Classification Assignment Report

- **Identify your problem statement:**

  Client requirement is he want to predict the chronic kidney disease (CKD), Classification is the output, so have clear requirement it's belongs to supervised machine learning

  Three Stages

  • Machine Learning- numbers

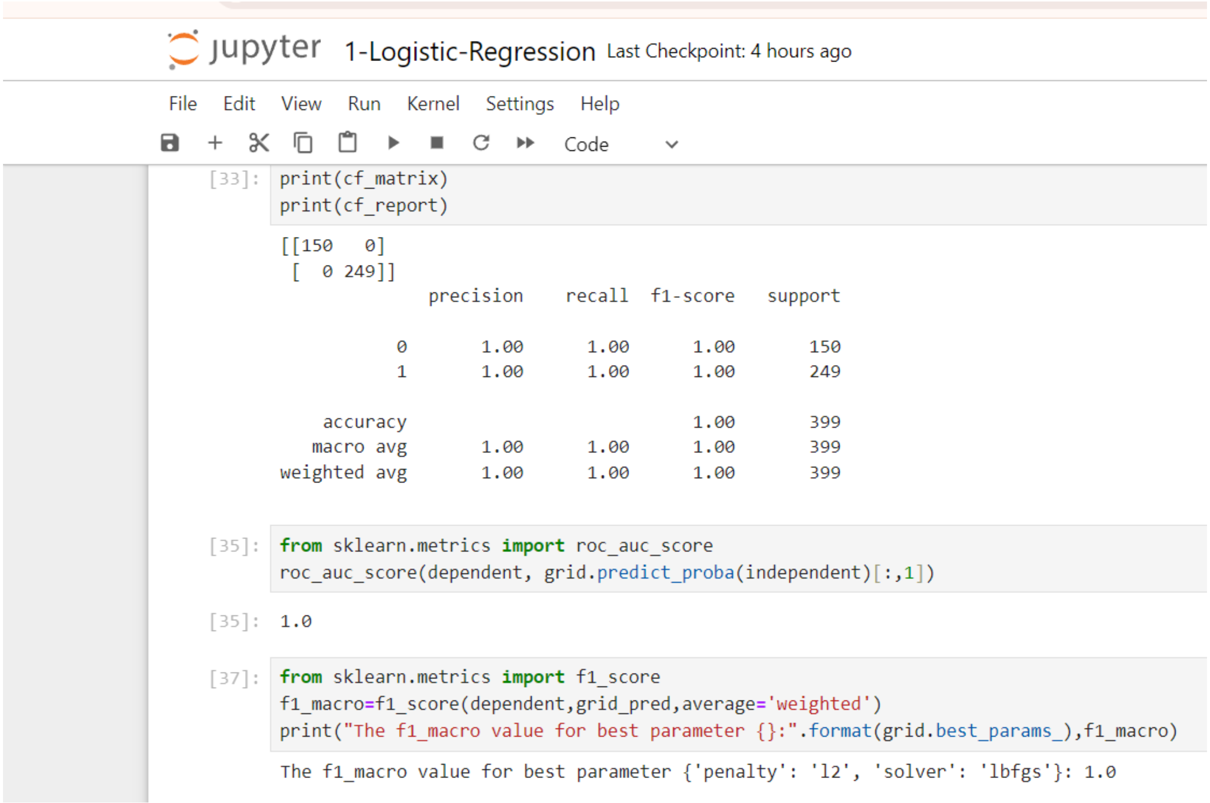
  • Supervised Machine Learning – requirement is very clear

  • Classification – output is continuous values

- **Tell basic info about the dataset**
  Rows: 399
  Columns:28
  Output variable: Classification

- **Mention the pre-processing method if you're doing any (like converting string to number – nominal data)**
  Pre-processing method is Nominal data (one hot encoding) because input contains text, we are using get_dummies method and the parameters we are passing dataset, dtype=int, drop_first=true

