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:*

They provide dataset in excel sheet. So we will take machine learning. After requirement is clear. Input and output are present here. So we will take Supervised learning. Then out put are Categorical value so we take Classification.

2.) *Tell basic info about the dataset (Total number of rows, columns):*

The dataset in 28 column and 399 Rows

3.) *Mention the pre-processing method if you're doing any (like converting string to number-nominal data)*

Dataset in Input 10 column are categorical values 1. rbc_normal *2.* pc_normal 3. pcc_present 4. ba_present 5. htn_yes 6. dm_yes 7. cad_yes 8. appet_yes 9. pe_yes 10. ane_yes

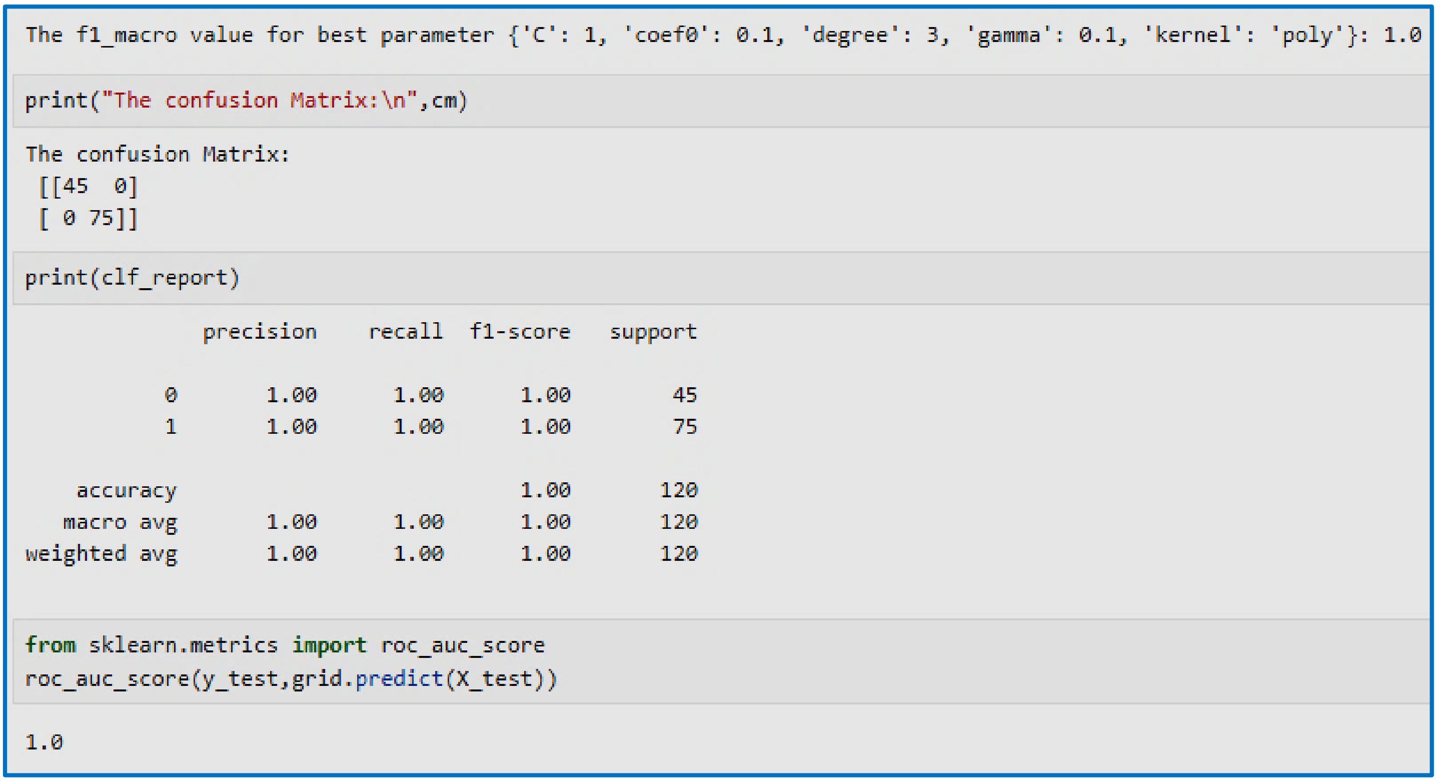
Out put 1 column categorical value 1. classification_yes

