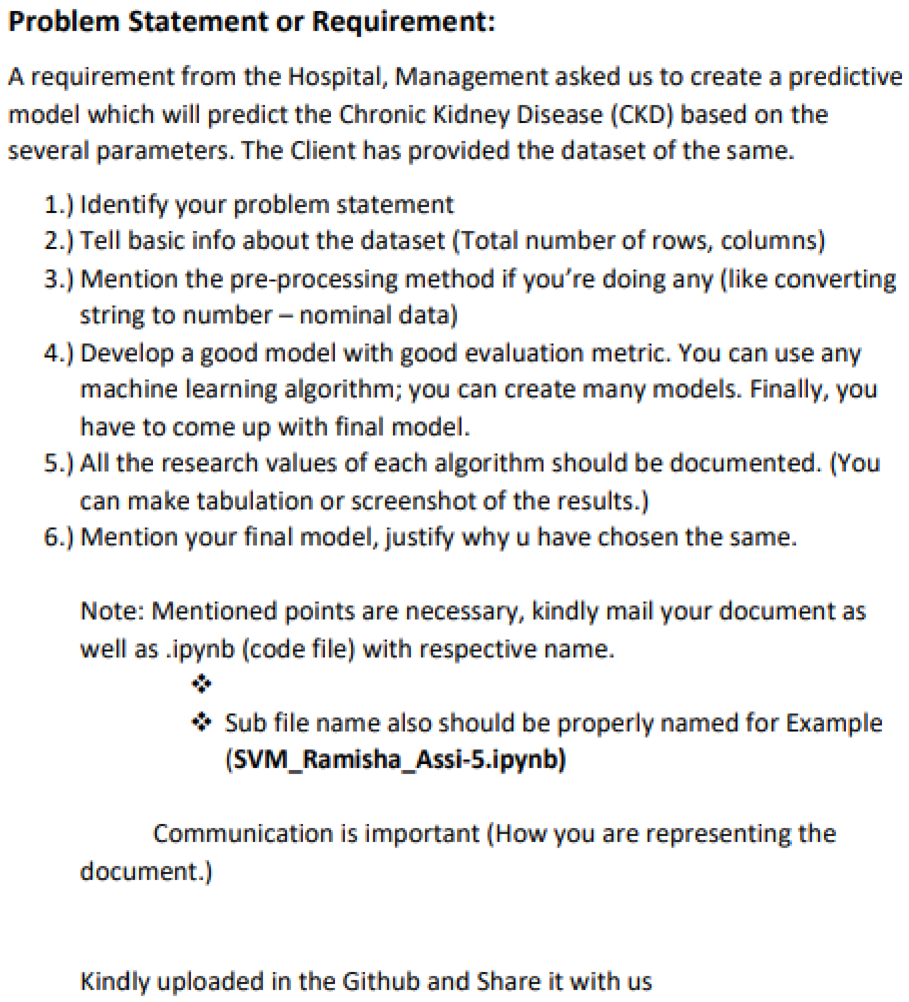
HOPE ARTIFICIAL INTELLIGENCE

MACHINE LEARNING CLASSIFICATION ASSIGNMENT QUESTIONS



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
2.) Tell basic info about the dataset (Total number of rows, columns)
3.) Mention the pre-processing method if you're doing any (like converting string to number – nominal data)
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.
5.) All the research values of each algorithm should be documented. (You can make tabulation or screenshot of the results.)
6.) Mention your final model, justify why u have chosen the same.

Note: Mentioned points are necessary, kindly mail your document as well as .ipynb (code file) with respective name.

❖

❖ Sub file name also should be properly named for Example **(SVM_Ramisha_Assi-5.ipynb)**

Communication is important (How you are representing the document.)

Kindly uploaded in the Github and Share it with us

MACHINE LEARNING CLASSIFICATION ASSIGNMENT ANSWERS

1. Identify your Problem Statement
   - Given dataset (Input & Output) has Numerical Values.Hence it is **Machine Learning** under **Supervised learning** process.
   - Given dataset (Output) has categorical data.So it comes under **Classification** Algorithm.

   The Client requests to make a model for Chronic Kidney Disease (CKD) prediction by using the given dataset.

