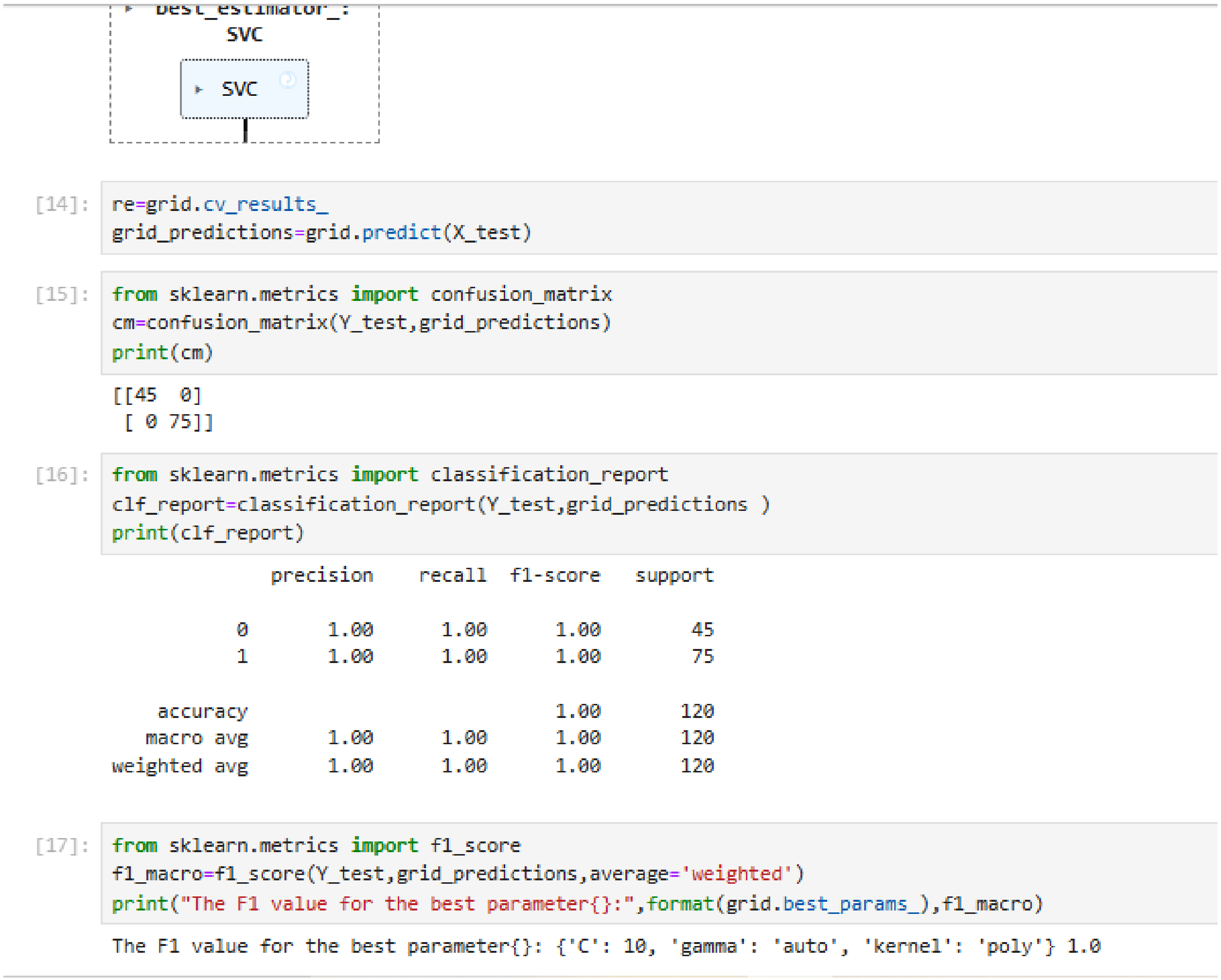
Classification Assignment

1. The given ask to find a model to predict, **Chronic Kidney Disease (CKD)** based on he given dataset
2. The dataset consist of **399 rows × 1 columns**
3. From the given dataset, the inputs like **rbc, pc, pcc, ba, htn, dm, cad, appet, pe, ane, classification** are in **string form.** So nominal data processing is done for converting the string information’s
4. Different Machine Learning algorithms are used to create the models. From that, best model is choosen