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

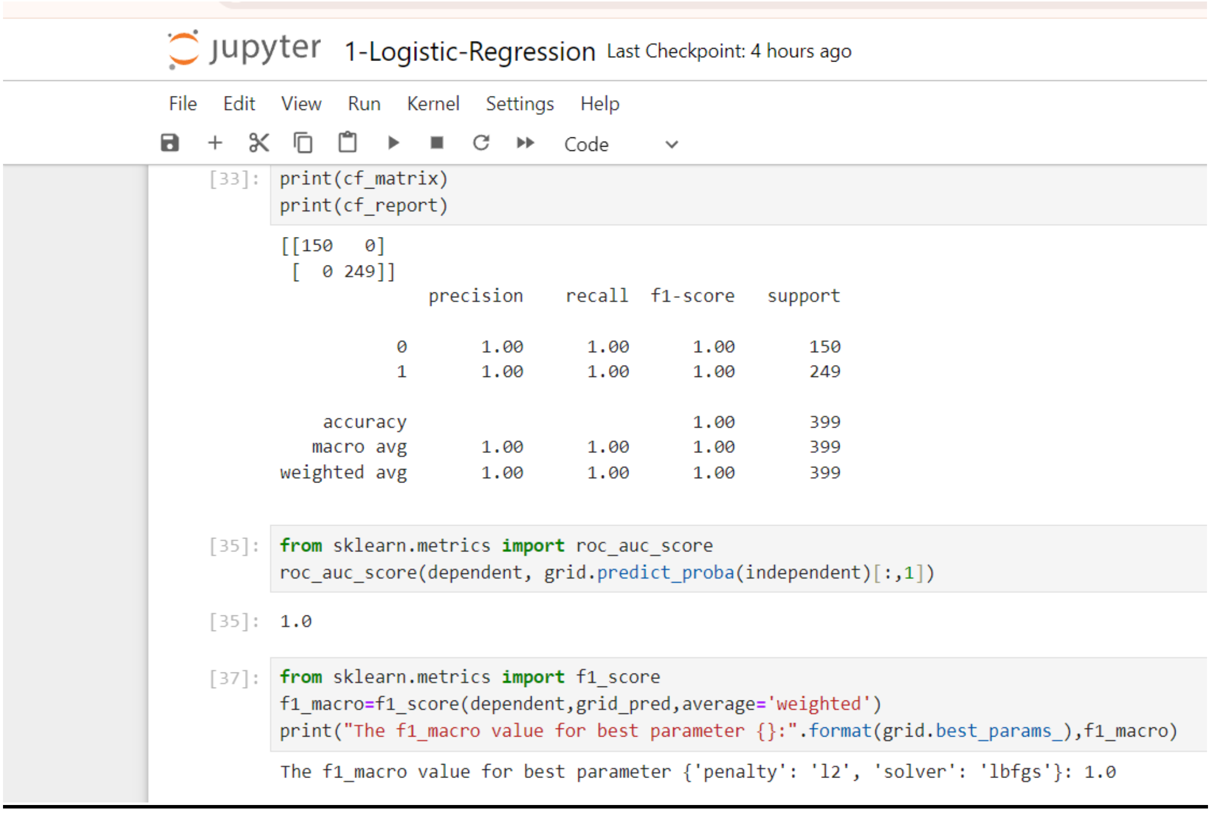


jupyter
1-Logistic-Regression Last Checkpoint: 4 hours ago
File Edit View Run Kernel Settings Help
Code

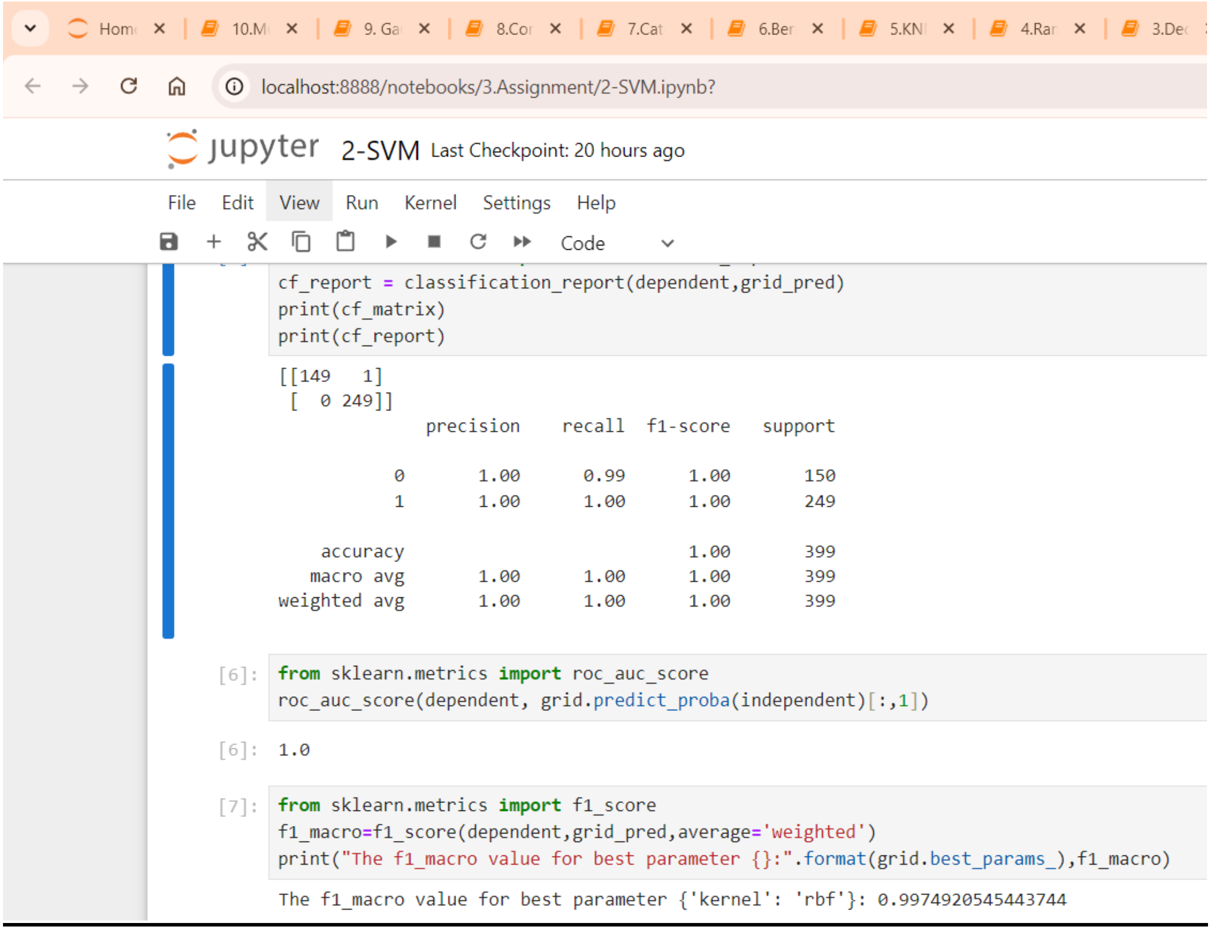
```
[33]: print(cf_matrix)
      print(cf_report)

      [[150   0]
       [  0 249]]
                    precision    recall  f1-score   support

                 0       1.00      1.00      1.00       150
                 1       1.00      1.00      1.00       249

          accuracy                           1.00       399
         macro avg       1.00      1.00      1.00       399
      weighted avg       1.00      1.00      1.00       399

[35]: from sklearn.metrics import roc_auc_score
      roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[35]: 1.0

[37]: from sklearn.metrics import f1_score
      f1_macro=f1_score(dependent,grid_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': 'lbfgs'}: 1.0
```