   This model name shall be **" CKD CARE BY USING AI"**

2. Tell basic info about the dataset (Total no of rows and columns) – Rows=399 , Columns=25

3. Mention the preprocessing method if you are doing any (like converting string to number-nominal data)

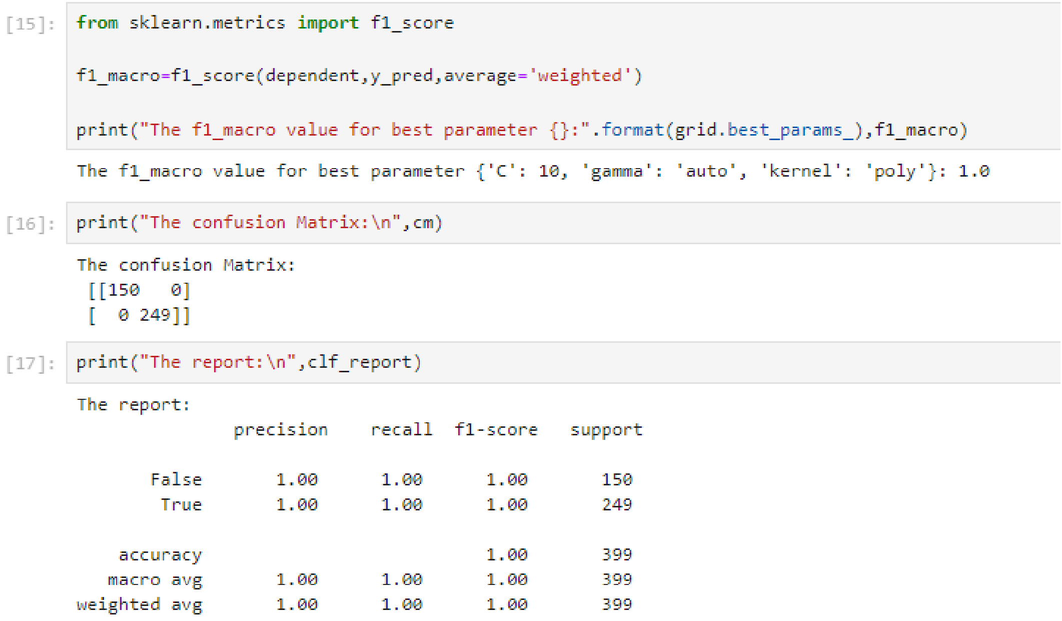
   Nominal data **One Hot Coding** is used for the conversion of string to number.

4. Develop a good model with good conversion metric.You can use any machine learning algorithm;you can create any models.

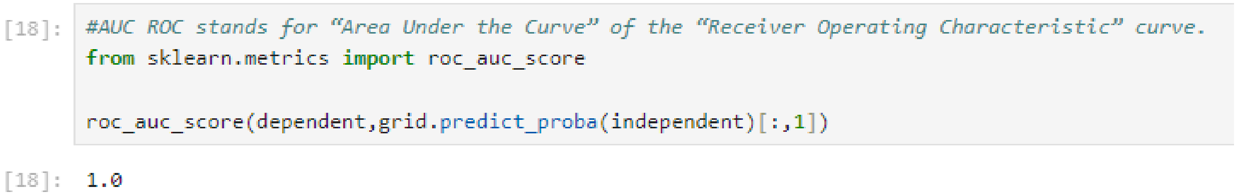
I have listed down the Machine learning classification algorithms that I have used to create a model for the subject (CKD Prediction).

1. SVM Classification Grid
2. Decision Tree Classification Grid
3. Random Forest Classification Grid
4. Logistic Regression Classification Grid
5. KNN Classification Grid
6. Naïve Bayes Gaussian Classification Grid
7. Naïve Bayes Multinomial Classification Grid
8. Naïve Bayes Complement Classification Grid
9. Naïve Bayes Bernoulli Classification Grid
10. Naïve Bayes Categorical Classification Grid

1. SVM CLASSIFICATION VALUES:

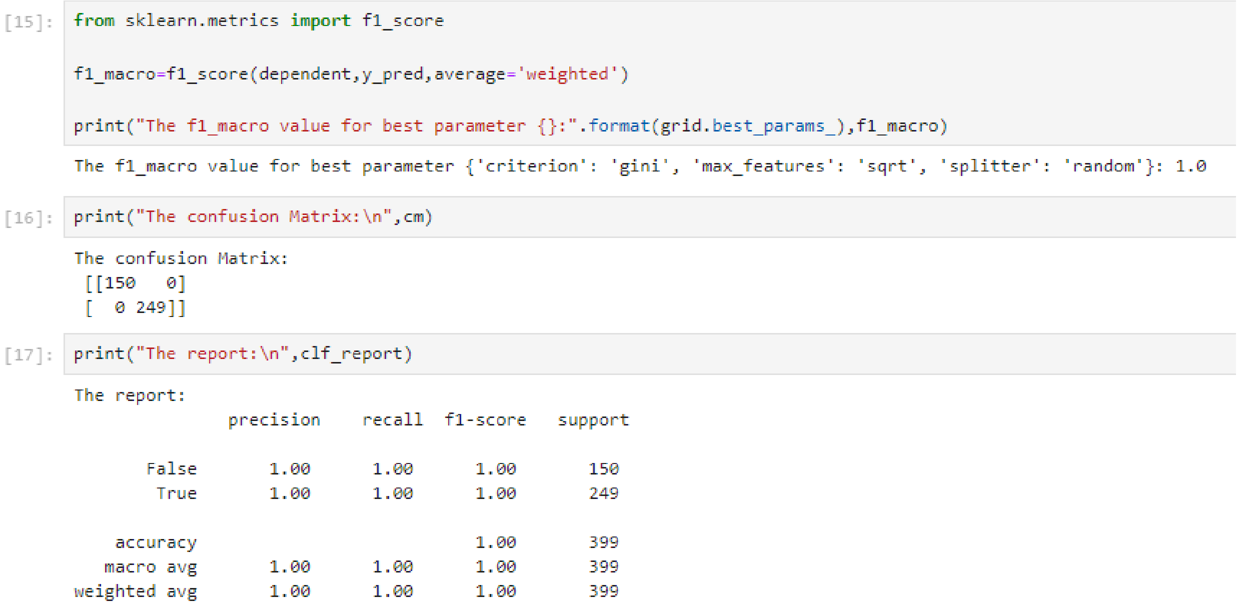


```
[15]: from sklearn.metrics import f1_score

      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 {'C': 10, 'gamma': 'auto', 'kernel': 'poly'}: 1.0

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

      The confusion Matrix:
       [[150   0]
       [  0 249]]

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

      The report:
                     precision    recall  f1-score   support

             False       1.00      1.00      1.00       150
              True       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
```