All columns are yes/no So I take one hot encoding .

dataset=pd.get_dummies(dataset,drop_first=True)

4.) *Develop a good model with good evaluation metric you can use any machine learning algorithm: you can create many models. Finally, you have to come up with final model.*

SUPPORT VECTOR MACHINE



```
The f1_macro value for best parameter {'C': 1, 'coef0': 0.1, 'degree': 3, 'gamma': 0.1, 'kernel': 'poly'}: 1.0
```

```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 0 75]]
```

```
print(clf_report)
```

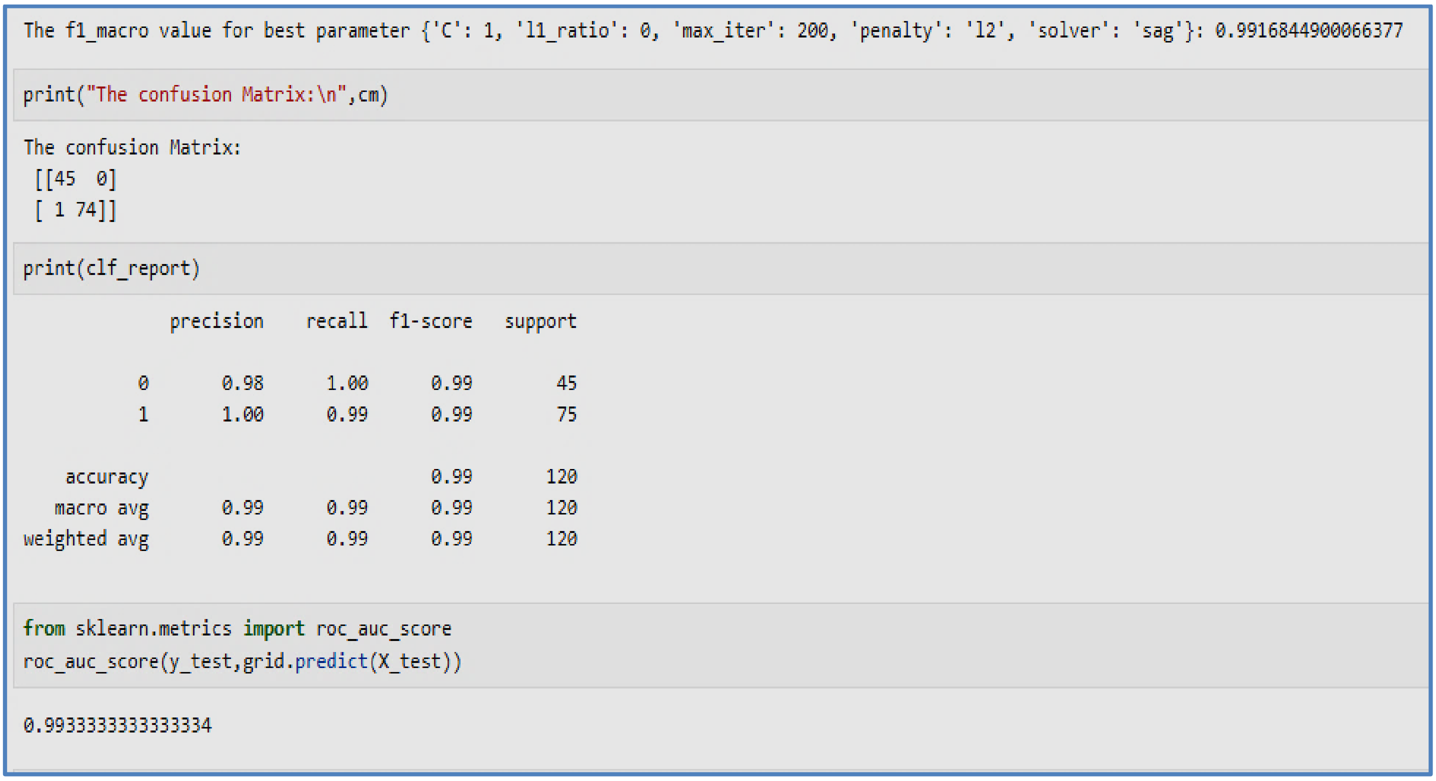
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        75

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict(X_test))
```

```
1.0
```

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

1. LOGISTICS REGRESSION