5. Classification-Assignment with Grid search for multiple algorithms

1. SVM Classifier:



[14]:
```
re=grid.cv_results_
grid_predictions=grid.predict(X_test)
```

[15]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print(cm)
```

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

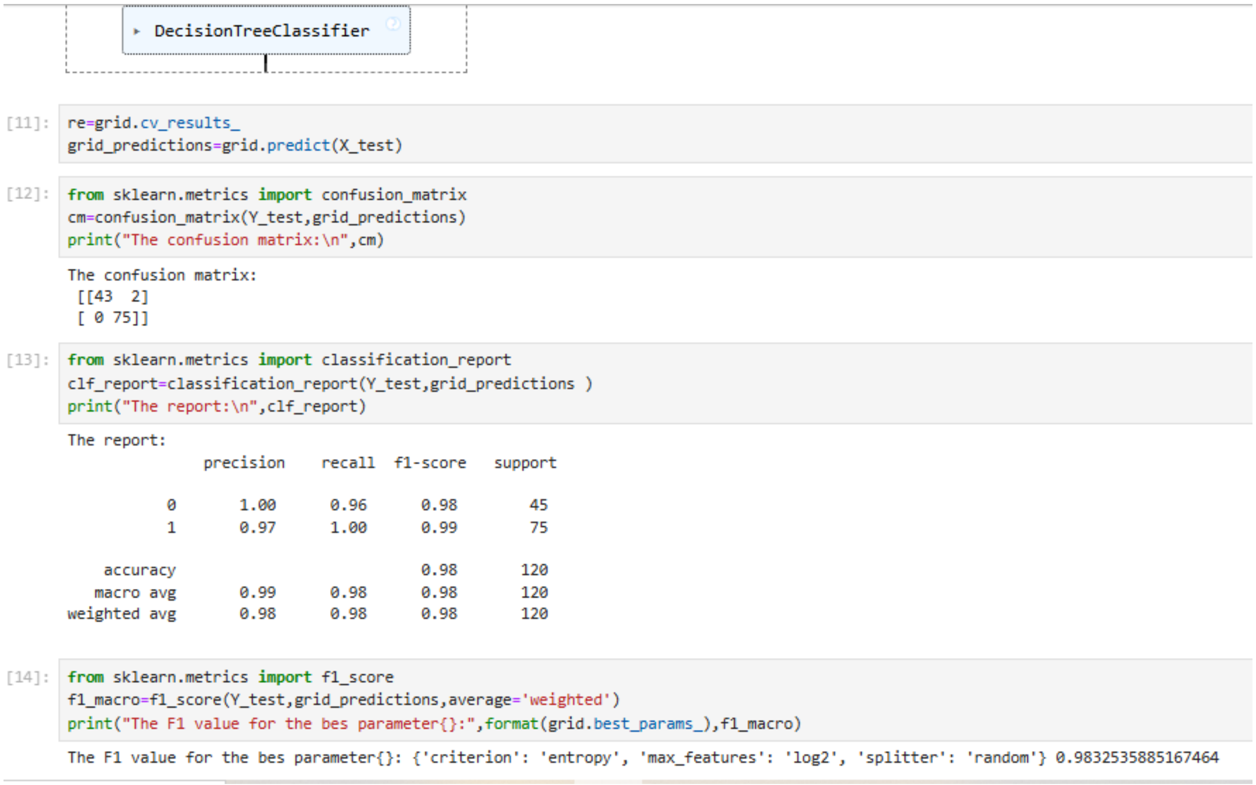
[16]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions )
print(clf_report)
```

```
              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
```

[17]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,grid_predictions,average='weighted')
print("The F1 value for the best parameter{}:",format(grid.best_params_),f1_macro)
```