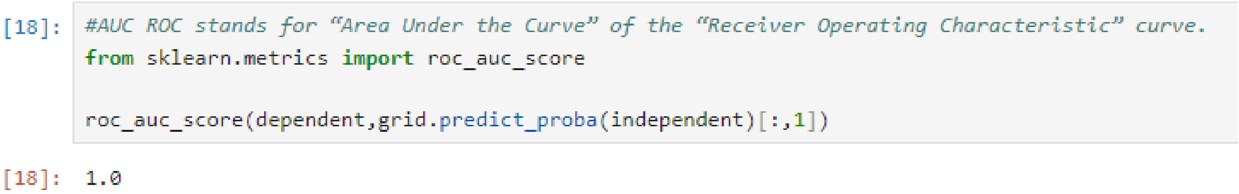


```
[18]: #AUC ROC stands for “Area Under the Curve” of the “Receiver Operating Characteristic” curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[18]: 1.0
```

2. DECISION TREE CLASSIFICATION VALUES:

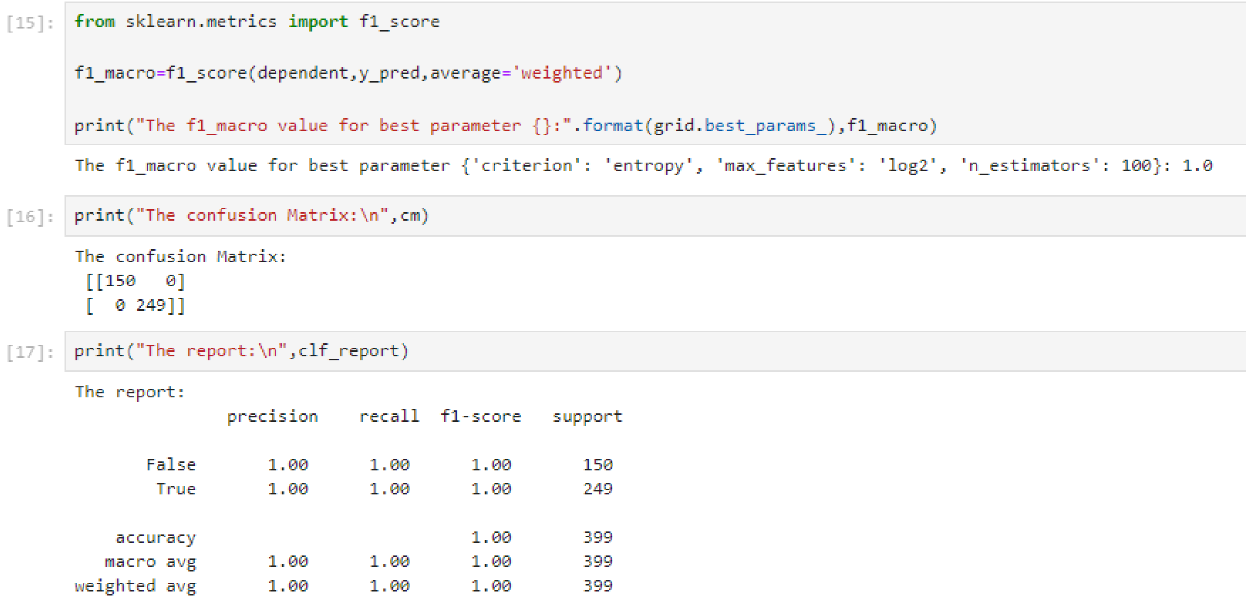


```
[15]: from sklearn.metrics import f1_score

      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': 'gini', 'max_features': 'sqrt', 'splitter': 'random'}: 1.0

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

      The confusion Matrix:
       [[150   0]
       [  0 249]]

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

      The report:
                    precision    recall  f1-score   support

             False       1.00      1.00      1.00       150
              True       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
```

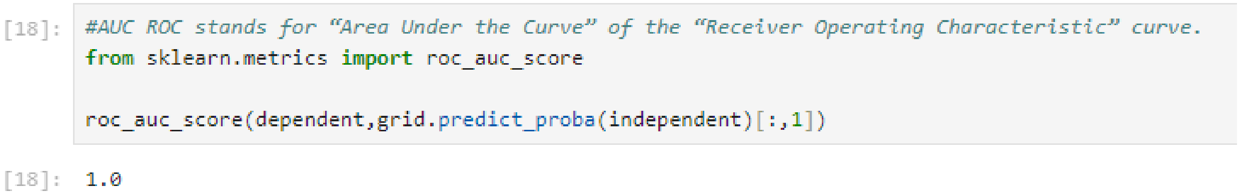


```
[18]: #AUC ROC stands for “Area Under the Curve” of the “Receiver Operating Characteristic” curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[18]: 1.0
```

