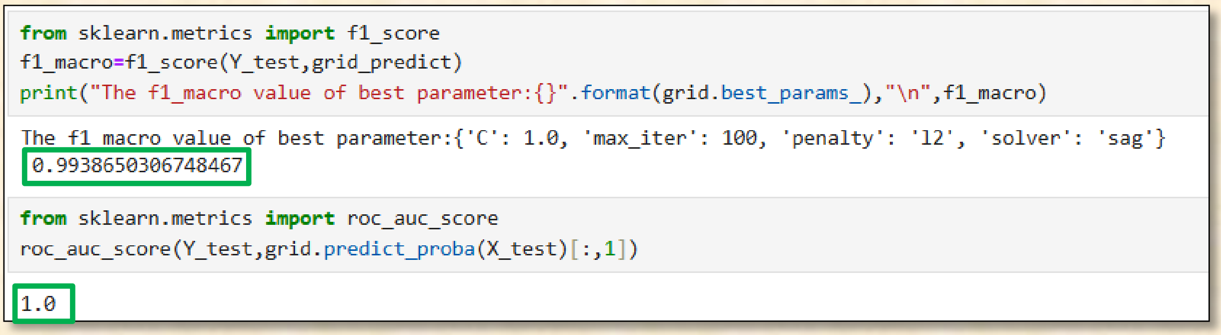
```
# evaluation metrics
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predict)
print("Confusion matrix:\n",cm)
```

```
Confusion matrix:
 [[51  0]
 [ 1 81]]
```

```
from sklearn.metrics import classification_report
clf_rpt=classification_report(Y_test,grid_predict)
print("Classification report:\n",clf_rpt)
```

```
Classification 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
```

f1_macro and roc_auc_score of Logistic Regression model is given below



```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,grid_predict)
print("The f1_macro value of best parameter:{}".format(grid.best_params_),"\n",f1_macro)
```

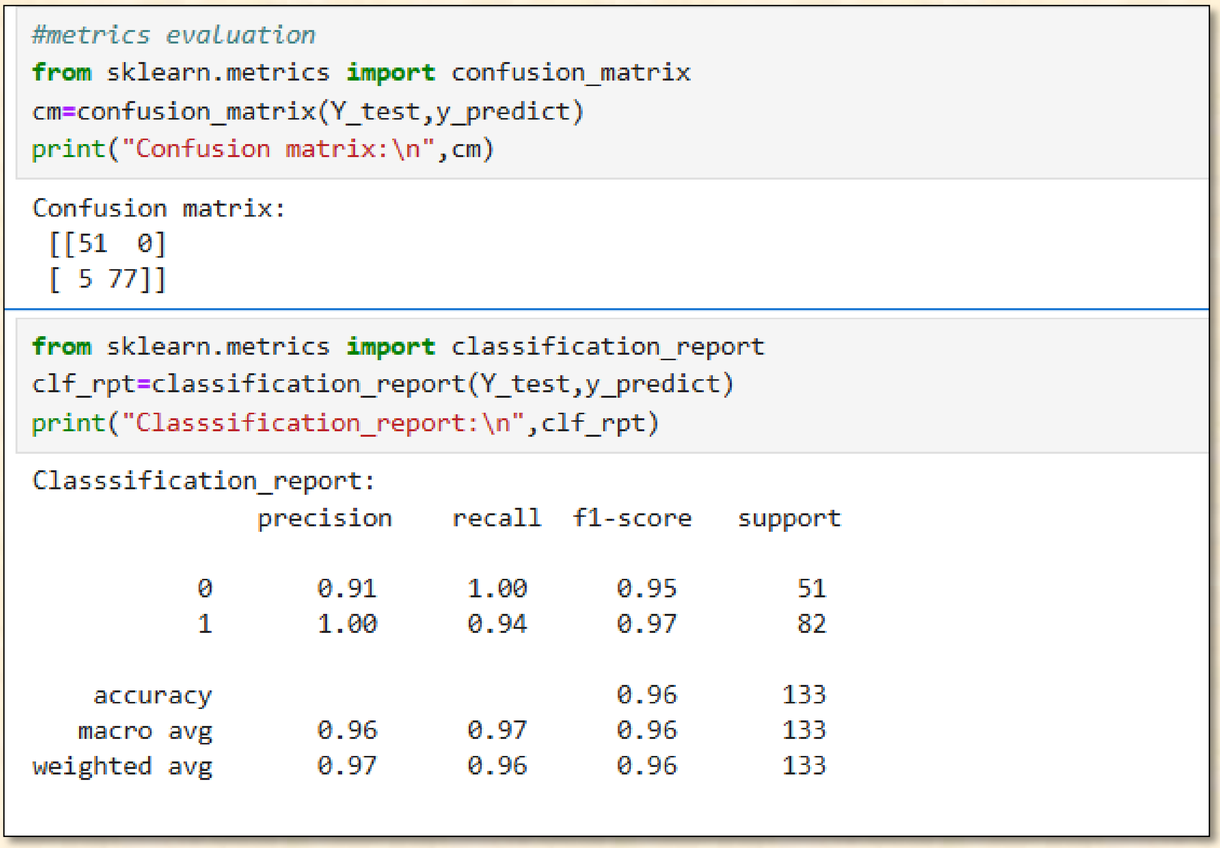
```
The f1_macro value of best parameter:{'C': 1.0, 'max_iter': 100, 'penalty': 'l2', 'solver': 'sag'}
0.9938650306748467
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

