Classification Assignment

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

Stage 1: Machine Learning
Stage 2: Supervised Learning
Stage 3: Classification

2.) Tell basic info about the dataset (Total number of rows, columns)

Total number of rows: 400
Total number of columns: 25

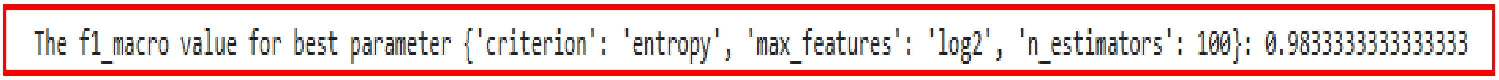
3.) Mention the pre-processing method if you're doing any (like converting string to number – nominal data)

Ordinal Data

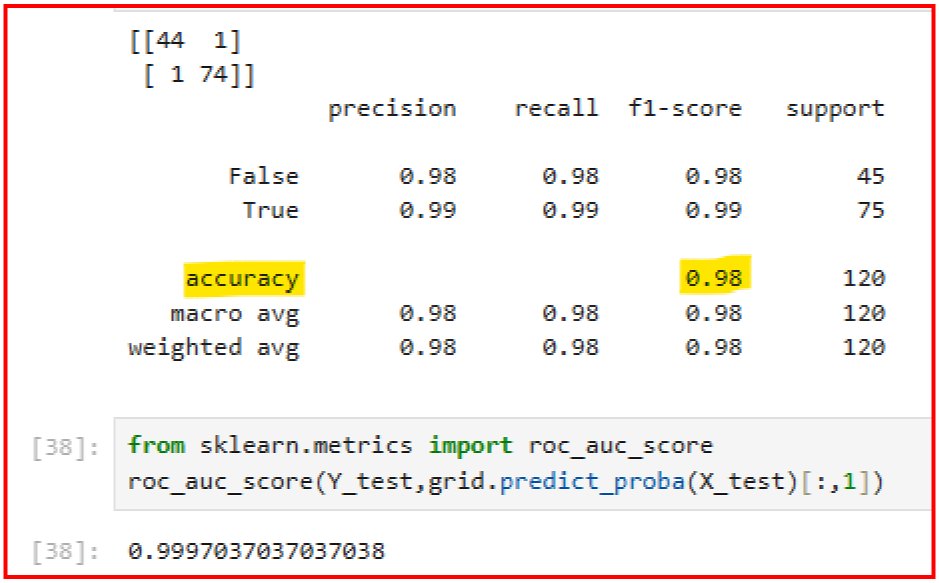
4.) Develop a good model with good evaluation metric. You can use any machine learning algorithm; you can create many models. Finally, you have to come up with final model.

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

1.Random Forest:



```
The f1_macro value for best parameter {'criterion': 'entropy', 'max_features': 'log2', 'n_estimators': 100}: 0.9833333333333333
```

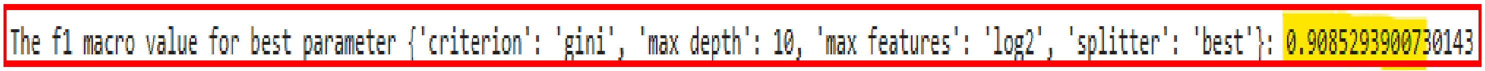


```
[[44  1]
 [ 1 74]]
              precision    recall  f1-score   support

       False       0.98      0.98      0.98        45
        True       0.99      0.99      0.99        75

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

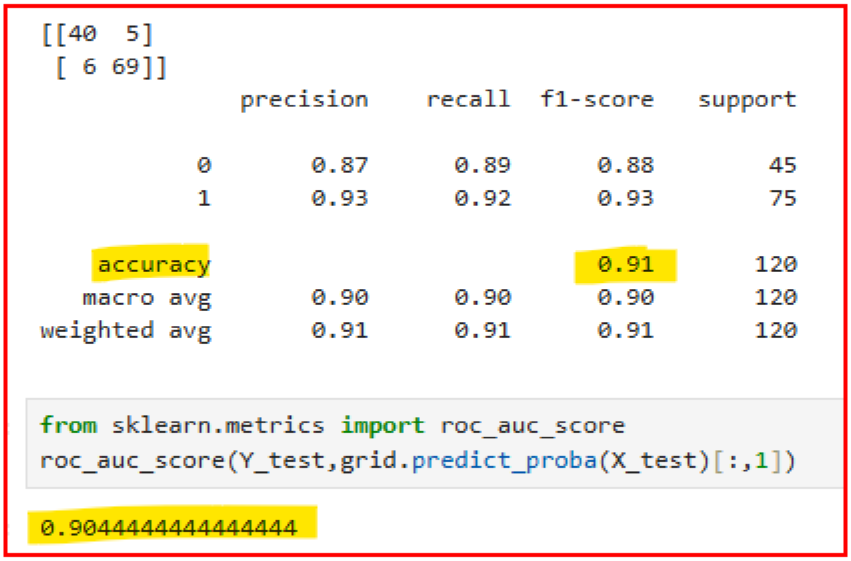
```
[38]: from sklearn.metrics import roc_auc_score
      roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

```
[38]: 0.9997037037037038
```

