Assignment-Regression Algorithm

1. The problem statement for this task is to develop a predictive model that can accurately estimate the **chronic kidney disease** (CKD) based on several input parameters.

2. The total number of rows of the dataset is 399 and the total number of columns is 25. For the model, Classification ais taken as a predictive value or output parameter and the remaining attributes are the input parameters.

3. I'm using One Hot Encoding method to convert the categorical (nominal) data, sg, rbc, pc, pcc, ba, htn, dm, cad, appet, pe, ane, and classification into numerical values.

4. Finalized Model 1:

   Logistic Regression - Phase 01 - Model Creation
   https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Final%20Model.ipynb

   Logistic Regression - Phase 02 - Deployment
   https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Deployment-Final%20Model.ipynb

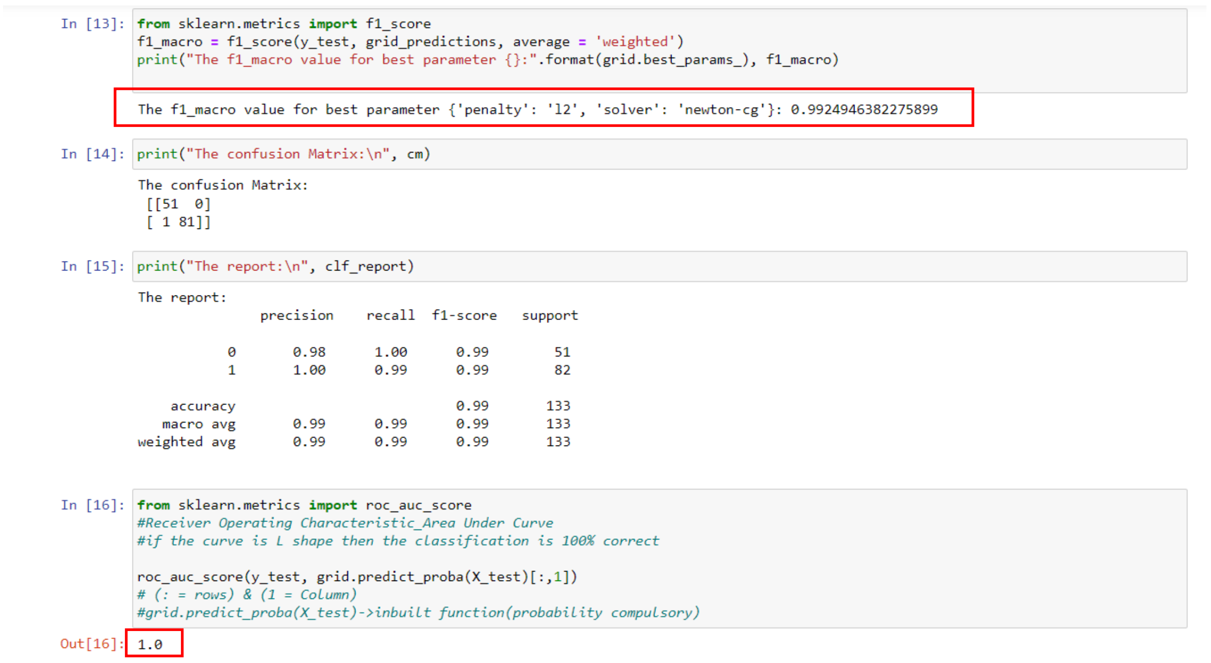
Finalized Model 1:

SVM Classification - Phase 01 - Model Creation
https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Final%20Model.ipynb

SVM Classification - Phase 02 - Deployment
https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Deployment-Final%20Model.ipynb