```
The F1 value for the best parameter{}: {'C': 10, 'gamma': 'auto', 'kernel': 'poly'} 1.0
```

2. Decision Tree Classifier:



[11]:
```
re=grid.cv_results_
grid_predictions=grid.predict(X_test)
```

[12]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print("The confusion matrix:\n",cm)
```

```
The confusion matrix:
 [[43  2]
 [ 0 75]]
```

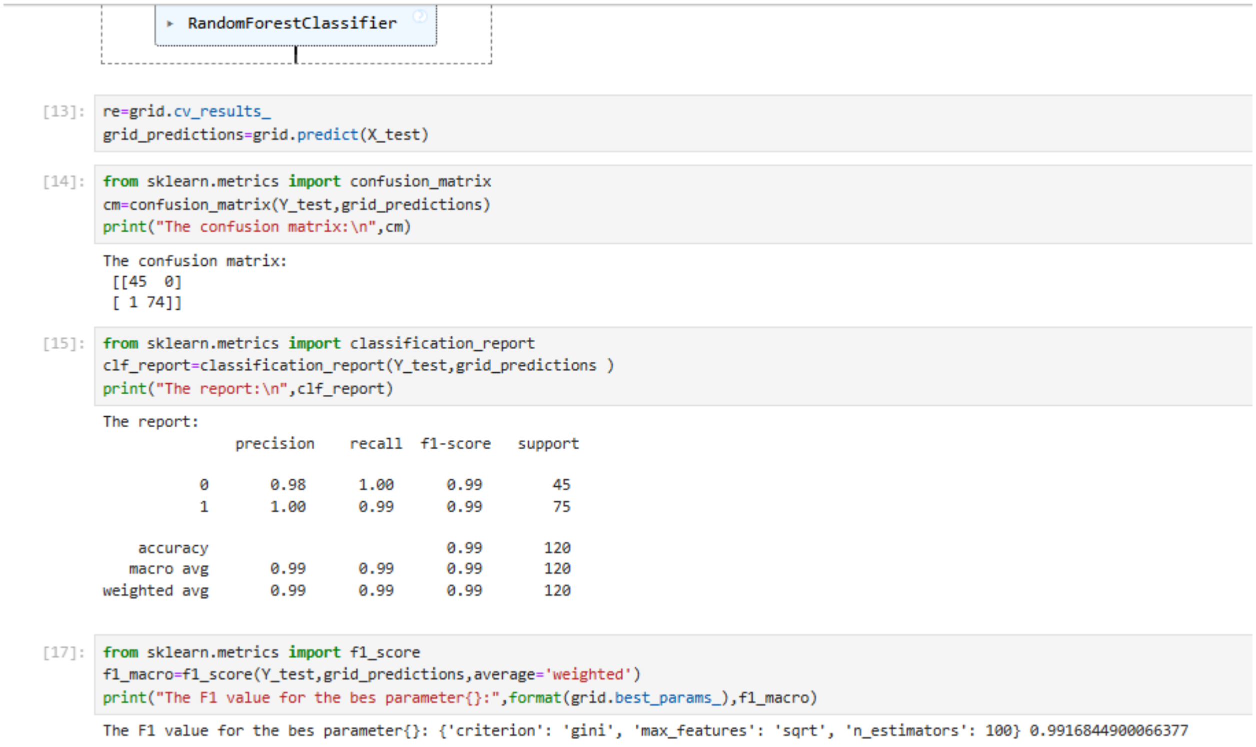
[13]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions )
print("The report:\n",clf_report)
```

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

           0       1.00      0.96      0.98        45
           1       0.97      1.00      0.99        75

    accuracy                           0.98       120
   macro avg       0.99      0.98      0.98       120
weighted avg       0.98      0.98      0.98       120
```

[14]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,grid_predictions,average='weighted')
print("The F1 value for the bes parameter{}:",format(grid.best_params_),f1_macro)
```