- **Mention your final model, justify why u have chosen the same**
  Final model is logistic-regression because the “roc_auc_score” 1.0, and accuracy is 100%

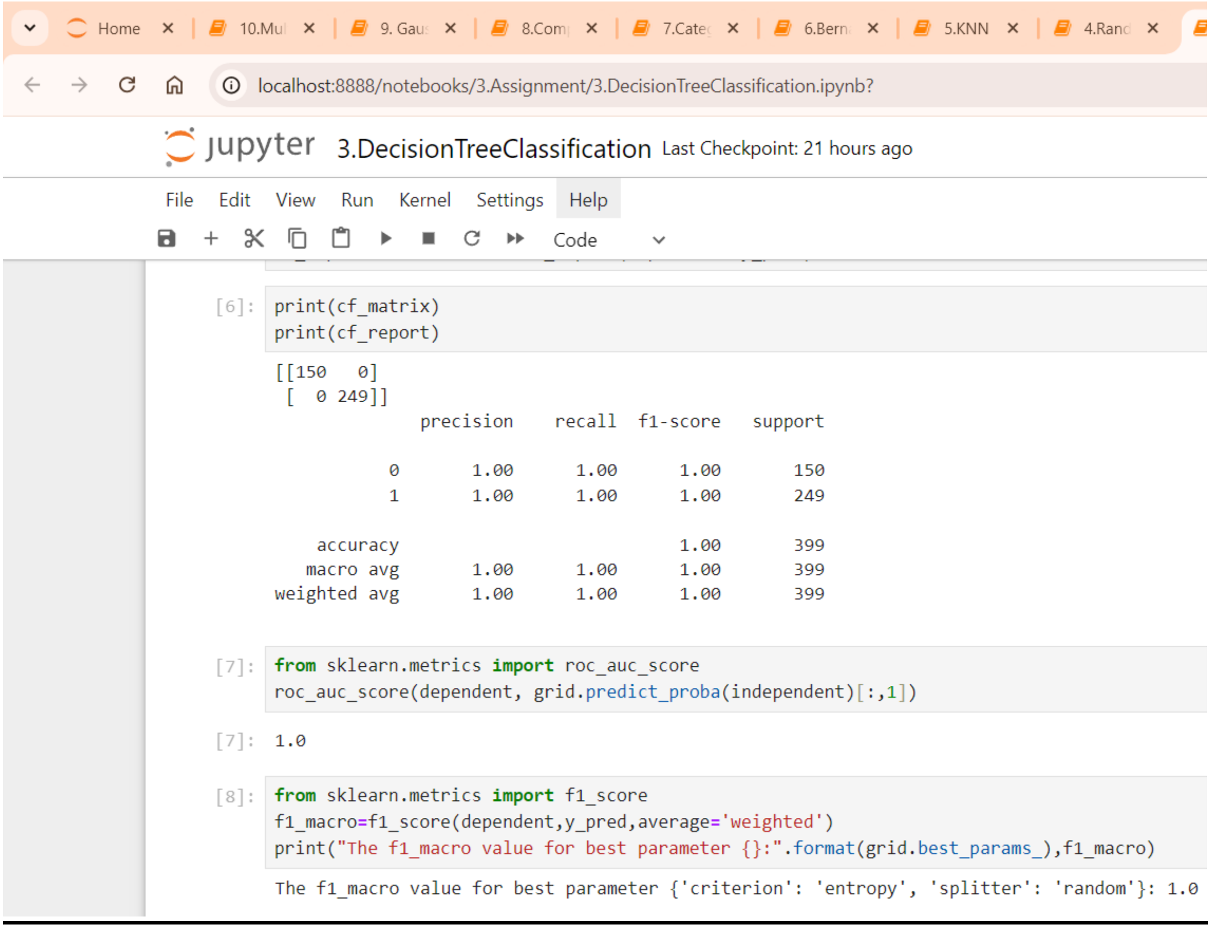
Complete report for all algorithms:



```
[33]: print(cf_matrix)
      print(cf_report)

      [[150   0]
       [  0 249]]
                    precision    recall  f1-score   support

                 0       1.00      1.00      1.00       150
                 1       1.00      1.00      1.00       249

          accuracy                           1.00       399
         macro avg       1.00      1.00      1.00       399
      weighted avg       1.00      1.00      1.00       399

[35]: from sklearn.metrics import roc_auc_score
      roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[35]: 1.0

[37]: from sklearn.metrics import f1_score
      f1_macro=f1_score(dependent,grid_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': 'lbfgs'}: 1.0
```


Hom
10.M
9. Ga
8.Co
7.Cat
6.Ber
5.KN
4.Rar
3.De
localhost:8888/notebooks/3.Assignment/2-SVM.ipynb?
jupyter 2-SVM Last Checkpoint: 20 hours ago
File
Edit
View
Run
Kernel
Settings
Help
Code