2.Decision Tree:



```
The f1 macro value for best parameter {'criterion': 'gini', 'max depth': 10, 'max features': 'log2', 'splitter': 'best'}: 0.9085293900730143
```

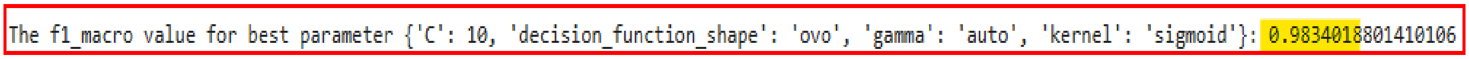


```
[[40  5]
 [ 6 69]]
              precision    recall  f1-score   support

           0       0.87      0.89      0.88        45
           1       0.93      0.92      0.93        75

    accuracy                           0.91       120
   macro avg       0.90      0.90      0.90       120
weighted avg       0.91      0.91      0.91       120
```

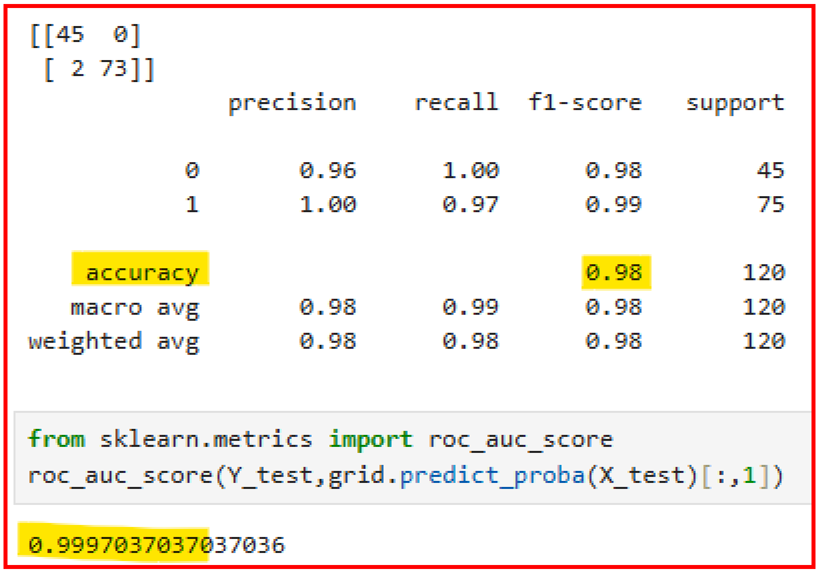
```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

```
0.9044444444444444
```

3.SVM:



```
The f1_macro value for best parameter {'C': 10, 'decision_function_shape': 'ovo', 'gamma': 'auto', 'kernel': 'sigmoid'}: 0.9834018801410106
```

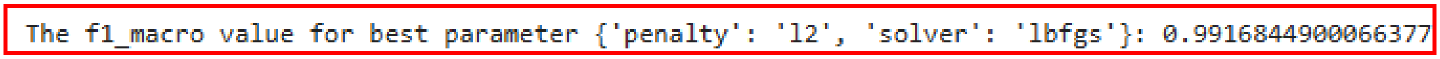