```
The F1 value for the bes parameter{}: {'criterion': 'entropy', 'max_features': 'log2', 'splitter': 'random'} 0.9832535885167464
```

3. Random Forest Classifier:



[13]:
```
re=grid.cv_results_
grid_predictions=grid.predict(X_test)
```

[14]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(Y_test,grid_predictions)
print("The confusion matrix:\n",cm)
```

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

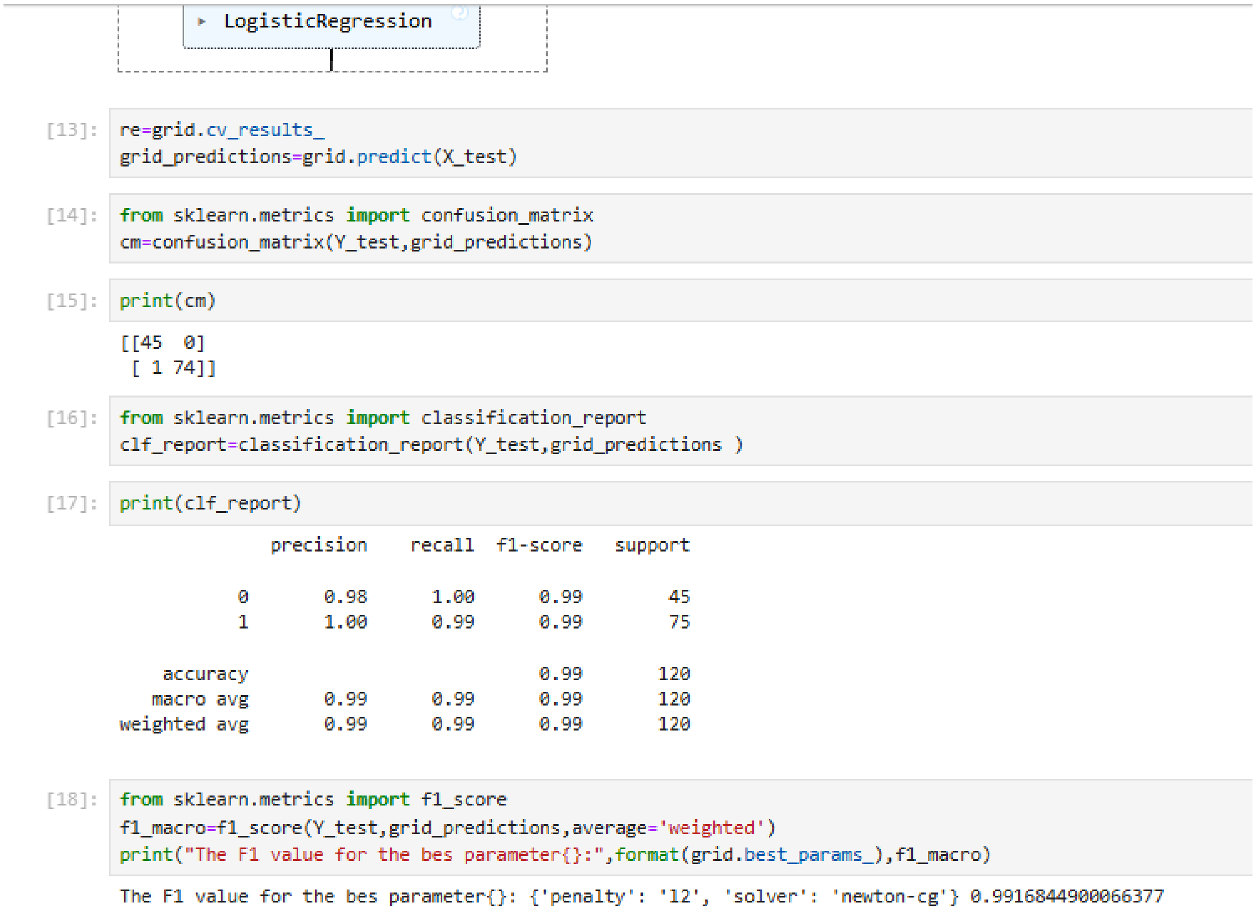
[15]:
```
from sklearn.metrics import classification_report
clf_report=classification_report(Y_test,grid_predictions )
print("The report:\n",clf_report)
```

```
The 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
```