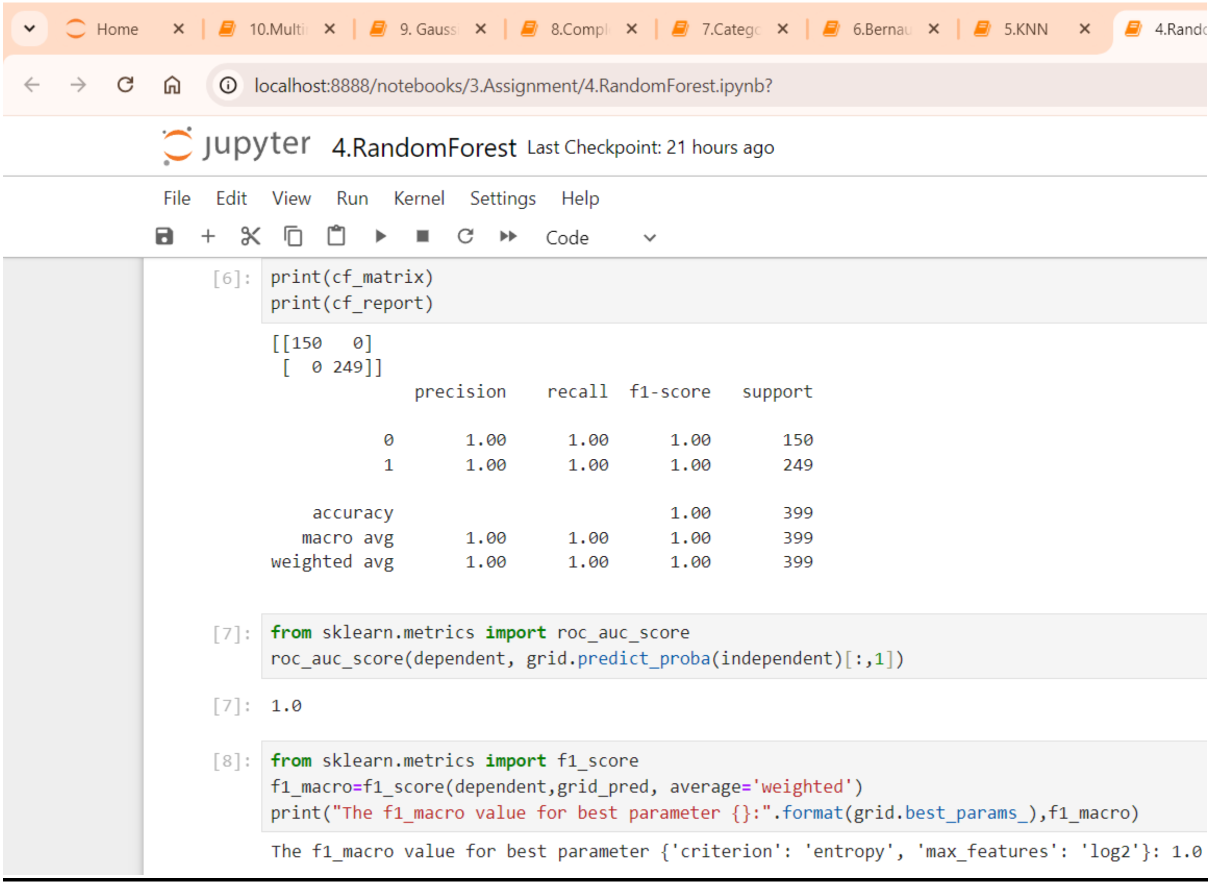
```
      cf_report = classification_report(dependent,grid_pred)
      print(cf_matrix)
      print(cf_report)

      [[149   1]
       [  0 249]]
                    precision    recall  f1-score   support

                 0       1.00      0.99      1.00       150
                 1       1.00      1.00      1.00       249

          accuracy                           1.00       399
         macro avg       1.00      1.00      1.00       399
      weighted avg       1.00      1.00      1.00       399

[6]:  from sklearn.metrics import roc_auc_score
      roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[6]:  1.0

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

      The f1_macro value for best parameter {'kernel': 'rbf'}: 0.9974920545443744
```


Home
10.Mu
9. Gau
8.Com
7.Cate
6.Bern
5.KNN
4.Rand
localhost:8888/notebooks/3.Assignment/3.DecisionTreeClassification.ipynb?
jupyter
3.DecisionTreeClassification
Last Checkpoint: 21 hours ago
File
Edit
View
Run
Kernel
Settings
Help
Code

```
[6]: print(cf_matrix)
     print(cf_report)

[[150   0]
 [  0 249]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
[7]: from sklearn.metrics import roc_auc_score
     roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[7]: 1.0
```

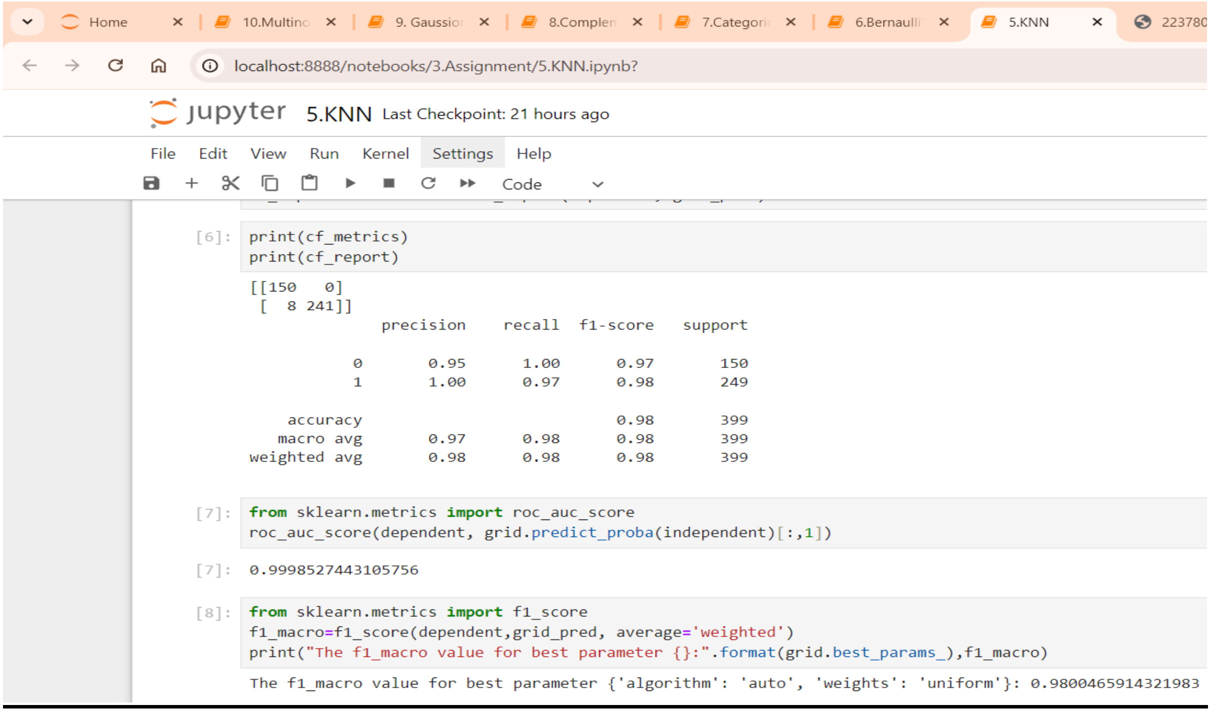
```
[8]: from sklearn.metrics import f1_score
     f1_macro=f1_score(dependent,y_pred,average='weighted')
     print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

The f1_macro value for best parameter {'criterion': 'entropy', 'splitter': 'random'}: 1.0
```