```
The f1_macro value for best parameter {'C': 1, 'l1_ratio': 0, 'max_iter': 200, 'penalty': 'l2', 'solver': 'sag'}: 0.9916844900066377
```

```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 1 74]]
```

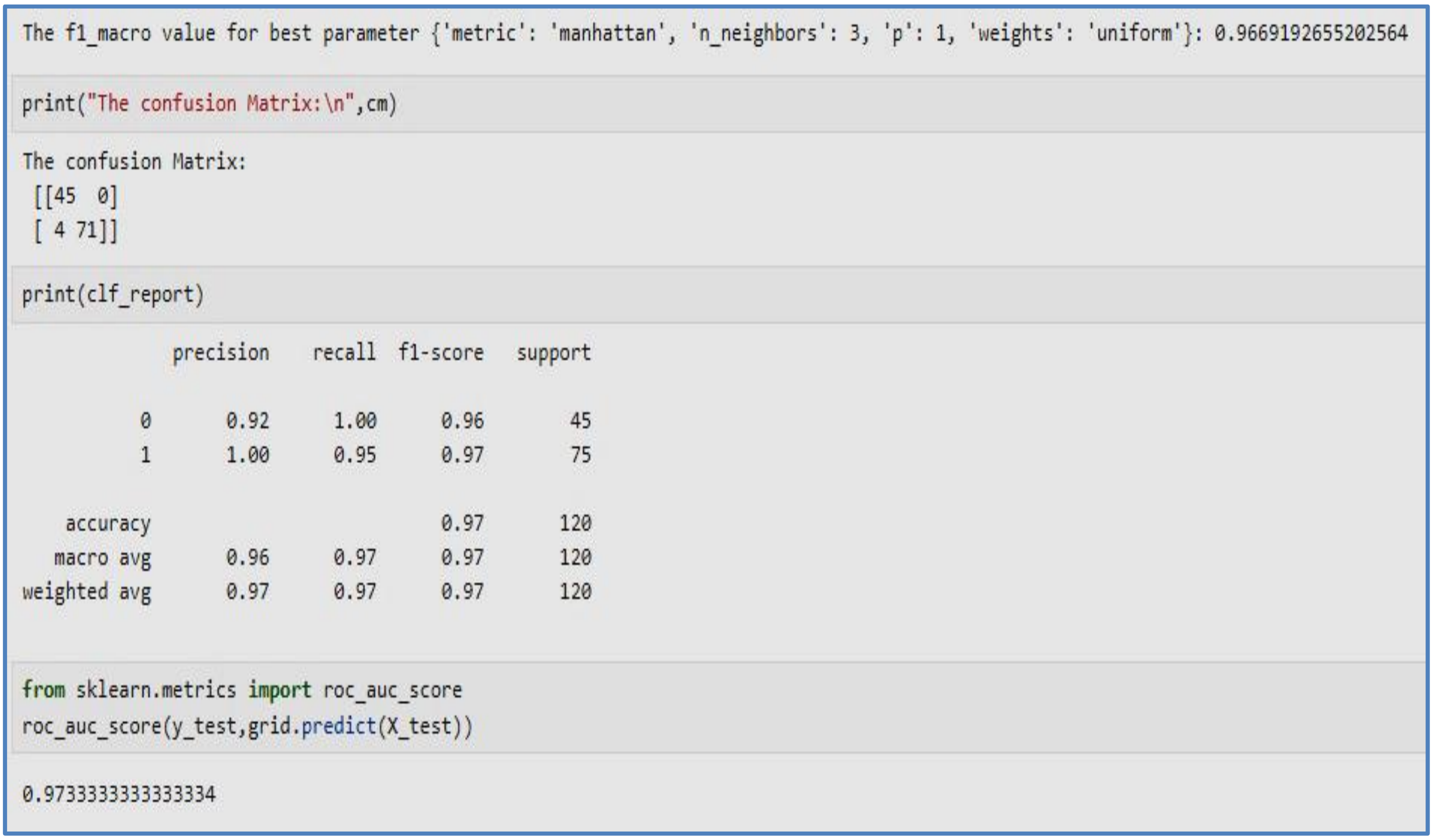
```
print(clf_report)
```

```
              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
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict(X_test))
```

```
0.9933333333333334
```

2. K-NEAREST NEIGHBOR



```
The f1_macro value for best parameter {'metric': 'manhattan', 'n_neighbors': 3, 'p': 1, 'weights': 'uniform'}: 0.9669192655202564
```