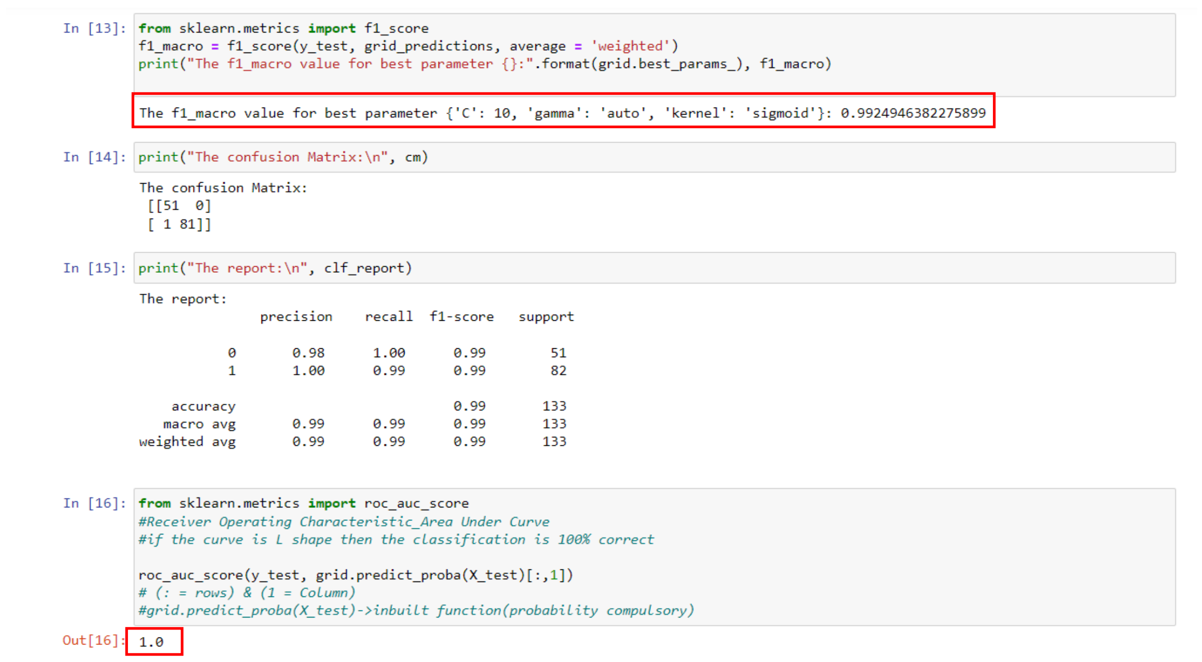
5.

Logistic Regression



```
In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, 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

In [14]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[51  0]
          [ 1 81]]

In [15]: print("The report:\n", clf_report)

         The 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

In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 1.0
```

SVM Classification



```
In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, 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

In [14]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[51  0]
          [ 1 81]]

In [15]: print("The report:\n", clf_report)

         The 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

In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 1.0
```

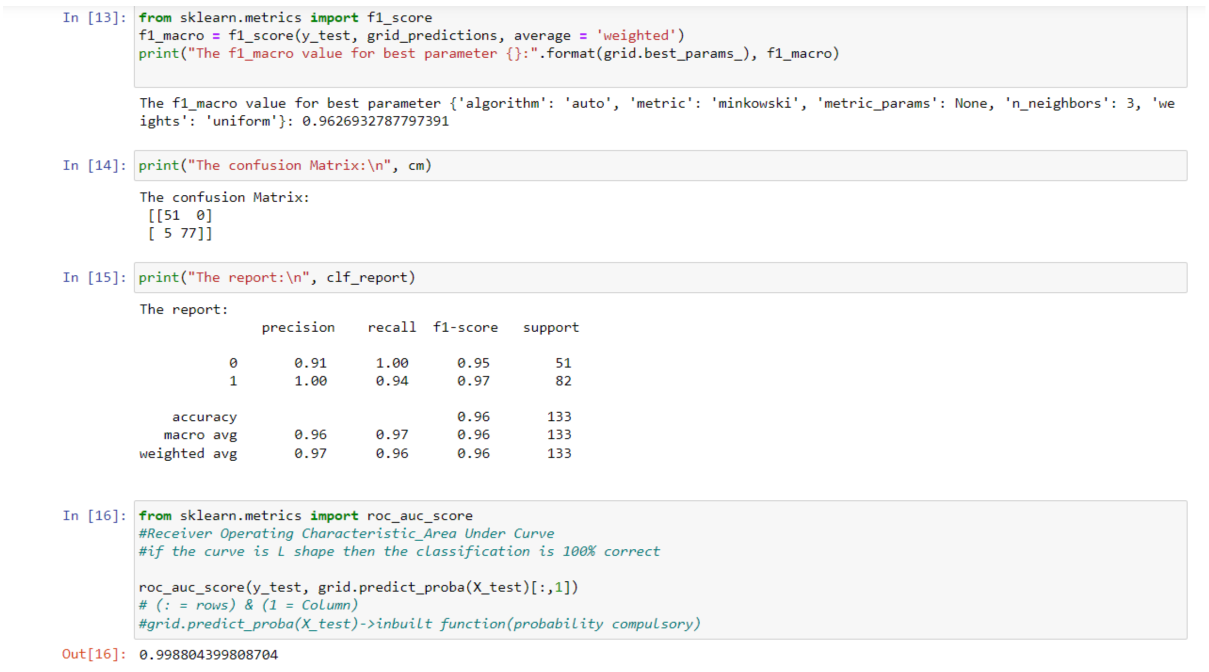
Decision Tree Classification

```
In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, average = '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', 'splitter': 'random'}: 0.9624731911379498

In [14]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[49  2]
          [ 3 79]]

In [15]: print("The report:\n", clf_report)

         The report:
                       precision    recall  f1-score   support

                    0       0.94      0.96      0.95        51
                    1       0.98      0.96      0.97        82

             accuracy                           0.96       133
            macro avg       0.96      0.96      0.96       133
         weighted avg       0.96      0.96      0.96       133


In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 0.9620994739359159
```