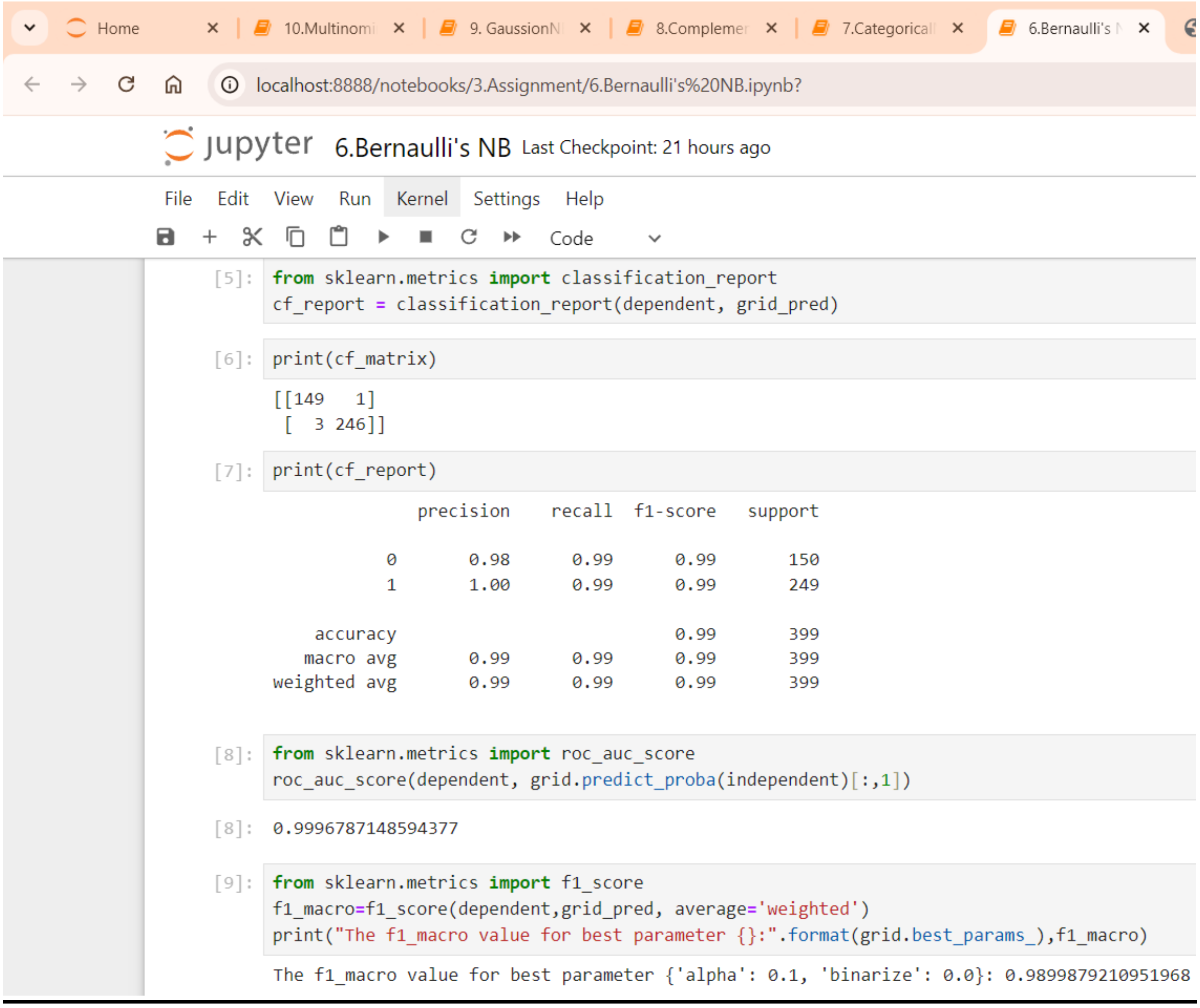
```
[6]: print(cf_matrix)
     print(cf_report)

[[150   0]
 [  0 249]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       150
           1       1.00      1.00      1.00       249

    accuracy                           1.00       399
   macro avg       1.00      1.00      1.00       399
weighted avg       1.00      1.00      1.00       399
```

```
[7]: from sklearn.metrics import roc_auc_score
     roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[7]: 1.0
```