[17]:
```
from sklearn.metrics import f1_score
f1_macro=f1_score(Y_test,grid_predictions,average='weighted')
print("The F1 value for the bes parameter{}:",format(grid.best_params_),f1_macro)
```

```
The F1 value for the bes parameter{}: {'criterion': 'gini', 'max_features': 'sqrt', 'n_estimators': 100} 0.9916844900066377
```

4. Logistic Regressor Classifier:



```
▸ LogisticRegression
```

```
[13]: re=grid.cv_results_
      grid_predictions=grid.predict(X_test)
```

```
[14]: from sklearn.metrics import confusion_matrix
      cm=confusion_matrix(Y_test,grid_predictions)
```

```
[15]: print(cm)
```

```
[[45  0]
 [ 1 74]]
```

```
[16]: from sklearn.metrics import classification_report
      clf_report=classification_report(Y_test,grid_predictions )
```

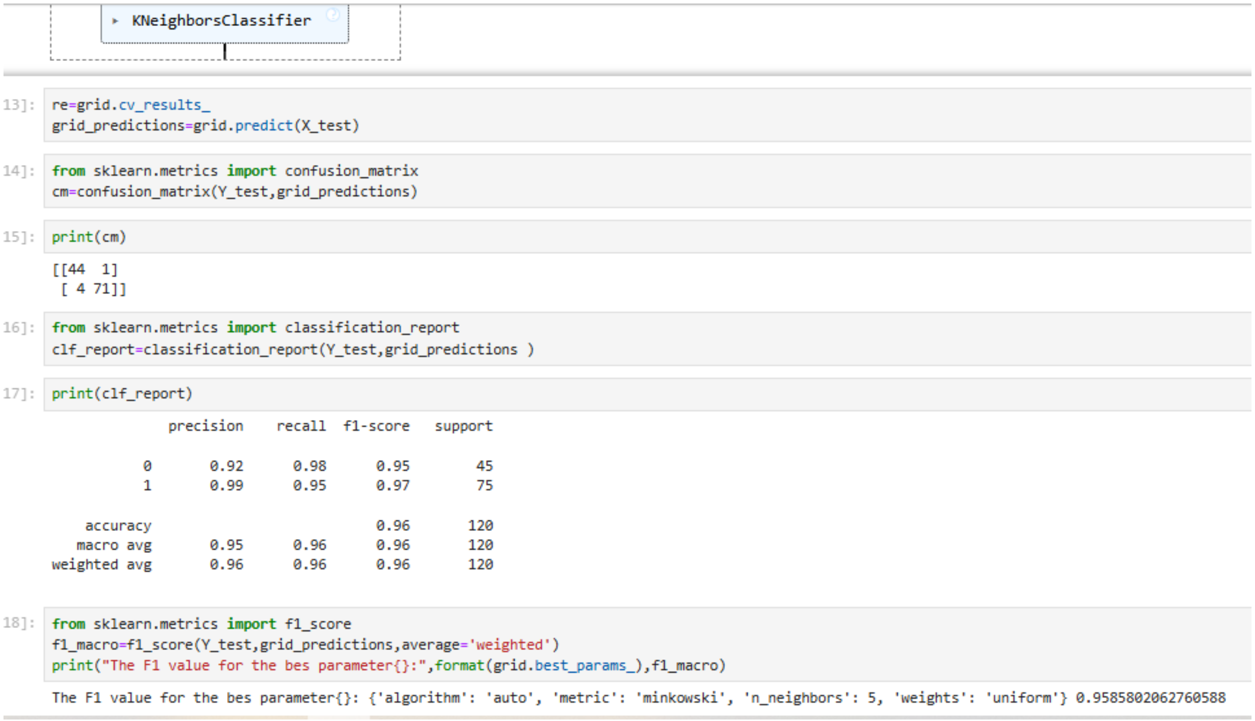
```
[17]: print(clf_report)
```

```
              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
```

```
[18]: from sklearn.metrics import f1_score
      f1_macro=f1_score(Y_test,grid_predictions,average='weighted')
      print("The F1 value for the bes parameter{}:",format(grid.best_params_),f1_macro)
```

```
The F1 value for the bes parameter{}: {'penalty': 'l2', 'solver': 'newton-cg'} 0.9916844900066377
```