Random Forest Classification

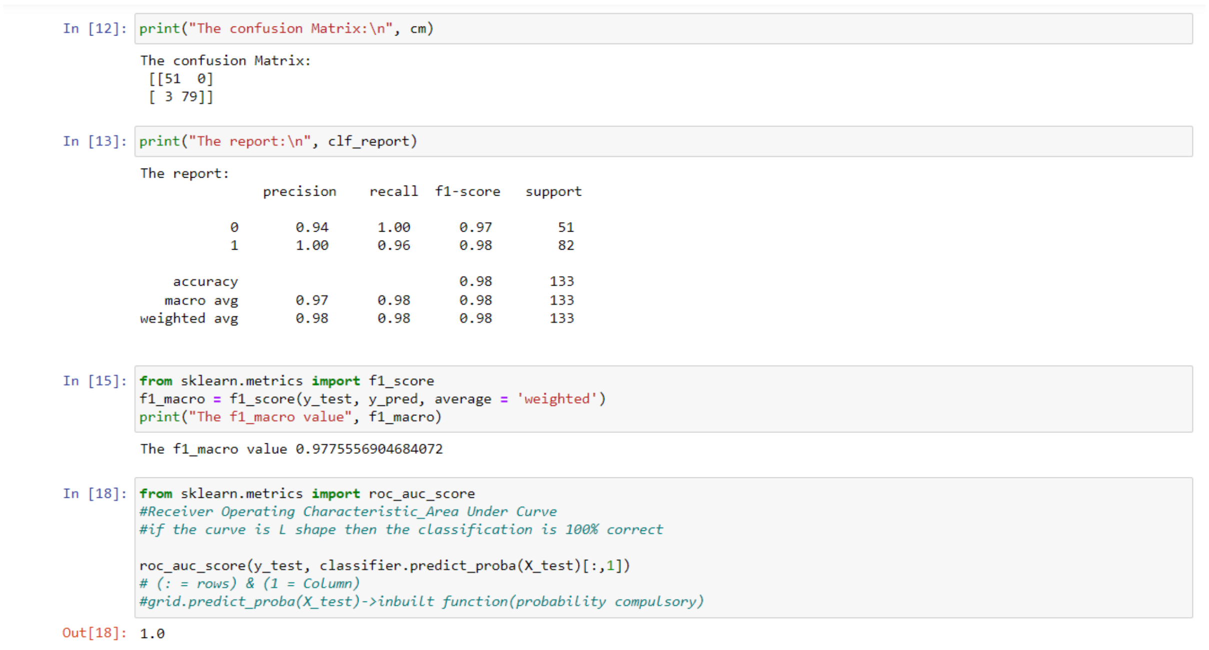
```
In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, average = 'weighted')
         print("The f1_macro value for best parameter {}:".format(grid.best_params_), f1_macro)

         The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'sqrt', 'n_estimators': 100}: 0.9849624060150376

In [14]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[50  1]
          [ 1 81]]

In [15]: print("The report:\n", clf_report)

         The 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


In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 0.9997608799617408
```