```
[[45  0]
 [ 2 73]]
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        45
           1       1.00      0.97      0.99        75

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

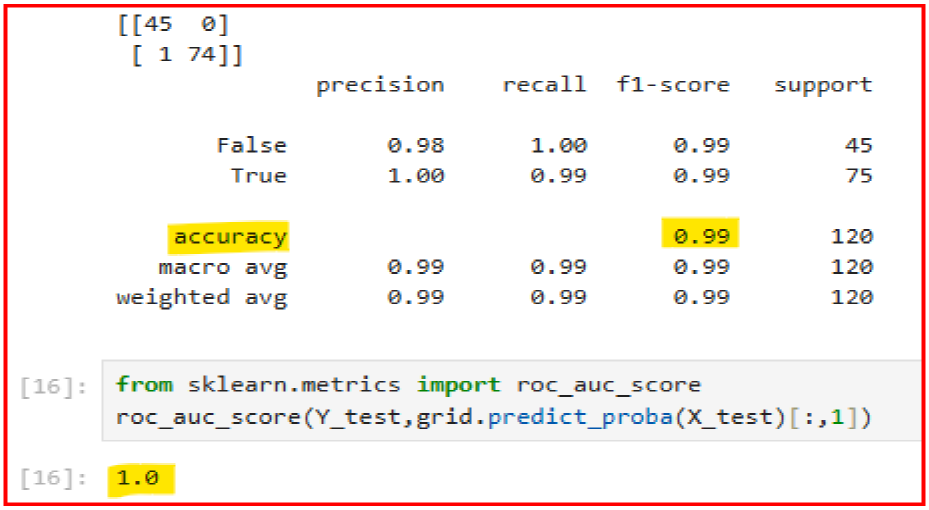
```
from sklearn.metrics import roc_auc_score
roc_auc_score(Y_test,grid.predict_proba(X_test)[:,1])
```

```
0.9997037037037036
```

4.Logistic Regression:

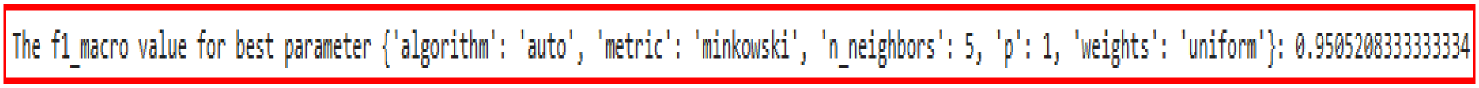


```
The f1_macro value for best parameter {'penalty': 'l2', 'solver': 'lbfgs'}: 0.9916844900066377
```

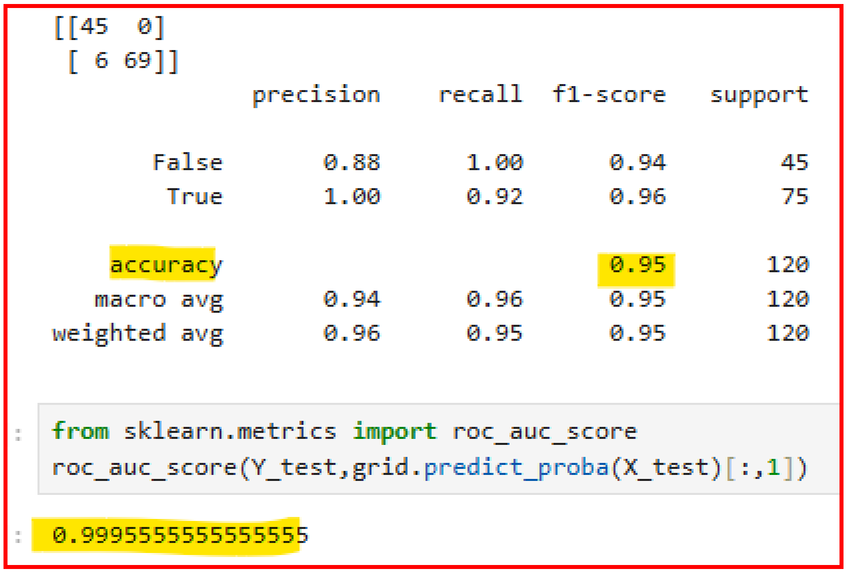


```
[[45  0]
 [ 1 74]]
              precision    recall  f1-score   support

       False       0.98      1.00      0.99        45
        True       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

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

[16]: 1.0
```