```
1.0
```

5. KNEIGHBOR CLASSIFIER:

Confusion matrix and classification report of KNeighbor Classifier is given below



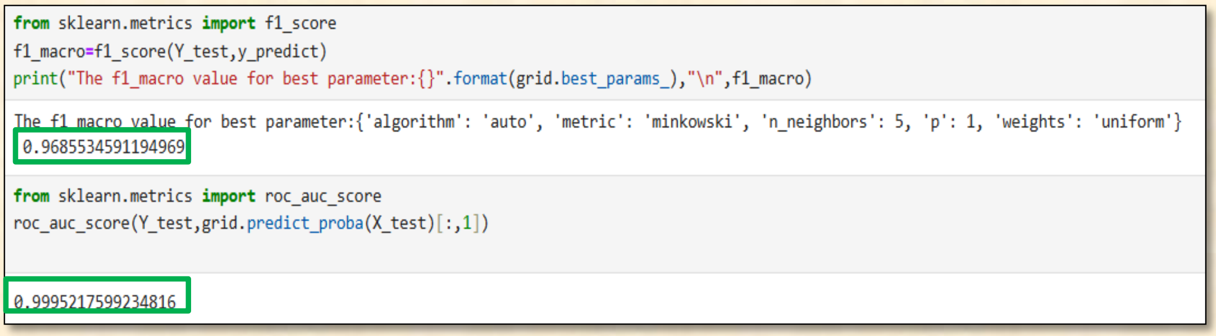
```
#metrics evaluation
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,y_predict)
print("Confusion matrix:\n",cm)
```

```
Confusion matrix:
 [[51  0]
 [ 5 77]]
```

```
from sklearn.metrics import classification_report
clf_rpt=classification_report(Y_test,y_predict)
print("Classsification_report:\n",clf_rpt)
```

```
Classsification_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
```

f1_macro and roc_auc_score of KNeigbor Classifier model is given below