K – Nearest Neighbor Classification



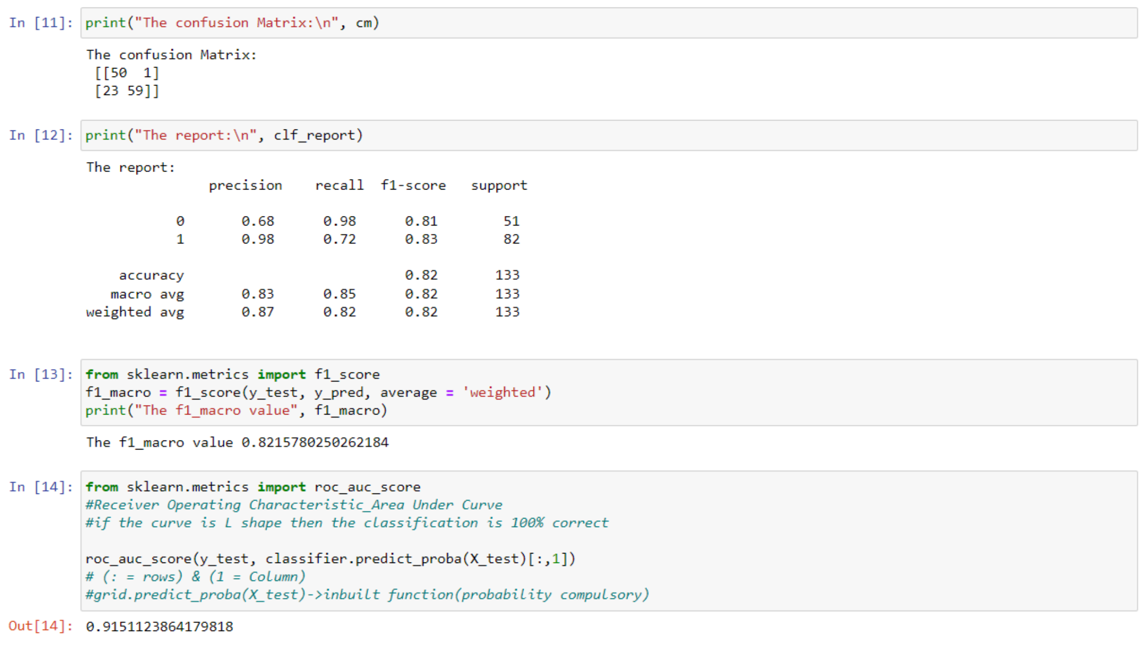
```
In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, average = 'weighted')
         print("The f1_macro value for best parameter {}:".format(grid.best_params_), f1_macro)

         The f1_macro value for best parameter {'algorithm': 'auto', 'metric': 'minkowski', 'metric_params': None, 'n_neighbors': 3, 'we
         ights': 'uniform'}: 0.9626932787797391

In [14]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[51  0]
          [ 5 77]]

In [15]: print("The report:\n", clf_report)

         The 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


In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 0.998804399808704
```