```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 4 71]]
```

```
print(clf_report)
```

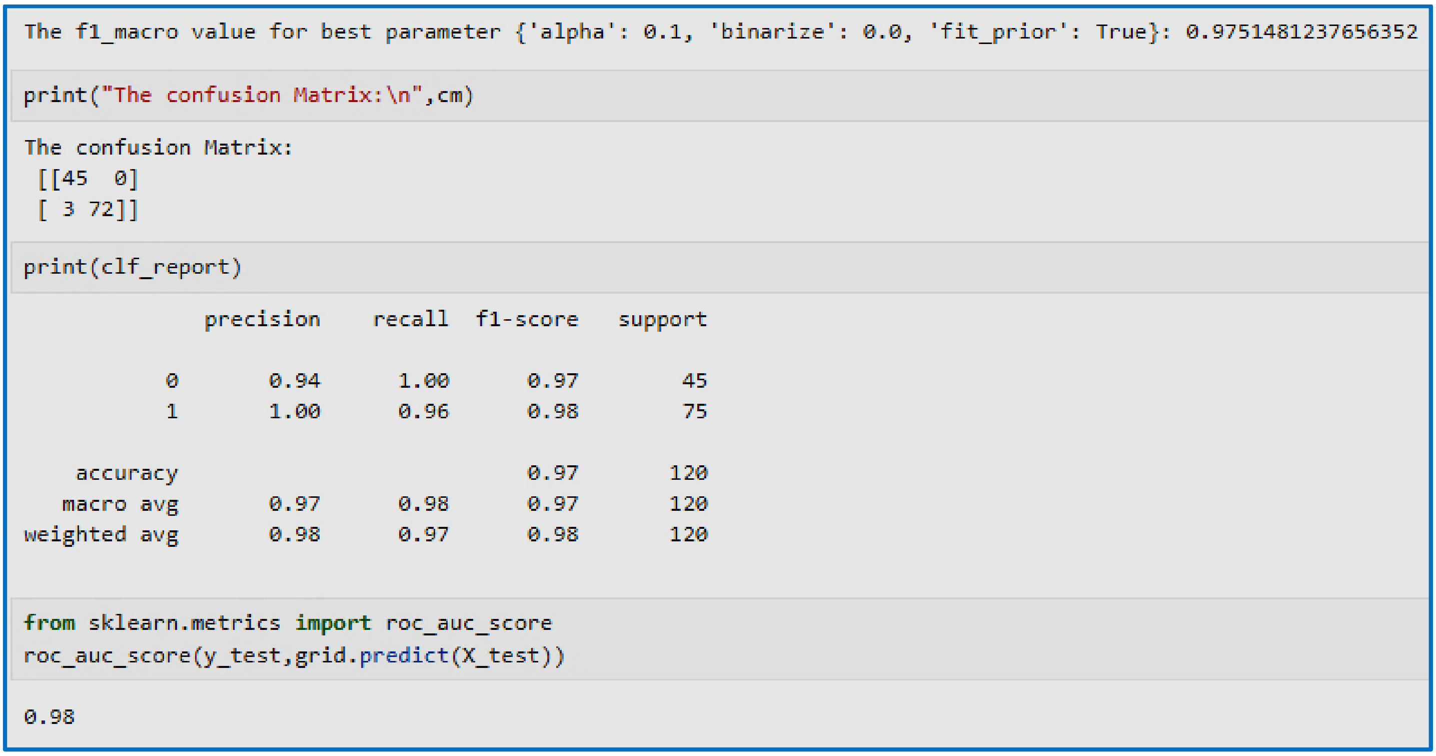
```
              precision    recall  f1-score   support

           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
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict(X_test))
```

```
0.9733333333333334
```

3. NAÏVE BAYES

➢ BERNOULLI NB



```
The f1_macro value for best parameter {'alpha': 0.1, 'binarize': 0.0, 'fit_prior': True}: 0.9751481237656352
```