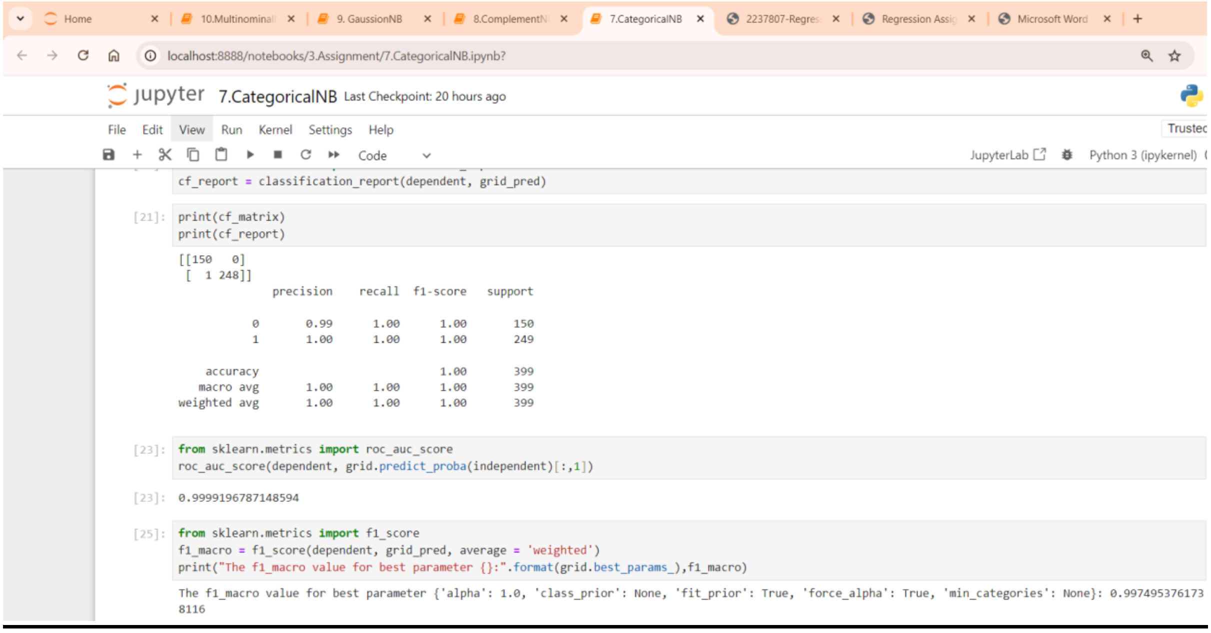
```
[8]: from sklearn.metrics import f1_score
     f1_macro=f1_score(dependent,grid_pred, average='weighted')
     print("The f1_macro value for best parameter {}:".format(grid.best_params_),f1_macro)

The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'log2'}: 1.0
```



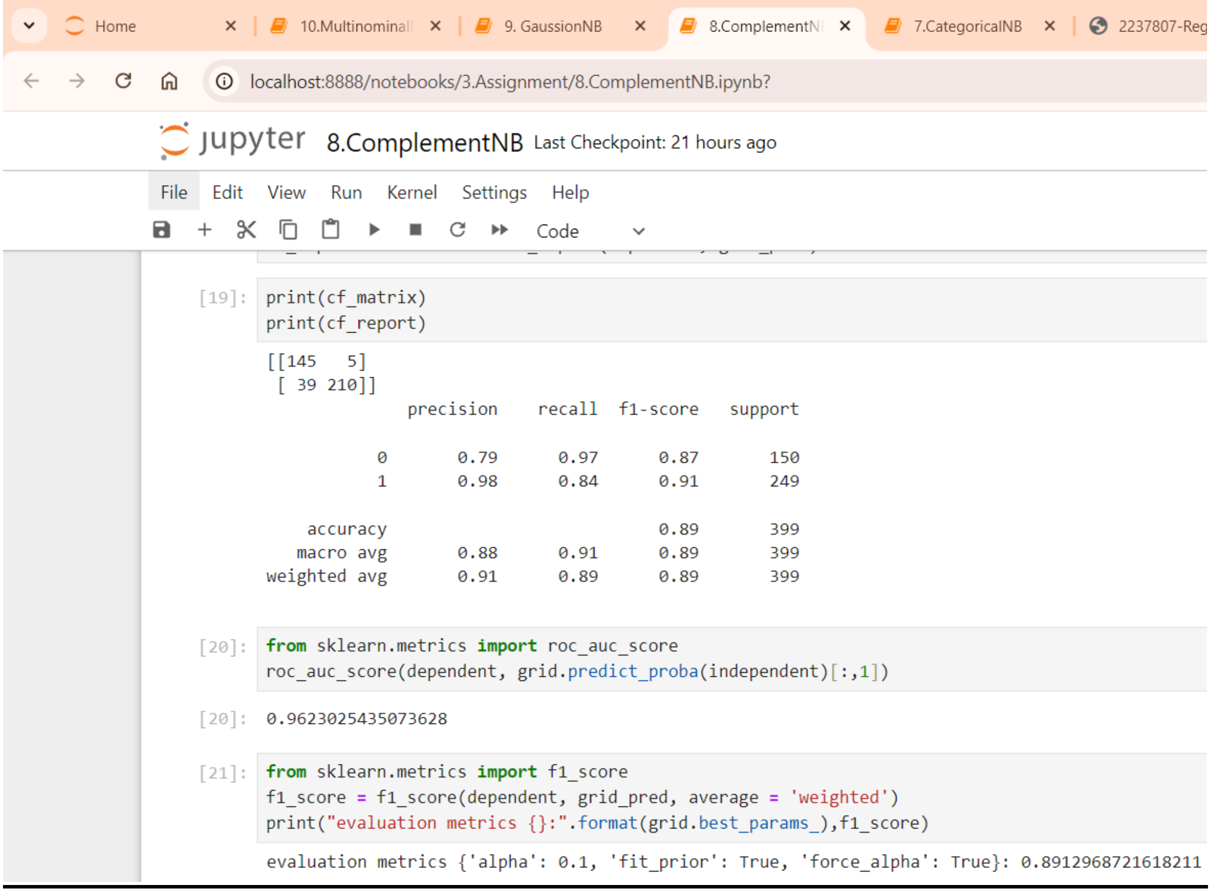
```
[6]: print(cf_metrics)
     print(cf_report)

     [[150   0]
      [  8 241]]
                   precision    recall  f1-score   support

                0       0.95      1.00      0.97       150
                1       1.00      0.97      0.98       249

         accuracy                           0.98       399
        macro avg       0.97      0.98      0.98       399
     weighted avg       0.98      0.98      0.98       399

[7]: from sklearn.metrics import roc_auc_score
     roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[7]: 0.9998527443105756

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

     The f1_macro value for best parameter {'algorithm': 'auto', 'weights': 'uniform'}: 0.9800465914321983
```