```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,y_predict)
print("The f1_macro value for best parameter:{}".format(grid.best_params_),"\n",f1_macro)
```

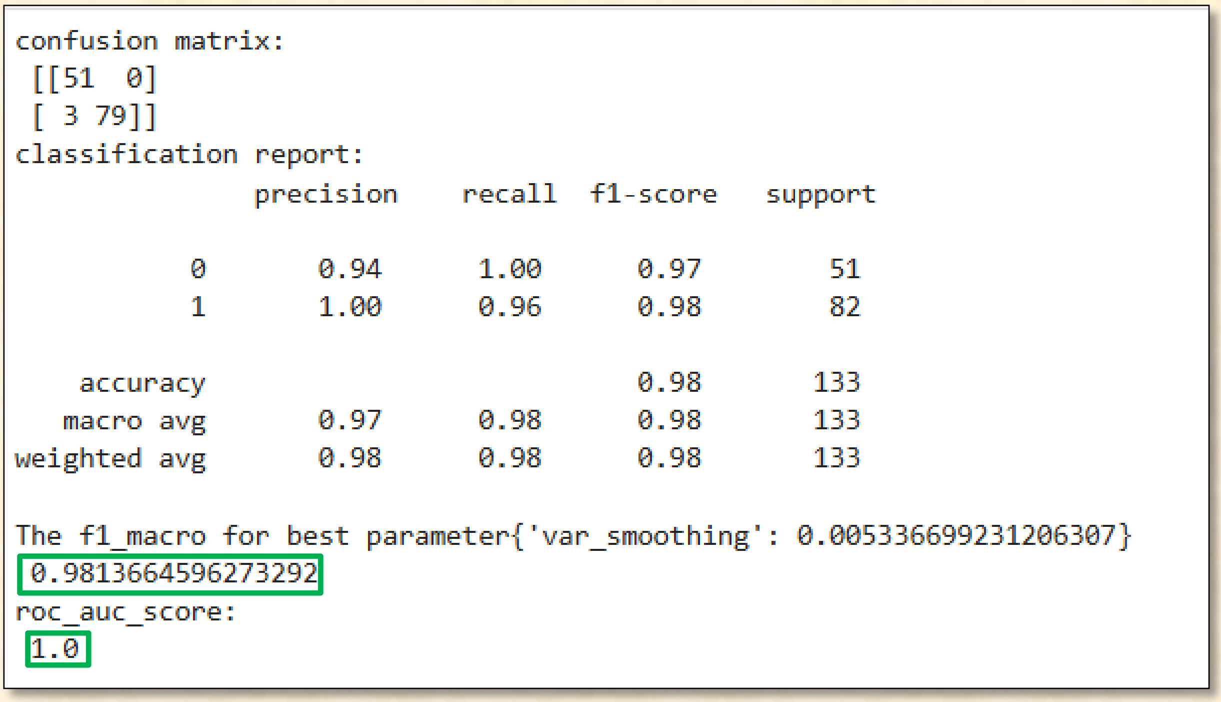
```
The f1_macro value for best parameter:{'algorithm': 'auto', 'metric': 'minkowski', 'n_neighbors': 5, 'p': 1, 'weights': 'uniform'}
 0.9685534591194969
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

```
0.9995217599234816
```

6a. NAIVE BAYES - GAUSSIANNB CLASSIFIER

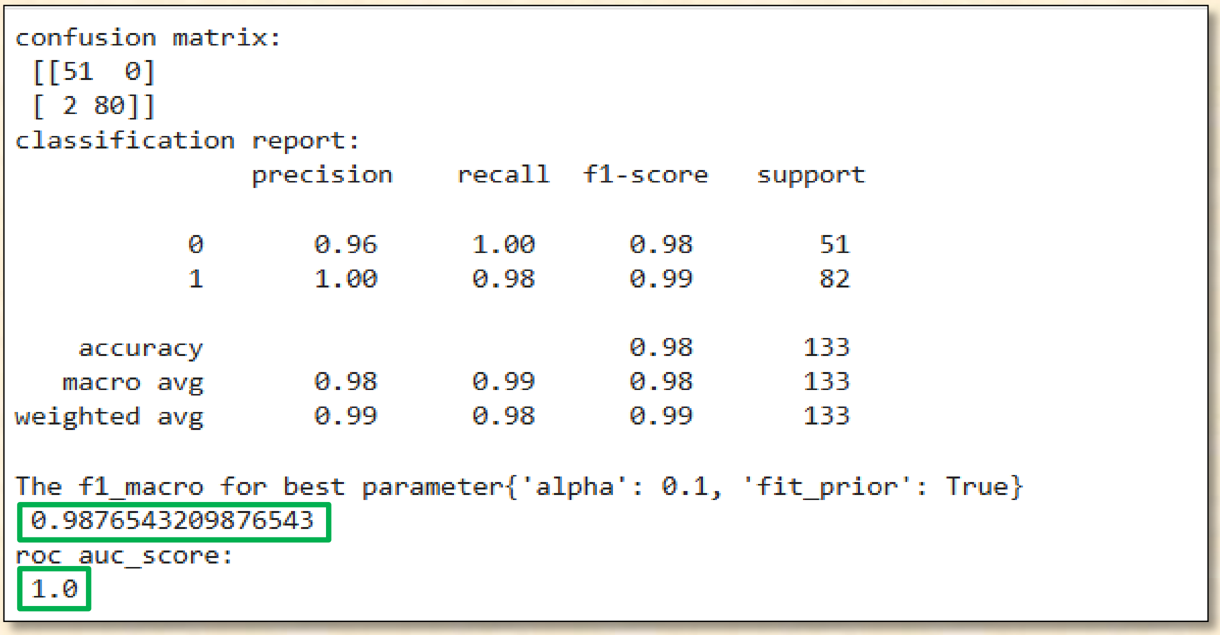
Confusion matrix,classification report,f1_macro and roc_auc_score of GaussianNB Classifier is given below



```
confusion matrix:
 [[51  0]
 [ 3 79]]
classification 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

The f1_macro for best parameter{'var_smoothing': 0.005336699231206307}
 0.9813664596273292
roc_auc_score:
 1.0
```