```
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 3 72]]
```

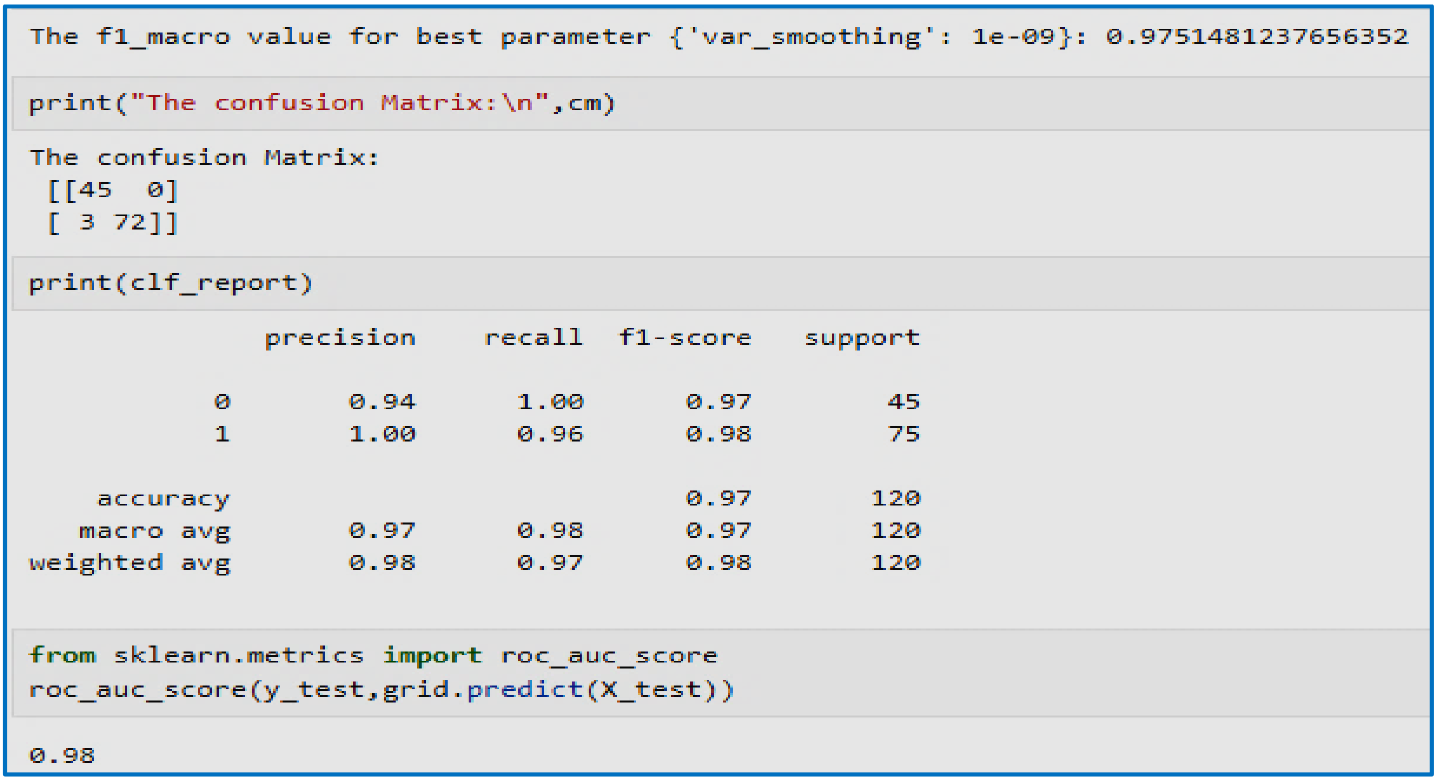
```
print(clf_report)
```

```
              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
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict(X_test))
```

```
0.98
```

➢ GAUSSIAN NB



```
The f1_macro value for best parameter {'var_smoothing': 1e-09}: 0.9751481237656352
```