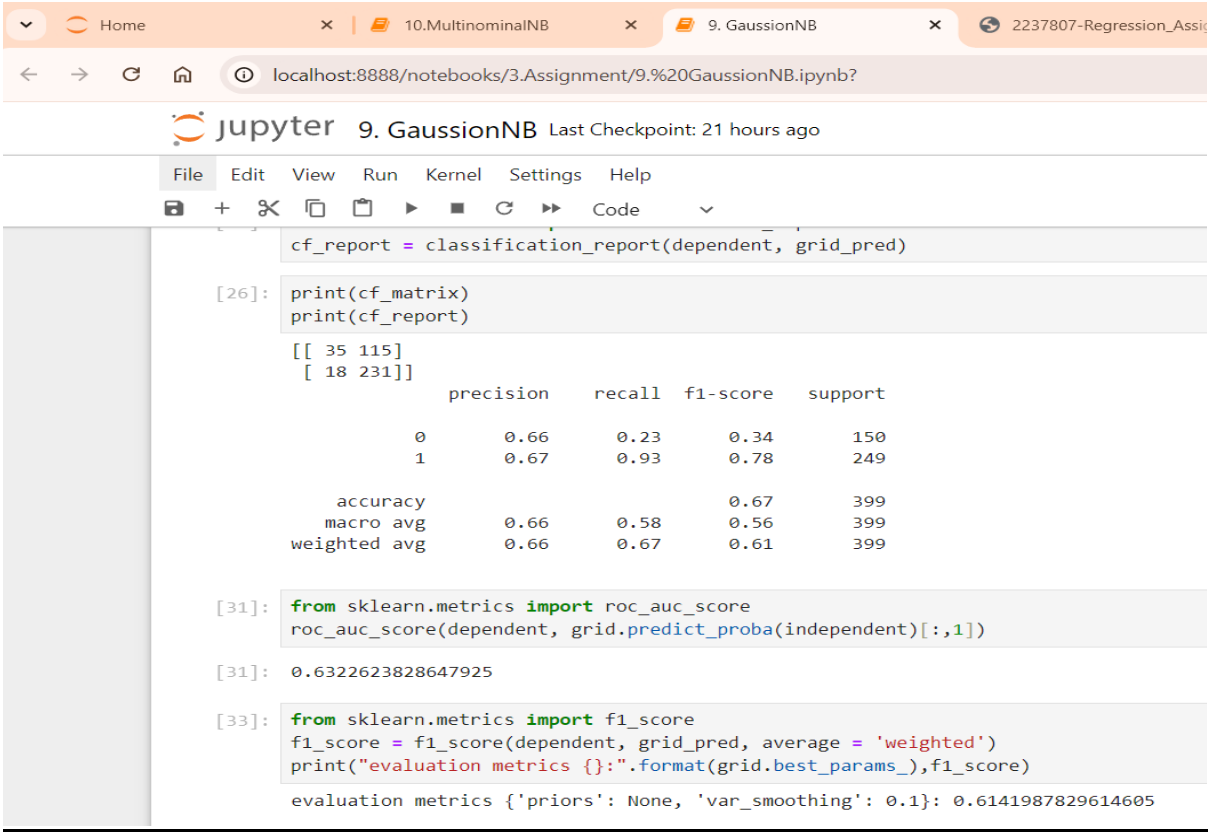
```
[5]: from sklearn.metrics import classification_report
     cf_report = classification_report(dependent, grid_pred)

[6]: print(cf_matrix)

     [[149   1]
      [  3 246]]

[7]: print(cf_report)

                   precision    recall  f1-score   support

                0       0.98      0.99      0.99       150
                1       1.00      0.99      0.99       249

         accuracy                           0.99       399
        macro avg       0.99      0.99      0.99       399
     weighted avg       0.99      0.99      0.99       399

[8]: from sklearn.metrics import roc_auc_score
     roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[8]: 0.9996787148594377

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

     The f1_macro value for best parameter {'alpha': 0.1, 'binarize': 0.0}: 0.9899879210951968
```



```
cf_report = classification_report(dependent, grid_pred)

[21]: print(cf_matrix)
      print(cf_report)

      [[150   0]
       [  1 248]]
                    precision    recall  f1-score   support

                 0       0.99      1.00      1.00       150
                 1       1.00      1.00      1.00       249

          accuracy                           1.00       399
         macro avg       1.00      1.00      1.00       399
      weighted avg       1.00      1.00      1.00       399

[23]: from sklearn.metrics import roc_auc_score
      roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[23]: 0.9999196787148594

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

      The f1_macro value for best parameter {'alpha': 1.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True, 'min_categories': None}: 0.997495376173
      8116
```



```
[19]: print(cf_matrix)
      print(cf_report)

      [[145   5]
       [ 39 210]]
                    precision    recall  f1-score   support

                 0       0.79      0.97      0.87       150
                 1       0.98      0.84      0.91       249

          accuracy                           0.89       399
         macro avg       0.88      0.91      0.89       399
      weighted avg       0.91      0.89      0.89       399

[20]: from sklearn.metrics import roc_auc_score
      roc_auc_score(dependent, grid.predict_proba(independent)[:,1])

[20]: 0.9623025435073628

[21]: from sklearn.metrics import f1_score
      f1_score = f1_score(dependent, grid_pred, average = 'weighted')
      print("evaluation metrics {}:".format(grid.best_params_),f1_score)

      evaluation metrics {'alpha': 0.1, 'fit_prior': True, 'force_alpha': True}: 0.8912968721618211
```



```
cf_report = classification_report(dependent, grid_pred)
```

```
[26]: print(cf_matrix)
      print(cf_report)
```