5. KNN Classifier:



```
▸ KNeighborsClassifier
```

```
13]: re=grid.cv_results_
     grid_predictions=grid.predict(X_test)
```

```
14]: from sklearn.metrics import confusion_matrix
     cm=confusion_matrix(Y_test,grid_predictions)
```

```
15]: print(cm)
```

```
[[44  1]
 [ 4 71]]
```

```
16]: from sklearn.metrics import classification_report
     clf_report=classification_report(Y_test,grid_predictions )
```

```
17]: print(clf_report)
```

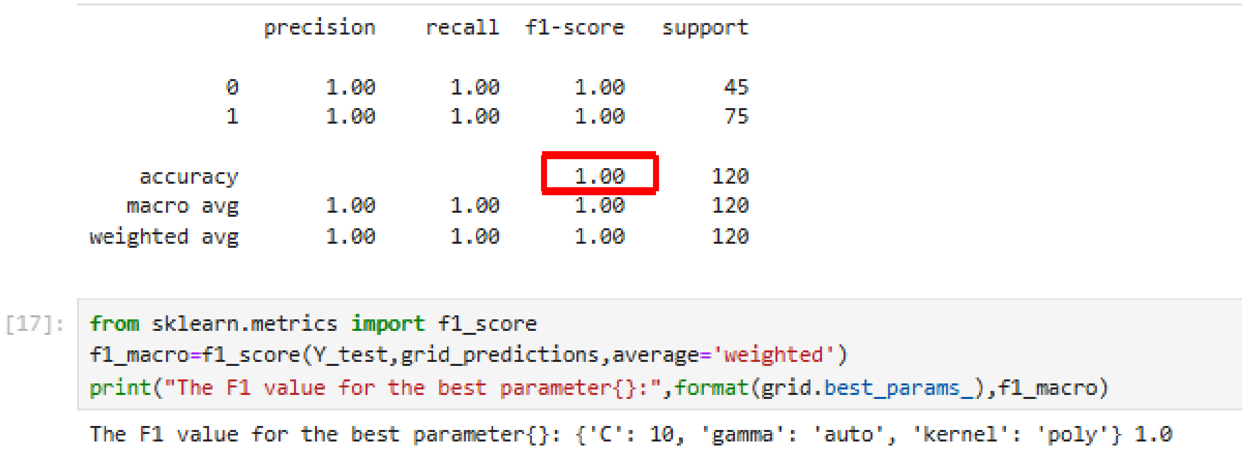
```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95        45
           1       0.99      0.95      0.97        75

    accuracy                           0.96       120
   macro avg       0.95      0.96      0.96       120
weighted avg       0.96      0.96      0.96       120
```

```
18]: from sklearn.metrics import f1_score
     f1_macro=f1_score(Y_test,grid_predictions,average='weighted')
     print("The F1 value for the bes parameter{}:",format(grid.best_params_),f1_macro)
```

```
The F1 value for the bes parameter{}: {'algorithm': 'auto', 'metric': 'minkowski', 'n_neighbors': 5, 'weights': 'uniform'} 0.9585802062760588
```

6. From the analysis, Support Vector Machine Classifier is chosen as a best model. Because, it produces **zero errors** hence the **accuracy is 100%**

Result of SVM Classifier:



```
              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
```

```
[17]: from sklearn.metrics import f1_score
      f1_macro=f1_score(Y_test,grid_predictions,average='weighted')
      print("The F1 value for the best parameter{}:",format(grid.best_params_),f1_macro)
```

```
The F1 value for the best parameter{}: {'C': 10, 'gamma': 'auto', 'kernel': 'poly'} 1.0
```