5.KNN:



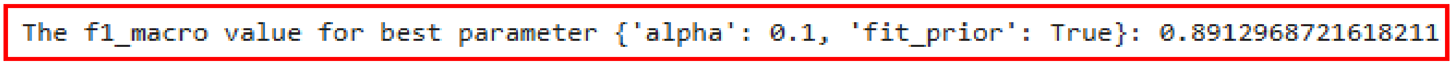
```
The f1_macro value for best parameter {'algorithm': 'auto', 'metric': 'minkowski', 'n_neighbors': 5, 'p': 1, 'weights': 'uniform'}: 0.9505208333333334
```



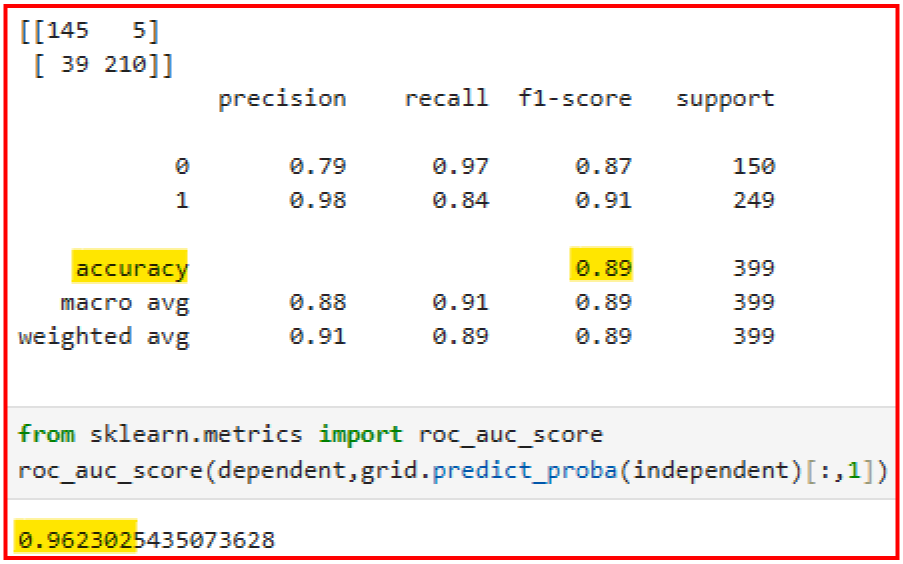
```
[[45  0]
 [ 6 69]]
              precision    recall  f1-score   support

       False       0.88      1.00      0.94        45
        True       1.00      0.92      0.96        75

    accuracy                           0.95       120
   macro avg       0.94      0.96      0.95       120
weighted avg       0.96      0.95      0.95       120

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

: 0.9995555555555555
```

8. Naivebayes:

A) ComplementNB



```
The f1_macro value for best parameter {'alpha': 0.1, 'fit_prior': True}: 0.8912968721618211
```

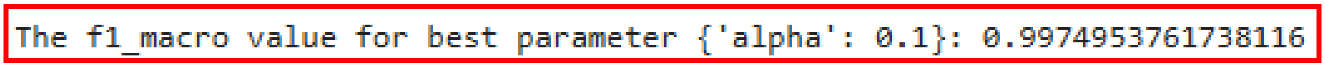


```
[[145   5]
 [ 39 210]]
              precision    recall  f1-score   support

           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
```