3. RANDOM FOREST CLASSIFICATION VALUES:

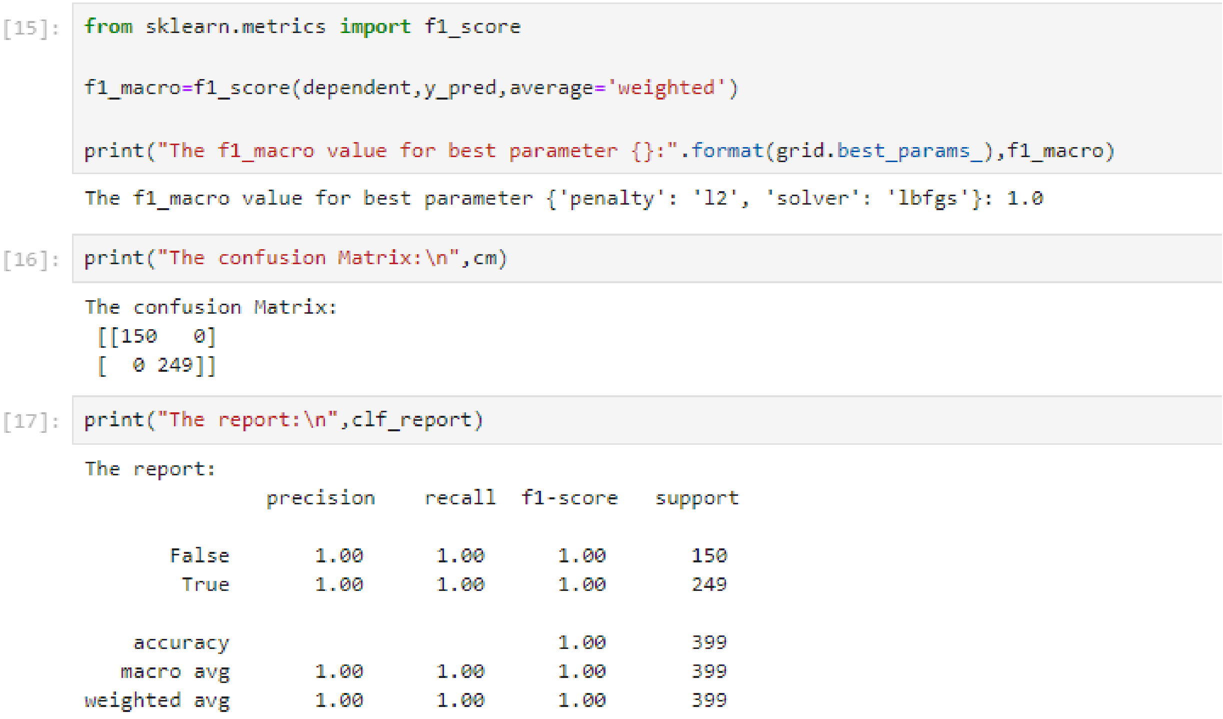


```
[15]: from sklearn.metrics import f1_score

      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', 'max_features': 'log2', 'n_estimators': 100}: 1.0

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

      The confusion Matrix:
       [[150   0]
       [  0 249]]

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

      The report:
                     precision    recall  f1-score   support

             False       1.00      1.00      1.00       150
              True       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
```

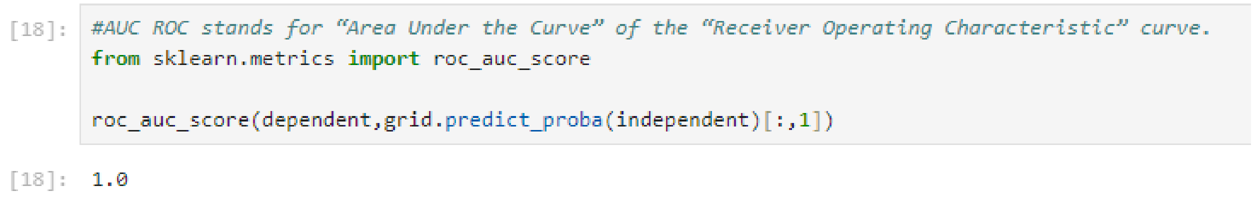


```
[18]: #AUC ROC stands for “Area Under the Curve” of the “Receiver Operating Characteristic” curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[18]: 1.0
```

4. LOGISTIC REGRESSION CLASSIFICATION VALUES:

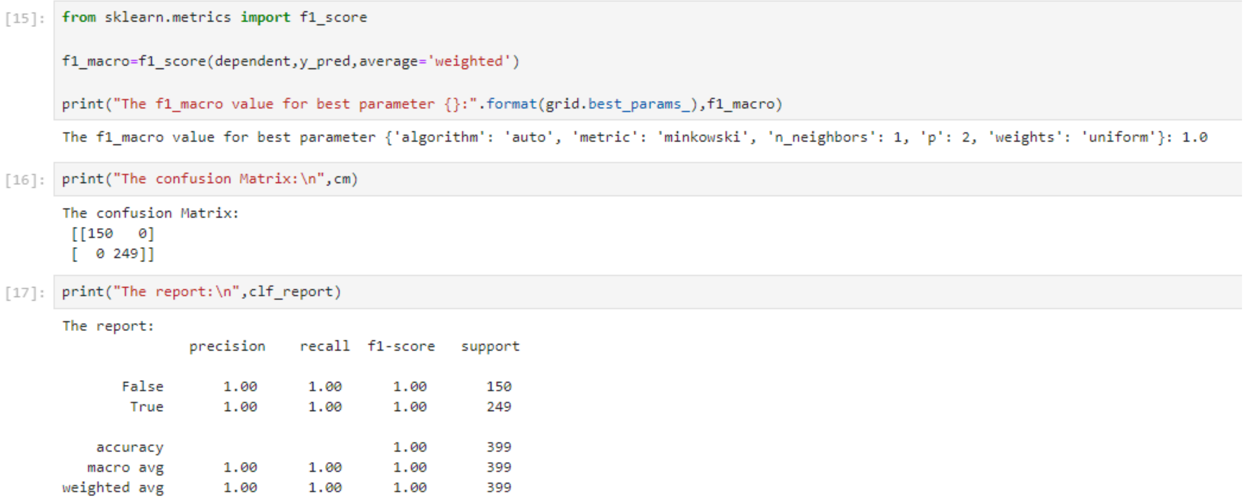


```
[15]: from sklearn.metrics import f1_score

      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 {'penalty': 'l2', 'solver': 'lbfgs'}: 1.0

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

      The confusion Matrix:
       [[150   0]
       [  0 249]]

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

      The report:
                     precision    recall  f1-score   support

             False       1.00      1.00      1.00       150
              True       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
```