6b. NAIVE BAYES - MULTINOMIALNB CLASSIFIER

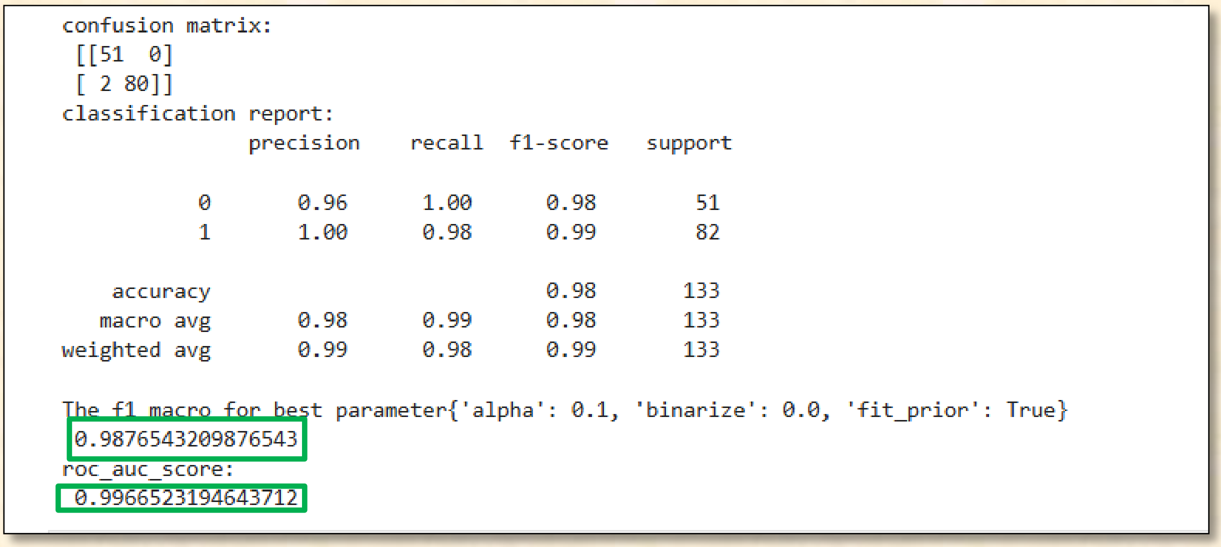
Confusion matrix,classification report,f1_macro and roc_auc_score of GaussianNB Classifier is given below



```
confusion matrix:
 [[51  0]
 [ 2 80]]
classification 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

The f1_macro for best parameter{'alpha': 0.1, 'fit_prior': True}
 0.9876543209876543
roc_auc_score:
 1.0
```

6c. NAIVE BAYES - BERNOULLINB CLASSIFIER

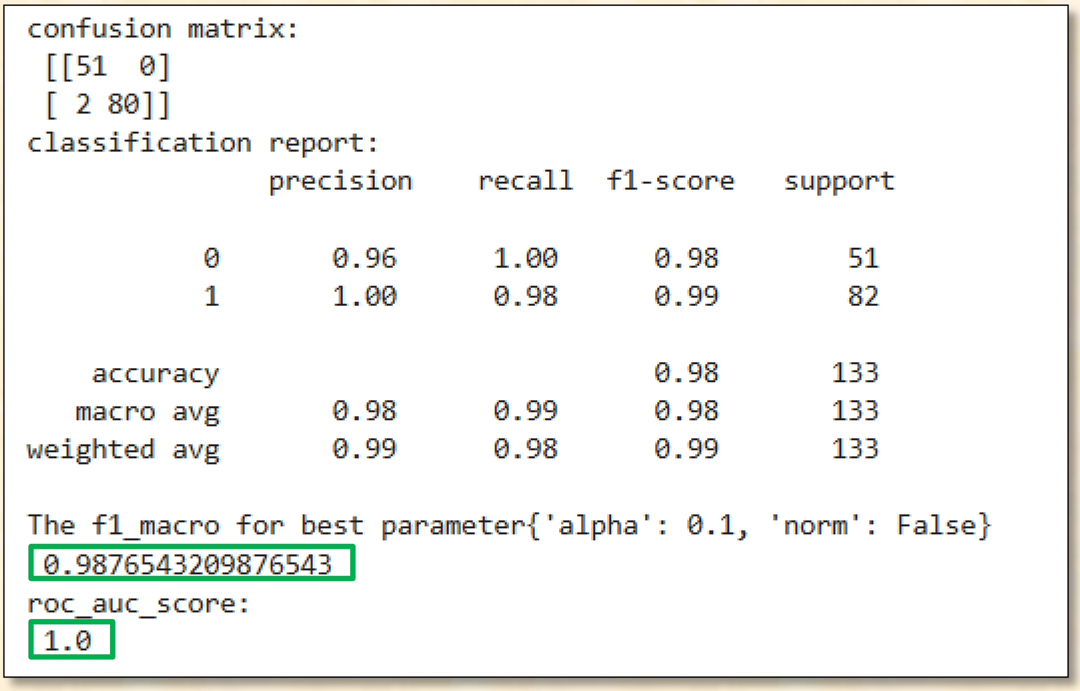
Confusion matrix,classification report,f1_macro and roc_auc_score of BernoulliNB Classifier is given below



```
confusion matrix:
 [[51  0]
 [ 2 80]]
classification 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

The f1_macro for best parameter{'alpha': 0.1, 'binarize': 0.0, 'fit_prior': True}
 0.9876543209876543
roc_auc_score:
 0.9966523194643712
```