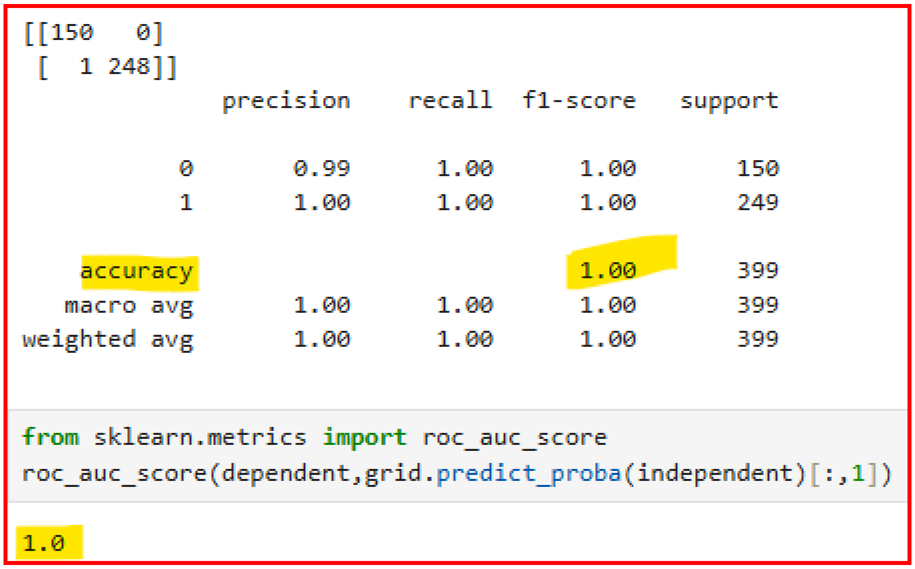
```
from sklearn.metrics import roc_auc_score
roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.9623025435073628
```

B) CategoricalNB



```
The f1_macro value for best parameter {'alpha': 0.1}: 0.9974953761738116
```

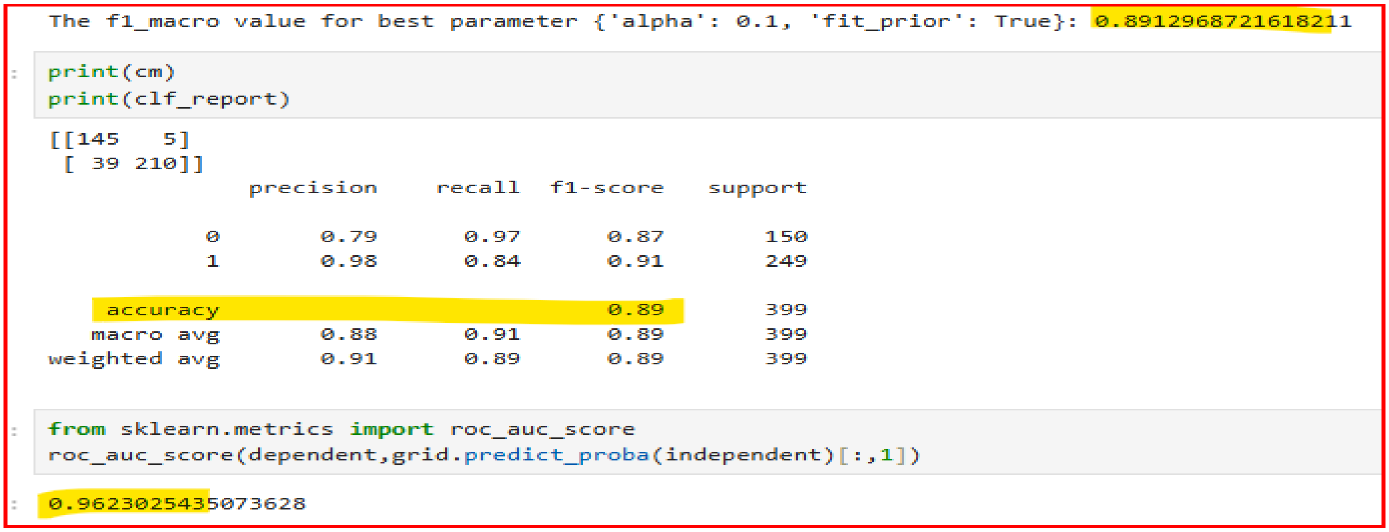


```
[[150   0]
 [  1 248]]
              precision    recall  f1-score   support

           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
```