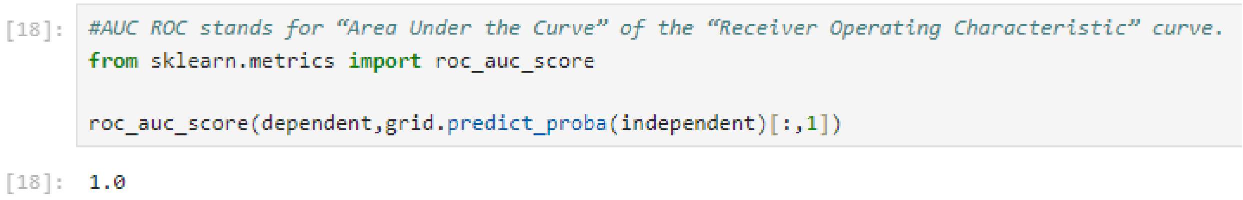


```
[18]: #AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[18]: 1.0
```

5. K-NEAREST NEIGHBOURS (KNN) CLASSIFICATION VALUES:

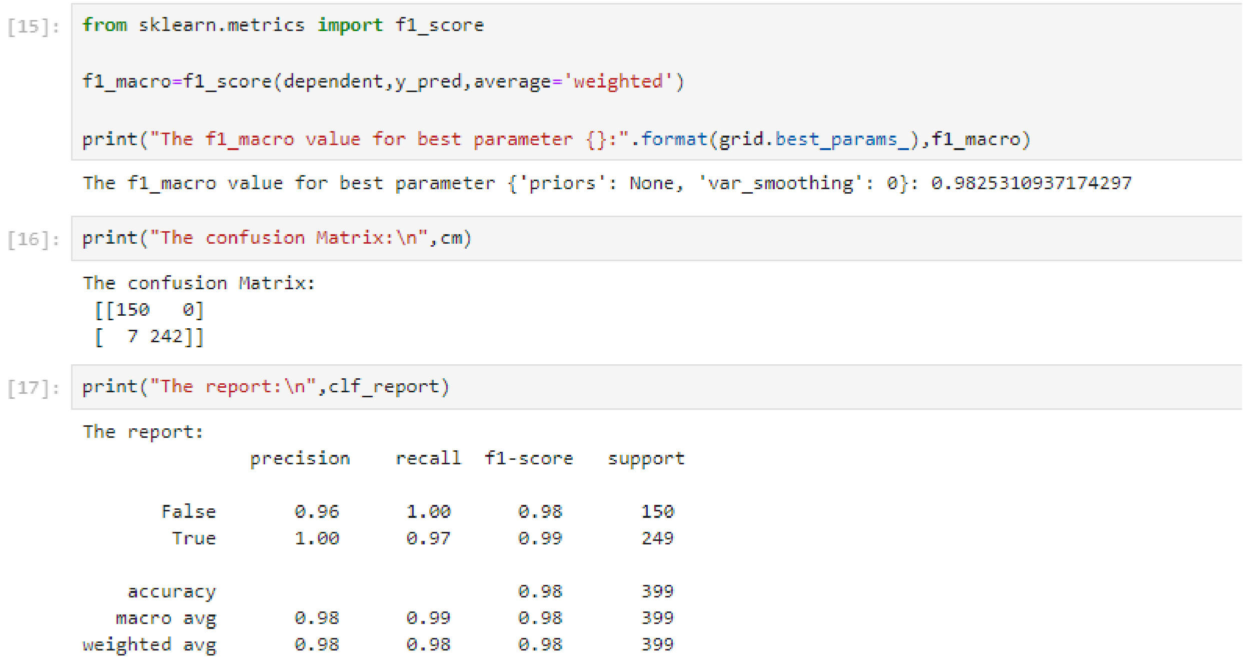


```
[15]: from sklearn.metrics import f1_score

      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 {'algorithm': 'auto', 'metric': 'minkowski', 'n_neighbors': 1, 'p': 2, 'weights': 'uniform'}: 1.0

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

      The confusion Matrix:
       [[150   0]
       [  0 249]]

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

      The report:
                    precision    recall  f1-score   support

             False       1.00      1.00      1.00       150
              True       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
```

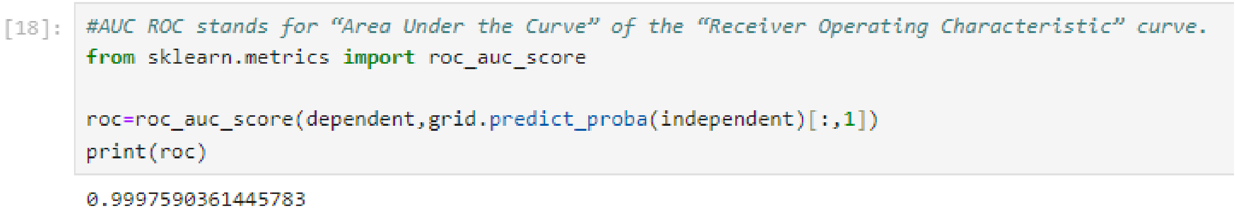


```
[18]: #AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[18]: 1.0
```