Navies' Bayes – Gaussian Classification



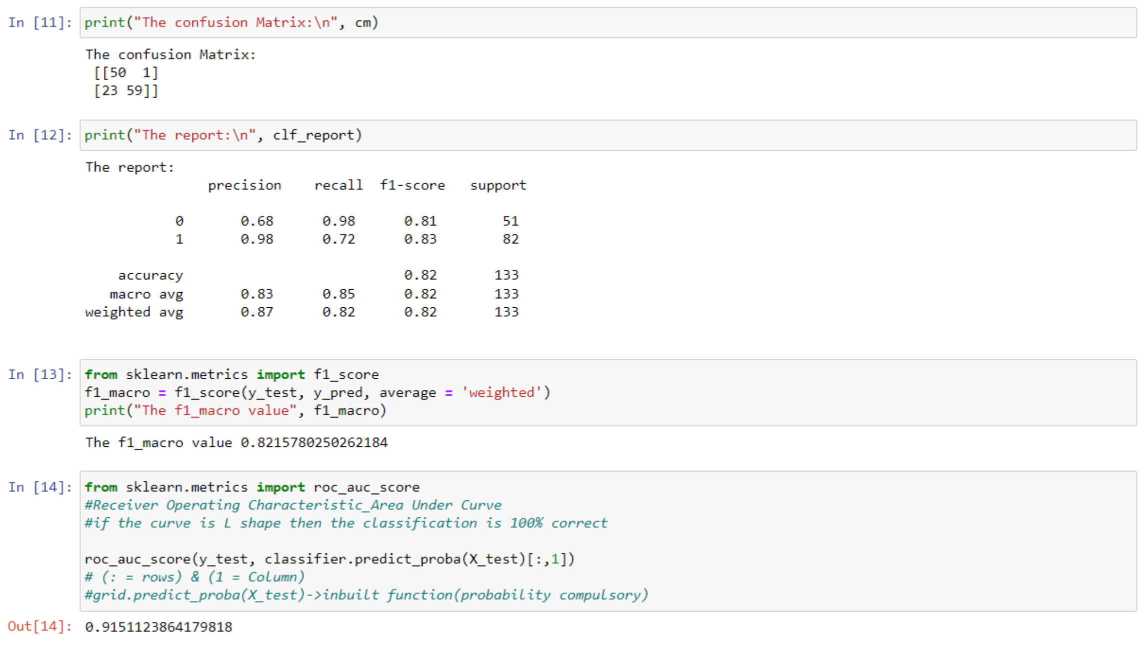
```
In [12]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[51  0]
          [ 3 79]]

In [13]: print("The report:\n", clf_report)

         The 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


In [15]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, y_pred, average = 'weighted')
         print("The f1_macro value", f1_macro)

         The f1_macro value 0.9775556904684072

In [18]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, classifier.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[18]: 1.0
```

Navies' Bayes – Multinomial Classification



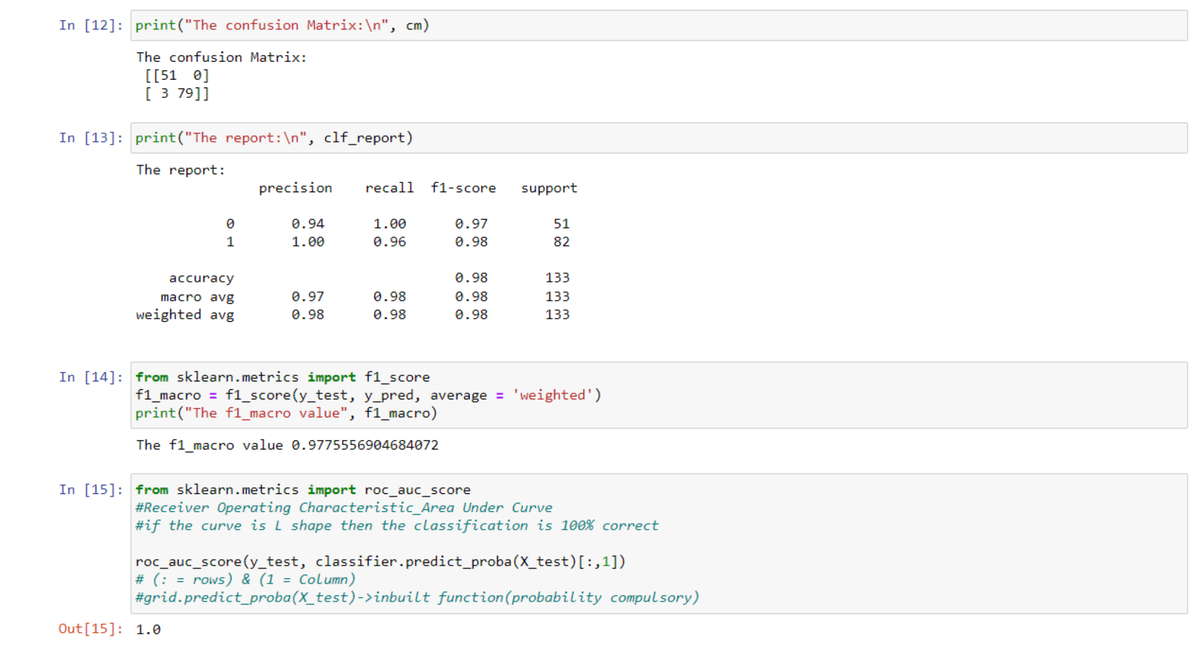
```
In [11]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[50  1]
          [23 59]]

In [12]: print("The report:\n", clf_report)

         The report:
                       precision    recall  f1-score   support

                    0       0.68      0.98      0.81        51
                    1       0.98      0.72      0.83        82

             accuracy                           0.82       133
            macro avg       0.83      0.85      0.82       133
         weighted avg       0.87      0.82      0.82       133


In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, y_pred, average = 'weighted')
         print("The f1_macro value", f1_macro)

         The f1_macro value 0.8215780250262184

In [14]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, classifier.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[14]: 0.9151123864179818
```