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

```
1.0
```

C) C)MultinomialNB



```
The f1_macro value for best parameter {'alpha': 0.1, 'fit_prior': True}: 0.8912968721618211
```

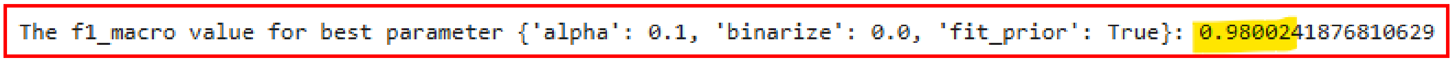
```
print(cm)
print(clf_report)
```

```
[[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
```

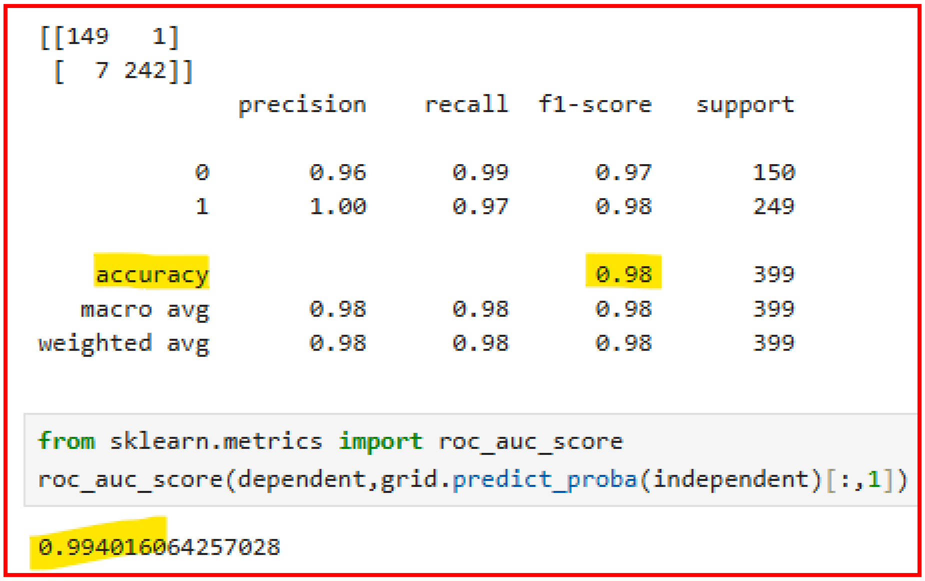
```
from sklearn.metrics import roc_auc_score
roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.9623025435073628
```

D) BernoulliNB



```
The f1_macro value for best parameter {'alpha': 0.1, 'binarize': 0.0, 'fit_prior': True}: 0.9800241876810629
```