6. NAÏVE BAYES GAUSSIAN CLASSIFICATION VALUES:

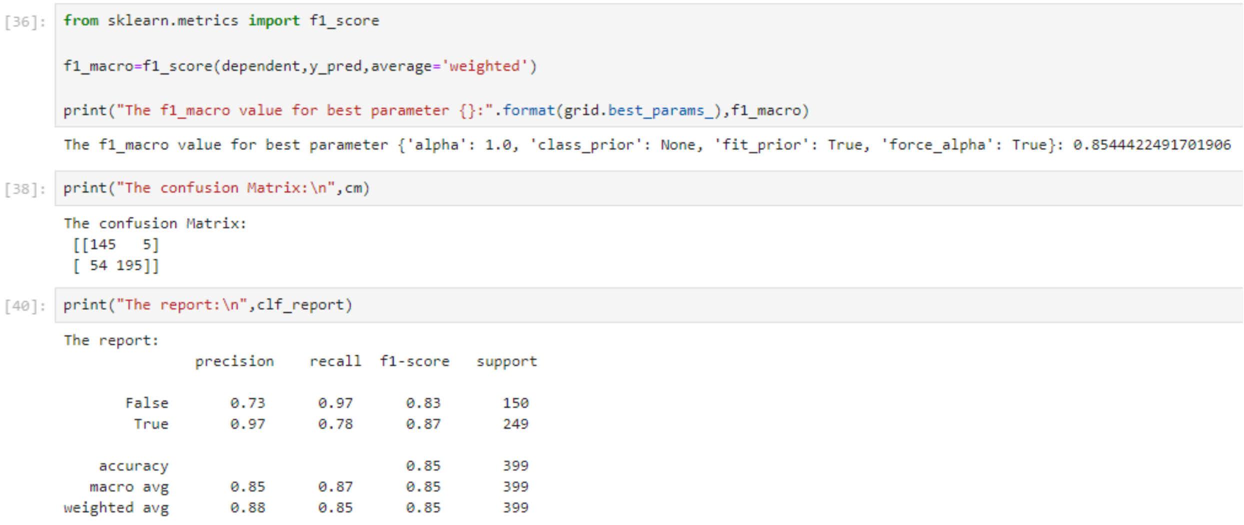


```
[15]: from sklearn.metrics import f1_score

      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 {'priors': None, 'var_smoothing': 0}: 0.9825310937174297

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

      The confusion Matrix:
       [[150   0]
       [  7 242]]

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

      The report:
                    precision    recall  f1-score   support

             False       0.96      1.00      0.98       150
              True       1.00      0.97      0.99       249

          accuracy                           0.98       399
         macro avg       0.98      0.99      0.98       399
      weighted avg       0.98      0.98      0.98       399
```

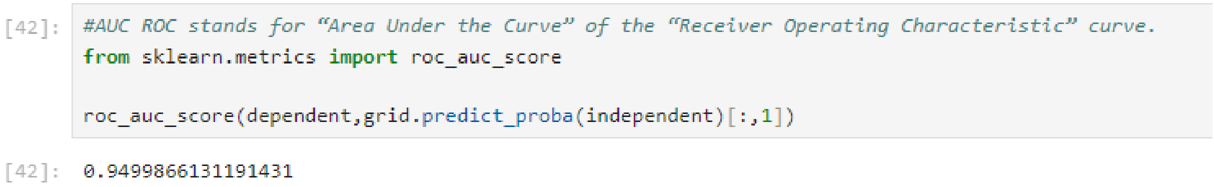


```
[18]: #AUC ROC stands for “Area Under the Curve” of the “Receiver Operating Characteristic” curve.
      from sklearn.metrics import roc_auc_score

      roc=roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
      print(roc)

      0.9997590361445783
```

7. NAÏVE BAYES MULTINOMIAL CLASSIFICATION VALUES:

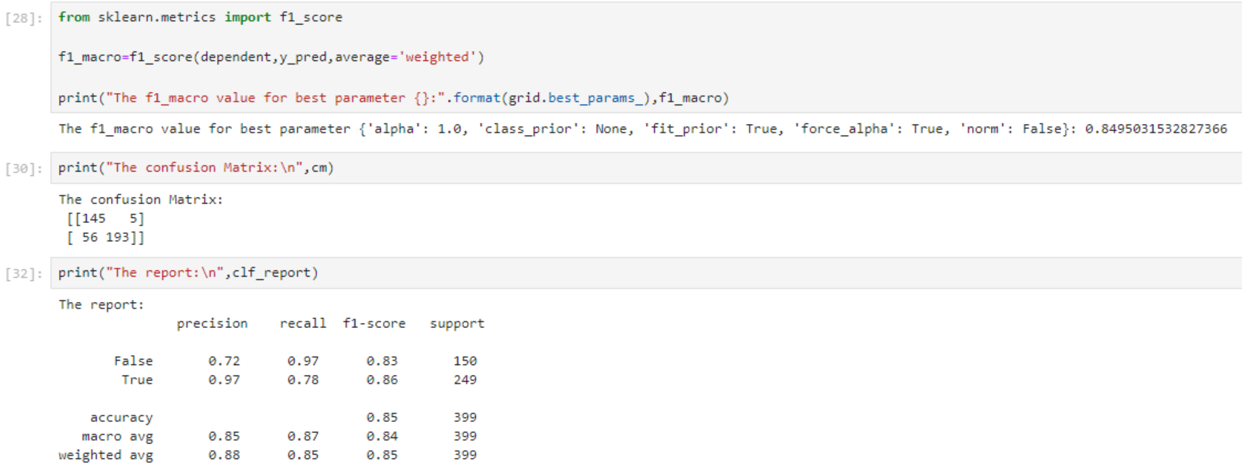


```
[36]: from sklearn.metrics import f1_score

      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 {'alpha': 1.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True}: 0.8544422491701906

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

      The confusion Matrix:
       [[145   5]
       [ 54 195]]

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

      The report:
                    precision    recall  f1-score   support

             False       0.73      0.97      0.83       150
              True       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
```