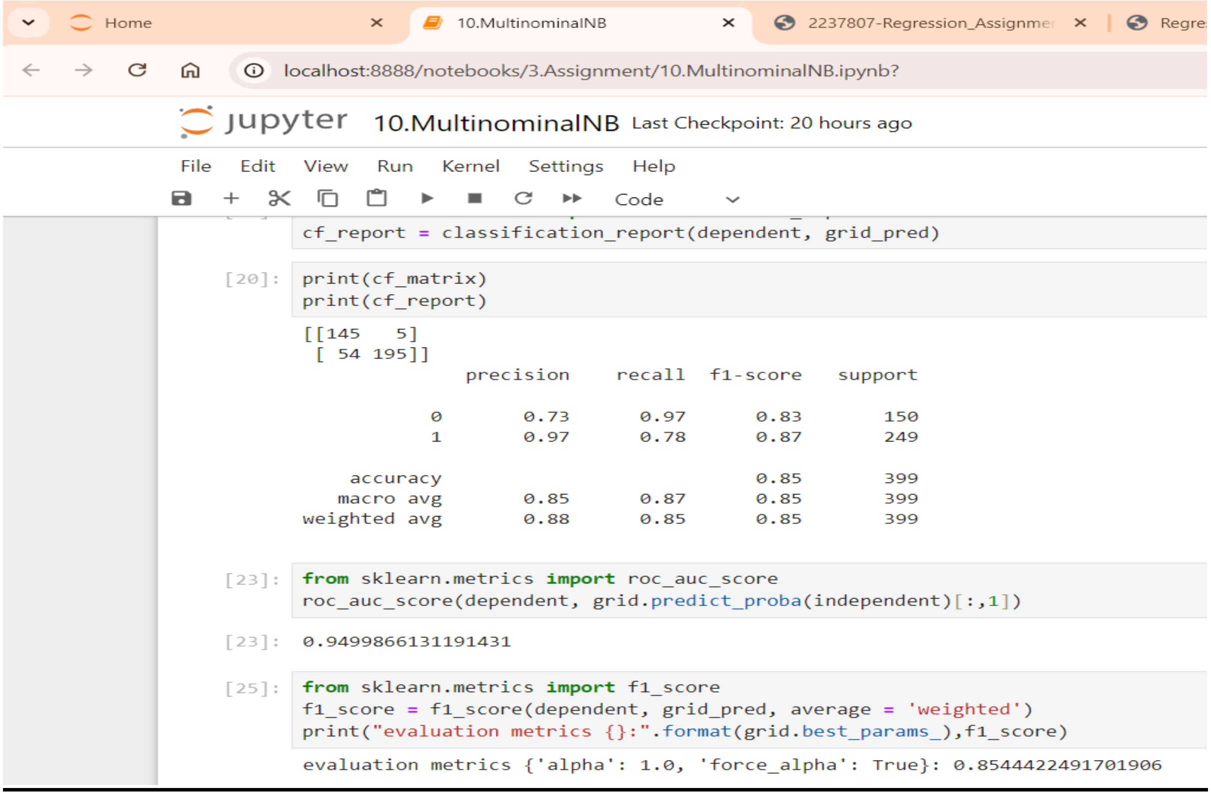
```
[[ 35 115]
 [ 18 231]]
              precision    recall  f1-score   support

           0       0.66      0.23      0.34       150
           1       0.67      0.93      0.78       249

    accuracy                           0.67       399
   macro avg       0.66      0.58      0.56       399
weighted avg       0.66      0.67      0.61       399
```

```
[31]: from sklearn.metrics import roc_auc_score
      roc_auc_score(dependent, grid.predict_proba(independent)[:,1])
```

```
[31]: 0.6322623828647925
```

```
[33]: from sklearn.metrics import f1_score
      f1_score = f1_score(dependent, grid_pred, average = 'weighted')
      print("evaluation metrics {}:".format(grid.best_params_),f1_score)
```

```
evaluation metrics {'priors': None, 'var_smoothing': 0.1}: 0.6141987829614605
```


Home
10.MultinominalNB
2237807-Regression_Assignme
Regre
localhost:8888/notebooks/3.Assignment/10.MultinominalNB.ipynb?
jupyter
10.MultinominalNB
Last Checkpoint: 20 hours ago
File
Edit
View
Run
Kernel
Settings
Help
Code

```
cf_report = classification_report(dependent, grid_pred)
```

```
[20]: print(cf_matrix)
      print(cf_report)
```

```
[[145   5]
 [ 54 195]]
              precision    recall  f1-score   support

           0       0.73      0.97      0.83       150
           1       0.97      0.78      0.87       249

    accuracy                           0.85       399
   macro avg       0.85      0.87      0.85       399
weighted avg       0.88      0.85      0.85       399
```

```
[23]: from sklearn.metrics import roc_auc_score
      roc_auc_score(dependent, grid.predict_proba(independent)[:,1])
```

```
[23]: 0.9499866131191431
```

```
[25]: from sklearn.metrics import f1_score
      f1_score = f1_score(dependent, grid_pred, average = 'weighted')
      print("evaluation metrics {}:".format(grid.best_params_),f1_score)
```

```
evaluation metrics {'alpha': 1.0, 'force_alpha': True}: 0.8544422491701906
```