```python
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 3 72]]
```

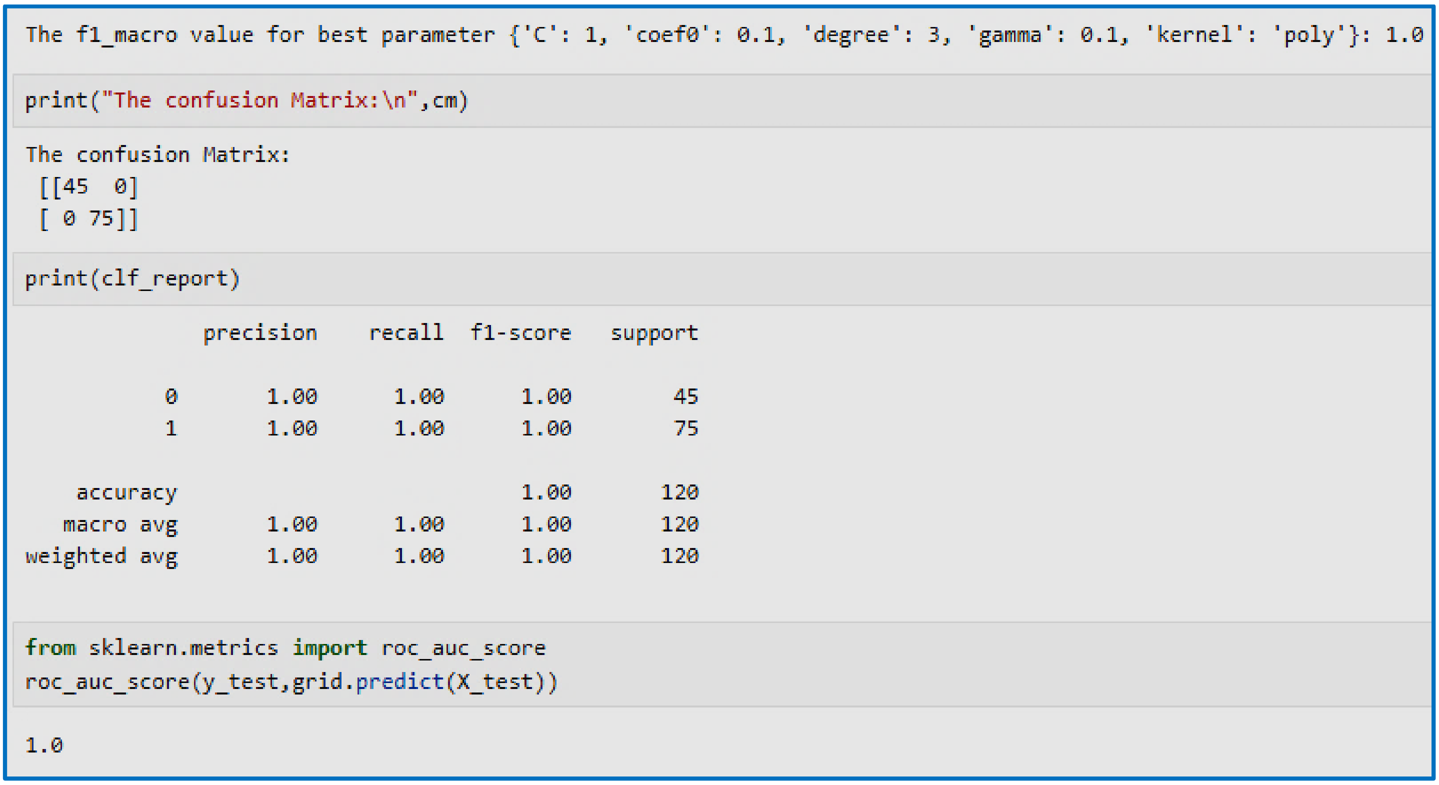
```python
print(clf_report)
```

```
              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
```

```python
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict(X_test))
```

```
0.98
```

4. SUPPORT VECTOR MACHINE



```
The f1_macro value for best parameter {'C': 1, 'coef0': 0.1, 'degree': 3, 'gamma': 0.1, 'kernel': 'poly'}: 1.0
```