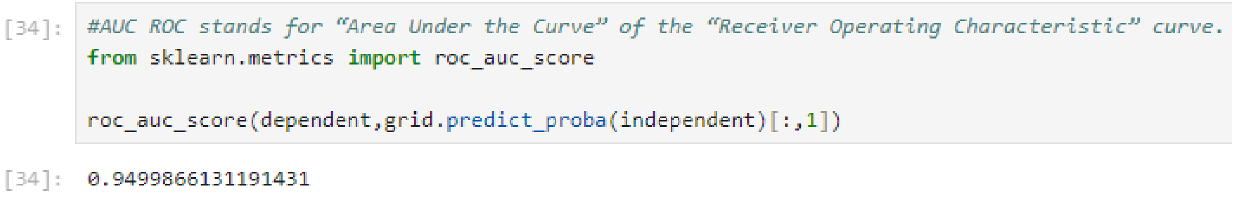


```
[42]: #AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[42]: 0.9499866131191431
```

8. NAÏVE BAYES COMPLEMENT CLASSIFICATION VALUES:

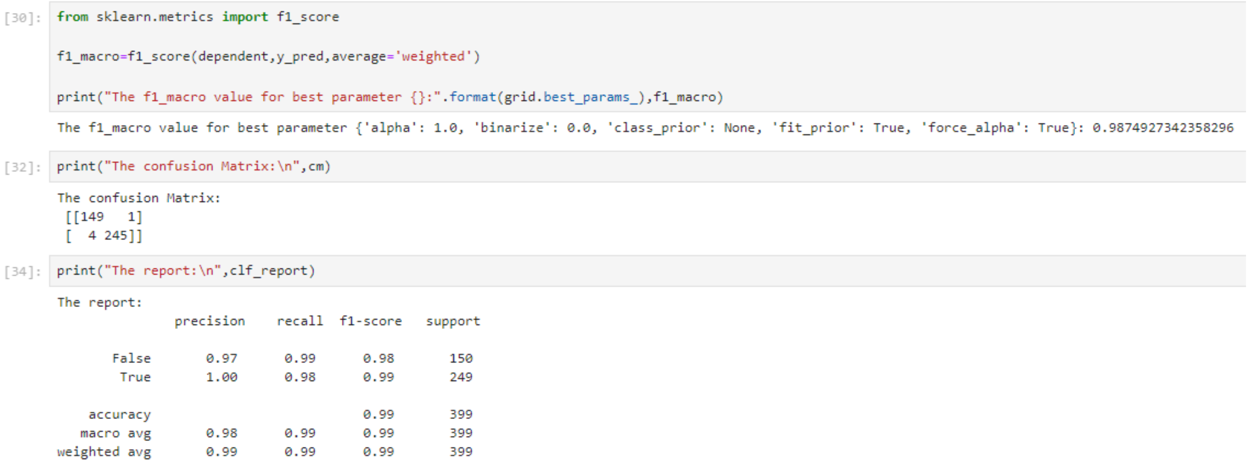


```
[28]: from sklearn.metrics import f1_score

      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 {'alpha': 1.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True, 'norm': False}: 0.8495031532827366

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

      The confusion Matrix:
       [[145   5]
       [ 56 193]]

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

      The report:
                     precision    recall  f1-score   support

             False       0.72      0.97      0.83       150
              True       0.97      0.78      0.86       249

          accuracy                           0.85       399
         macro avg       0.85      0.87      0.84       399
      weighted avg       0.88      0.85      0.85       399
```

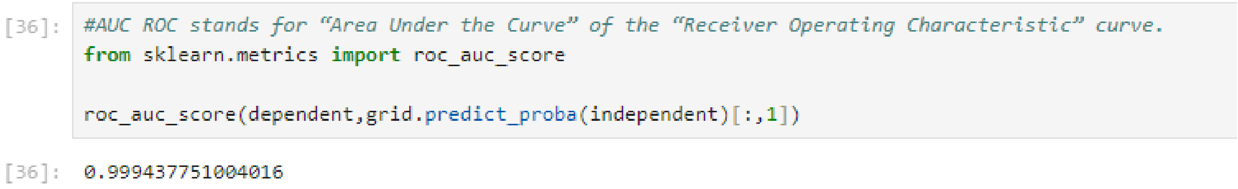


```
[34]: #AUC ROC stands for "Area Under the Curve" of the "Receiver Operating Characteristic" curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[34]: 0.9499866131191431
```