6d. NAIVE BAYES – COMPLEMENTNB CLASSIFIER

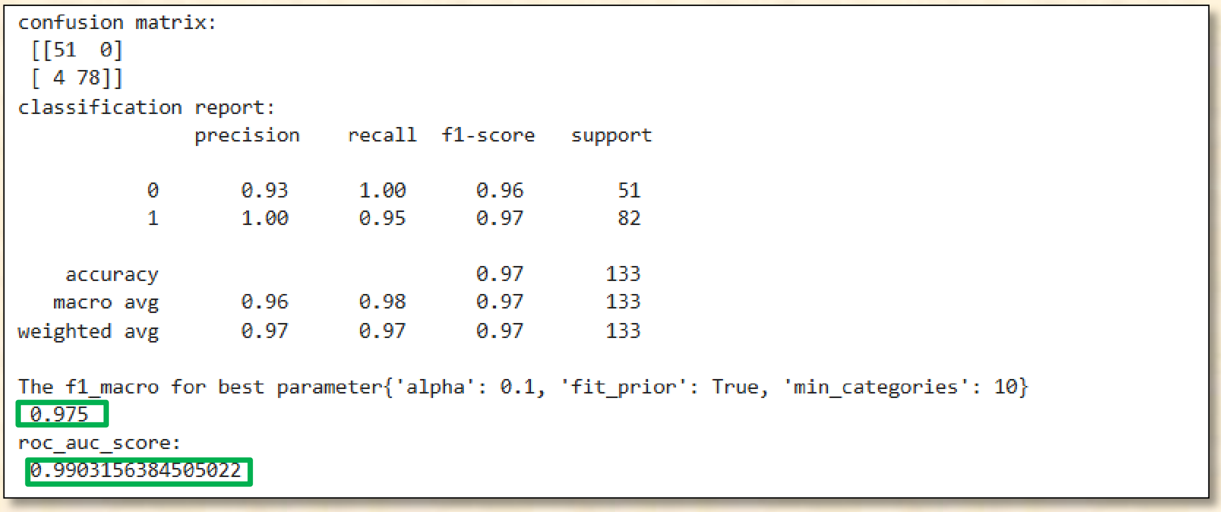
Confusion matrix,classification report,f1_macro and roc_auc_score of ComplementNB Classifier is given below



```
confusion matrix:
 [[51  0]
 [ 2 80]]
classification 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

The f1_macro for best parameter{'alpha': 0.1, 'norm': False}
 0.9876543209876543
roc_auc_score:
 1.0
```

6e. NAIVE BAYES – CATEGORICALNB CLASSIFIER

Confusion matrix,classification report,f1_macro and roc_auc_score of CategoricalNB Classifier is given below



```
confusion matrix:
 [[51  0]
 [ 4 78]]
classification report:
               precision    recall  f1-score   support

           0       0.93      1.00      0.96        51
           1       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

The f1_macro for best parameter{'alpha': 0.1, 'fit_prior': True, 'min_categories': 10}
 0.975
roc_auc_score:
 0.9903156384505022
```

Best models for the CKD dataset are **Support Vector Machine and Logistic regression** with **f1 macro – 99.38** and and **roc_auc_score =1.0**