```python
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 0 75]]
```

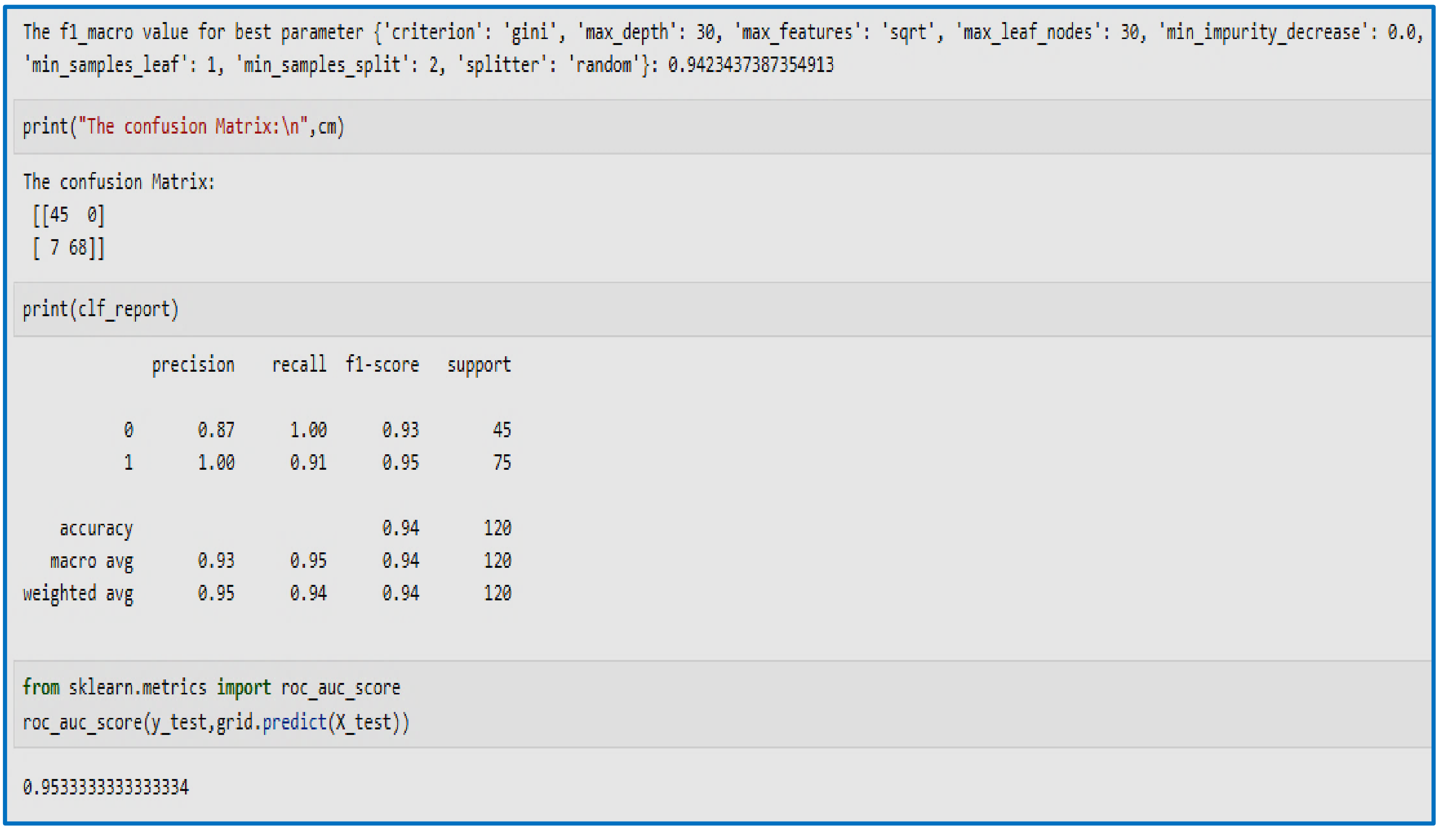
```python
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        45
           1       1.00      1.00      1.00        75

    accuracy                           1.00       120
   macro avg       1.00      1.00      1.00       120
weighted avg       1.00      1.00      1.00       120
```

```python
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict(X_test))
```

```
1.0
```

5. DECISION TREE



```
The f1_macro value for best parameter {'criterion': 'gini', 'max_depth': 30, 'max_features': 'sqrt', 'max_leaf_nodes': 30, 'min_impurity_decrease': 0.0,
'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'random'}: 0.9423437387354913
```

```python
print("The confusion Matrix:\n",cm)
```

```
The confusion Matrix:
 [[45  0]
 [ 7 68]]
```

```python
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.87      1.00      0.93        45
           1       1.00      0.91      0.95        75

    accuracy                           0.94       120
   macro avg       0.93      0.95      0.94       120
weighted avg       0.95      0.94      0.94       120
```

```python
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test,grid.predict(X_test))
```

```
0.9533333333333334
```

RESULT :

SVM is the best method for the given model.

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

- ***Why I have take SVM means it Accuracy is 1.00 Then confusion_matrix:***

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

it is a good model so I taked SVM .

- ***Logistics Regression, k-nearest neighbor, Naïve bayes, Decision Tree, all Accuracy is Lesser then SVM so I don't take this.***