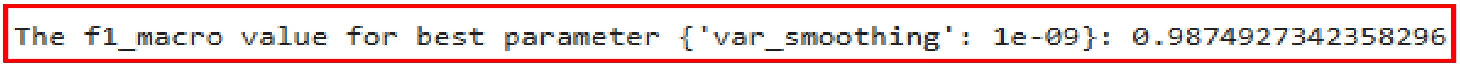


```
[[149   1]
 [  7 242]]
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       150
           1       1.00      0.97      0.98       249

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

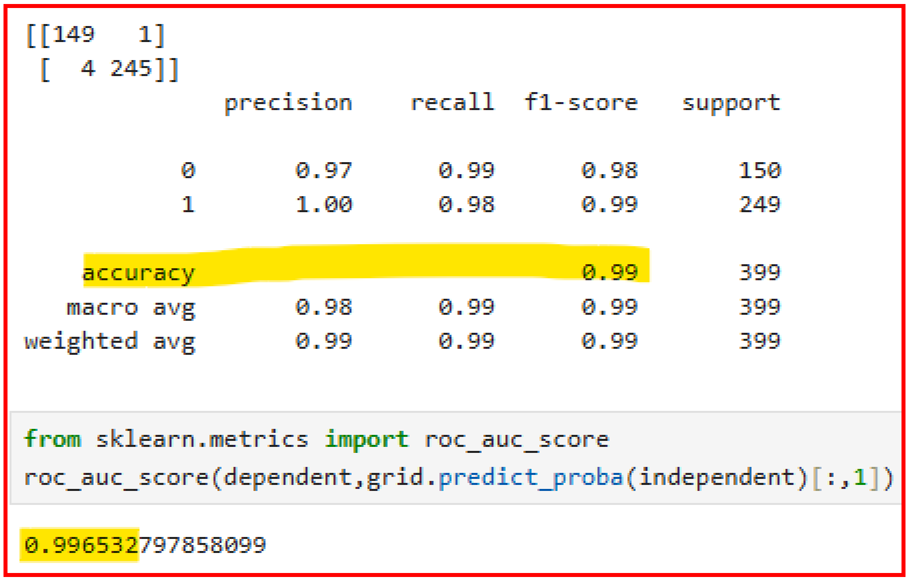
```python
from sklearn.metrics import roc_auc_score
roc_auc_score(dependent,grid.predict_proba(independent)[:,1])
```

```
0.994016064257028
```

E) GaussianNB



```
The f1_macro value for best parameter {'var_smoothing': 1e-09}: 0.9874927342358296
```



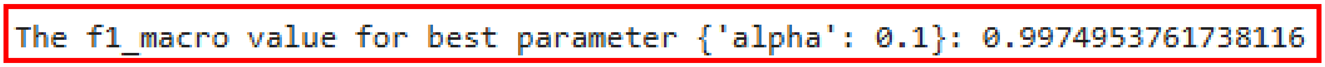
```
[[149   1]
 [  4 245]]
              precision    recall  f1-score   support

           0       0.97      0.99      0.98       150
           1       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
```

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

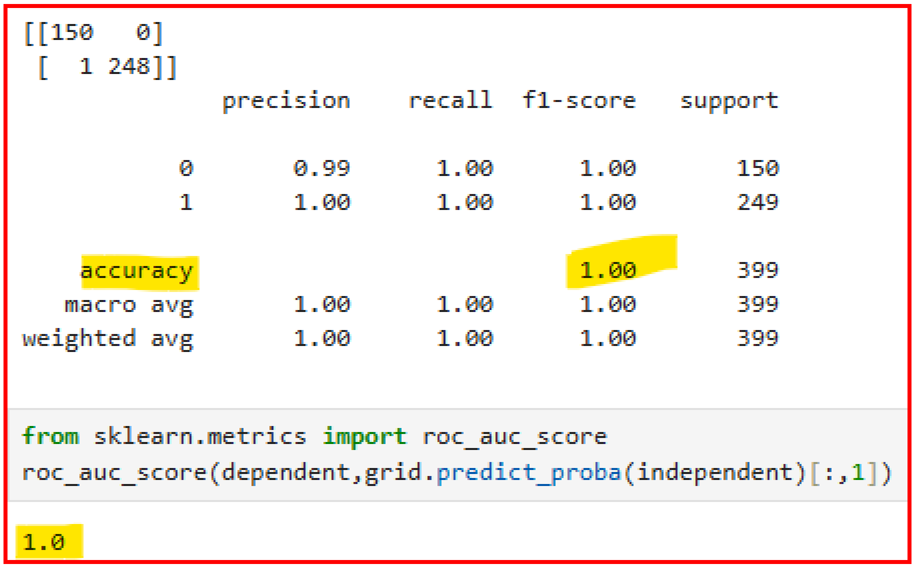
```
0.996532797858099
```

6.) Mention your final model, justify why u have chosen the same.

Best model for Machine Learning-Classification is CategoricalNB .



```
The f1_macro value for best parameter {'alpha': 0.1}: 0.9974953761738116
```



```
[[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
```

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

```
1.0
```