

## 01.LG Classification Final Module Assignment

01.LG\_Classification\_Final\_Module\_Assignment

```
[13]: result = grid.cv_results_
      y_pred = grid.predict(X_test)

      from sklearn.metrics import confusion_matrix
      confusion_matrix = confusion_matrix(y_test, y_pred)
      print(confusion_matrix)

      from sklearn.metrics import classification_report
      clf_report = classification_report(y_test, y_pred)
      print(clf_report)

[[51  0]
 [ 1 81]]

              precision    recall  f1-score   support

         0       0.98        1.00        0.99         51
         1       1.00        0.99        0.99         82

 accuracy          0.99
 macro avg          0.99
weighted avg          0.99
```

[14]: #clf\_report.\_\_add\_\_

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

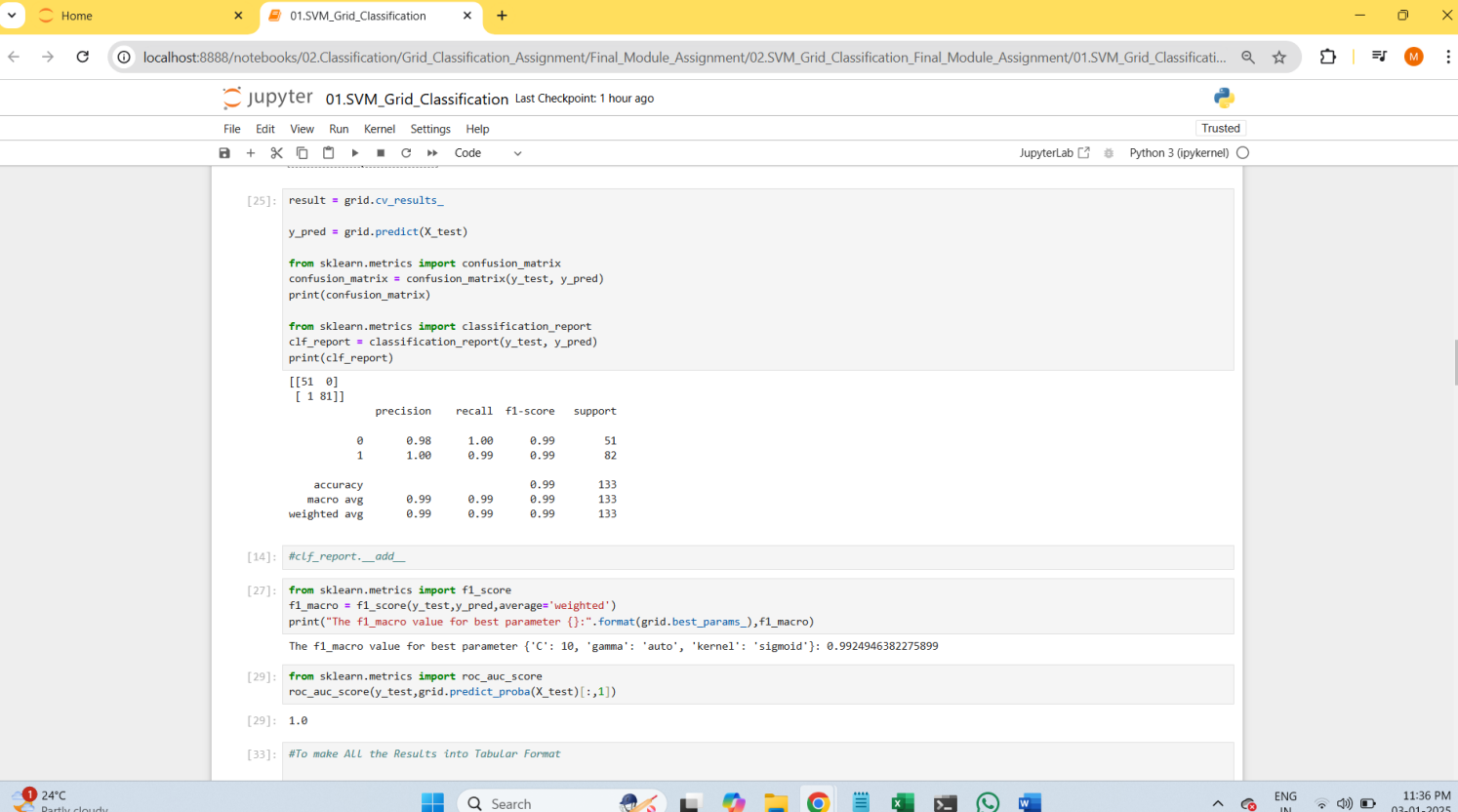
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'newton-cg'}: 0.9924946382275899
```

[16]: from sklearn.metrics import roc\_auc\_score
 roc\_auc\_score(y\_test, grid.predict\_proba(X\_test)[:,1])

[16]: 1.0

[33]: #To make ALL the Results into Tabular Format

## 02.SVM Grid Classification



The screenshot shows a Jupyter Notebook interface with the following content:

```
[25]: result = grid.cv_results_

y_pred = grid.predict(X_test)

from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)

from sklearn.metrics import classification_report
clf_report = classification_report(y_test, y_pred)
print(clf_report)

[[51  0]
 [ 1 81]]

              precision    recall  f1-score   support

      0       0.98        1.00        0.99         51
      1       1.00        0.99        0.99         82

 accuracy          0.99
 macro avg          0.99
 weighted avg       0.99
```

```
[14]: #clf_report.__add__

[27]: from sklearn.metrics import f1_score
f1_macro = f1_score(y_test, y_pred, 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': 'sigmoid'}: 0.9924946382275899

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

[29]: 1.0

[33]: #To make ALL the Results into Tabular Format
```

The bottom of the image shows a Windows taskbar with the date 03-01-2025 and time 11:36 PM.

## 03. Decision Tree Grid Classification Final Module Assignment

The screenshot shows a JupyterLab notebook titled "01.DT\_Grid\_Classification" with the following code and output:

```
[25]: result = grid.cv_results_  
y_pred = grid.predict(X_test)  
  
from sklearn.metrics import confusion_matrix  
confusion_matrix = confusion_matrix(y_test, y_pred)  
print(confusion_matrix)  
  
from sklearn.metrics import classification_report  
clf_report = classification_report(y_test, y_pred)  
print(clf_report)
```

Output:

```
[[48  3]  
 [ 5 77]]
```

	precision	recall	f1-score	support
0	0.91	0.94	0.92	51
1	0.96	0.94	0.95	82
accuracy			0.94	133
macro avg	0.93	0.94	0.94	133
weighted avg	0.94	0.94	0.94	133

```
[14]: #clf_report.__add__  
  
[27]: from sklearn.metrics import f1_score  
f1_macro = f1_score(y_test, y_pred, averages='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': 'log2', 'splitter': 'random'}: 0.9400566944426594  
  
[29]: from sklearn.metrics import roc_auc_score  
roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])  
  
[29]: 0.9401004304160688  
  
[31]: #To make ALL the Results into Tabular Format
```

## 04. Random Forest Grid Classification Final Module Assignment

The screenshot shows a JupyterLab notebook titled "01.RF\_Grid\_Classification" with the following code and output:

```
[13]: result = grid.cv_results_  
y_pred = grid.predict(X_test)  
  
from sklearn.metrics import confusion_matrix  
confusion_matrix = confusion_matrix(y_test, y_pred)  
print(confusion_matrix)  
  
from sklearn.metrics import classification_report  
clf_report = classification_report(y_test, y_pred)  
print(clf_report)
```

Output:

```
[[49  2]  
 [ 1 81]]
```

	precision	recall	f1-score	support
0	0.98	0.96	0.97	51
1	0.98	0.99	0.98	82
accuracy			0.98	133
macro avg	0.98	0.97	0.98	133
weighted avg	0.98	0.98	0.98	133

```
[14]: #clf_report.__add__  
  
[15]: from sklearn.metrics import f1_score  
f1_macro = f1_score(y_test, y_pred, averages='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', 'n_estimators': 10}: 0.9774002964206194  
  
[16]: from sklearn.metrics import roc_auc_score  
roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])  
  
[16]: 0.9990435198469632  
  
[33]: #To make ALL the Results into Tabular Format
```

## Overall Result

- The Best Model would be saved as Deployment Phase for the One which has Both the **Highest Accuracy Value and roc\_auc\_score** as Highlighted Below :

S.No	Type of Classification	Hyper Tuned Parameters	Accuracy	roc_auc_score
1	LG	param_grid = {'solver':['newton-cg','lbfgs','liblinear','saga'], 'penalty':['l2']}	0.99	NA
2	SVM	param_grid = {'kernel':['linear','rbf','poly','sigmoid'], 'gamma':['auto','scale'], 'C':[10,100,1000,2000,3000]}	0.99	1.00
3	Decision Tree	param_grid = {'criterion':['gini','entropy'], 'max_features': ['auto','sqrt','log2'], 'splitter':['best','random']}	0.94	0.94
4	Random Forest	param_grid = {'criterion':['gini','entropy'], 'max_features': ['auto','sqrt','log2'], 'n_estimators':[10,100]}	0.98	0.99

.....