9. NAÏVE BAYES BERNOULLI CLASSIFICATION VALUES:

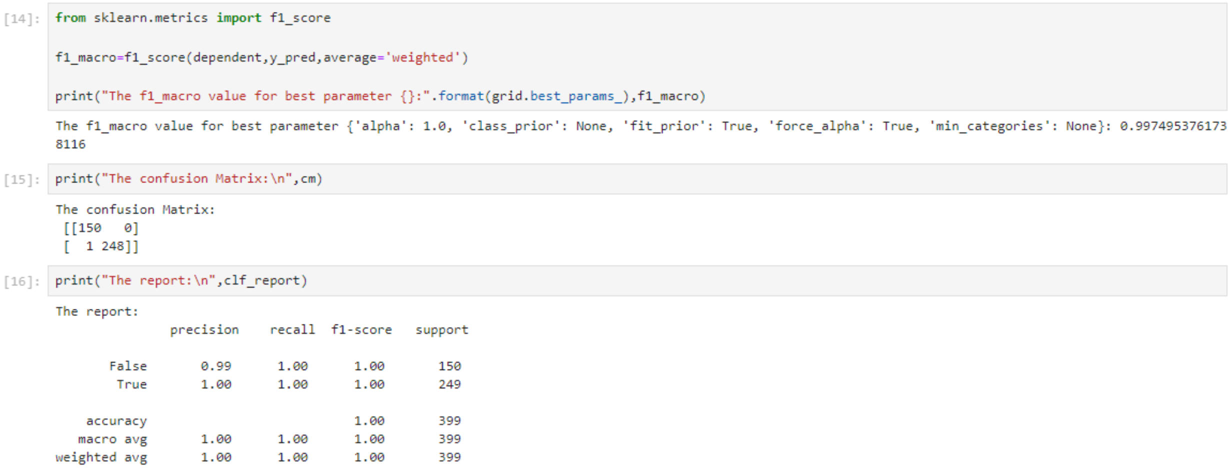


```
[30]: from sklearn.metrics import f1_score

      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 {'alpha': 1.0, 'binarize': 0.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True}: 0.9874927342358296

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

      The confusion Matrix:
       [[149   1]
       [  4 245]]

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

      The report:
                     precision    recall  f1-score   support

             False       0.97      0.99      0.98       150
              True       1.00      0.98      0.99       249

          accuracy                           0.99       399
         macro avg       0.98      0.99      0.99       399
      weighted avg       0.99      0.99      0.99       399
```

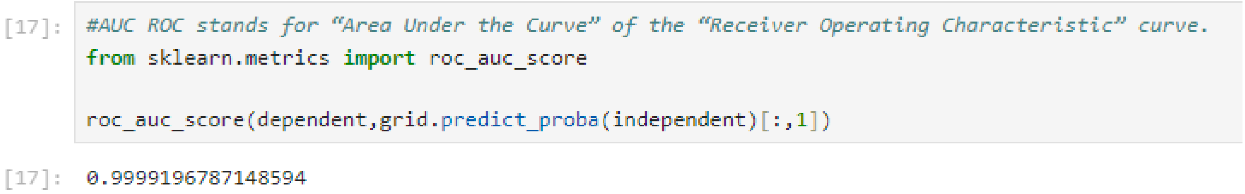


```
[36]: #AUC ROC stands for “Area Under the Curve” of the “Receiver Operating Characteristic” curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[36]: 0.999437751004016
```

10. NAÏVE BAYES CATEGORICAL CLASSIFICATION VALUES:



```
[14]: 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 {'alpha': 1.0, 'class_prior': None, 'fit_prior': True, 'force_alpha': True, 'min_categories': None}: 0.997495376173
      8116

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

      The confusion Matrix:
       [[150   0]
       [  1 248]]

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

      The report:
                     precision    recall  f1-score   support

             False       0.99      1.00      1.00       150
              True       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
```



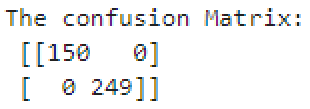
```
[17]: #AUC ROC stands for “Area Under the Curve” of the “Receiver Operating Characteristic” curve.
      from sklearn.metrics import roc_auc_score

      roc_auc_score(dependent,grid.predict_proba(independent)[:,1])

[17]: 0.9999196787148594
```

5. Final Result:

I have got the best results while using SVM, Decision Tree, Random Forest, Logistic Regression & KNN Classification Algorithms for the provided data.

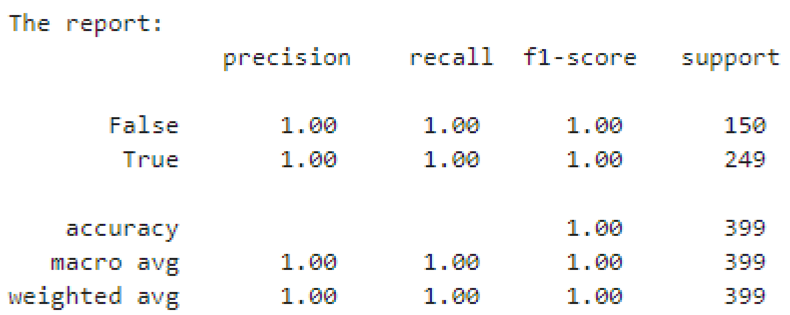
However,I have created model and deployment for Logistic Regression Classification as given below.

Confusion Matrix:



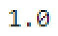
```
The confusion Matrix:
 [[150   0]
 [  0 249]]
```

Classification Report:



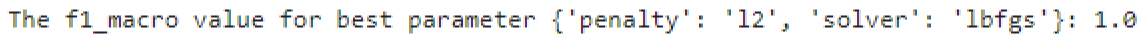
```
The report:
              precision    recall  f1-score   support

       False       1.00      1.00      1.00       150
        True       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
```

’’Area Under the Curve’’ of the “Receiver Operating Characteristic” curve score:



```
1.0
```

The f1 macro value for best parameter:



```
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'lbfgs'}: 1.0
```