Navies' Bayes – Complement Classification



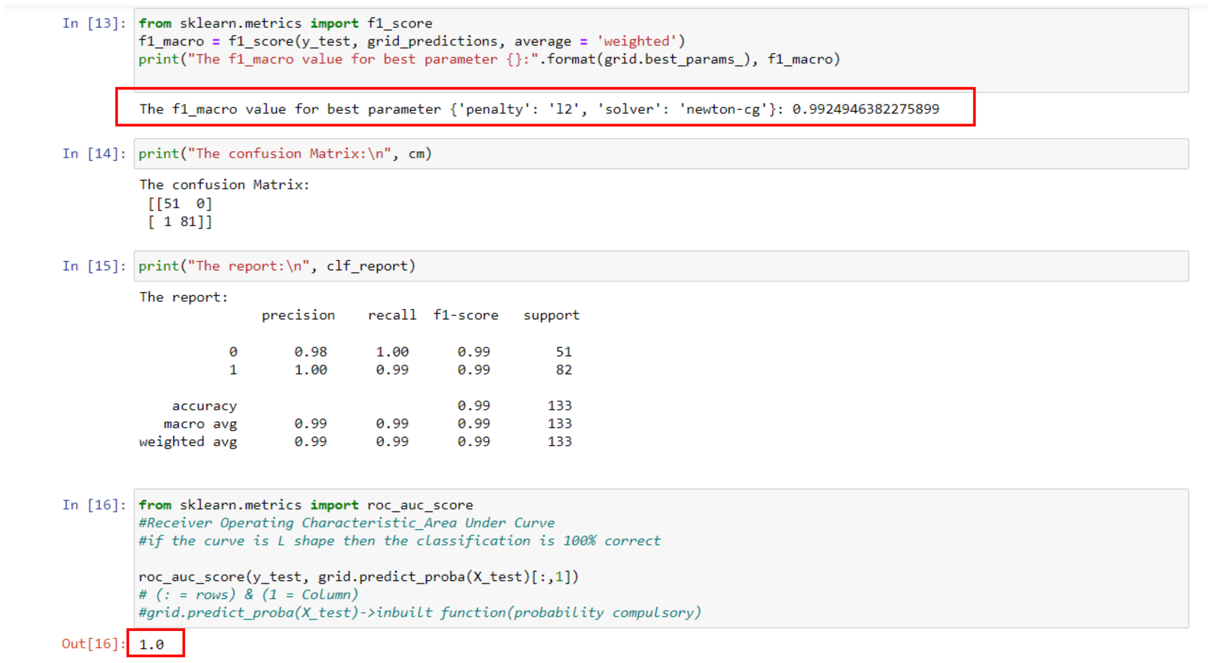
```
In [11]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[50  1]
          [23 59]]

In [12]: print("The report:\n", clf_report)

         The report:
                       precision    recall  f1-score   support

                    0       0.68      0.98      0.81        51
                    1       0.98      0.72      0.83        82

             accuracy                           0.82       133
            macro avg       0.83      0.85      0.82       133
         weighted avg       0.87      0.82      0.82       133


In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, y_pred, average = 'weighted')
         print("The f1_macro value", f1_macro)

         The f1_macro value 0.8215780250262184

In [14]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, classifier.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[14]: 0.9151123864179818
```

Navies’ Bayes – Bernoulli Classification



```
In [12]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[51  0]
          [ 3 79]]

In [13]: print("The report:\n", clf_report)

         The 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


In [14]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, y_pred, average = 'weighted')
         print("The f1_macro value", f1_macro)

         The f1_macro value 0.9775556904684072

In [15]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, classifier.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[15]: 1.0
```

6. Final Model



```
In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, 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

In [14]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[51  0]
          [ 1 81]]

In [15]: print("The report:\n", clf_report)

         The 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


In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 1.0
```

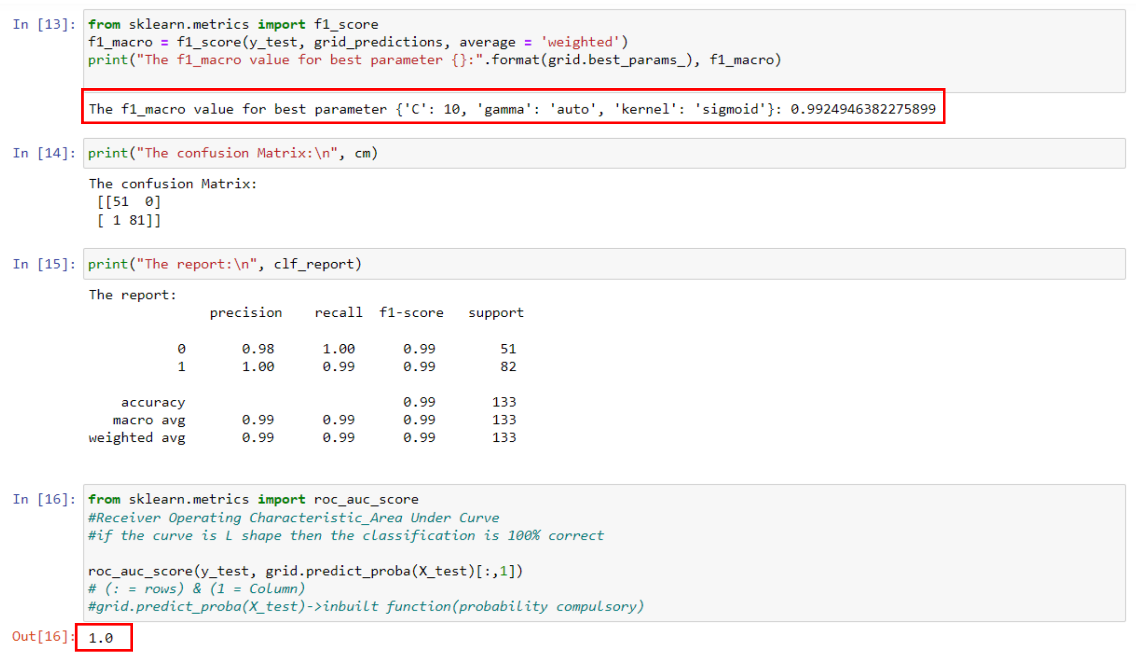
Model 1: Logistic Regression

Logistic Regression - Phase 01 - Model Creation

https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Final%20Model.ipynb

Logistic Regression - Phase 02 - Deployment

https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-Logistic-Deployment-Final%20Model.ipynb



```
In [13]: from sklearn.metrics import f1_score
         f1_macro = f1_score(y_test, grid_predictions, 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

In [14]: print("The confusion Matrix:\n", cm)

         The confusion Matrix:
          [[51  0]
          [ 1 81]]

In [15]: print("The report:\n", clf_report)

         The 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


In [16]: from sklearn.metrics import roc_auc_score
         #Receiver Operating Characteristic_Area Under Curve
         #if the curve is L shape then the classification is 100% correct

         roc_auc_score(y_test, grid.predict_proba(X_test)[:,1])
         # (: = rows) & (1 = Column)
         #grid.predict_proba(X_test)->inbuilt function(probability compulsory)

Out[16]: 1.0
```

Model 2: SVM Classification

SVM Classification - Phase 01 - Model Creation
https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Final%20Model.ipynb

SVM Classification - Phase 02 - Deployment

https://github.com/GauthamOfficial/2.Machine-Learning/blob/f344bfba992f5c5934d34509cde6da16d925b9a5/2.Machine%20Learning%20Classification/Assignments/Final%20Assignment/Grid-SVM-Deployment-Final%20Model.ipynb

These two models are the final models, because both models have the same and highest **Precision** value, **Recall** value, **F1-score** value, **Accuracy** value, **Macro average** value and **Weighted average** value. Especially both models have the highest Receiver Operating Characteristic Area Under Curve Score (**roc_auc_score**).