CLASSIFICATION ASSIGNMENT USING GRID WITH DATA PREPROCESSING

Problem statement

The client is from a health sector and they want help in classifying the people based on the results from their health check reports.

Dataset

The dataset contains 400 rows and 25 columns of data.

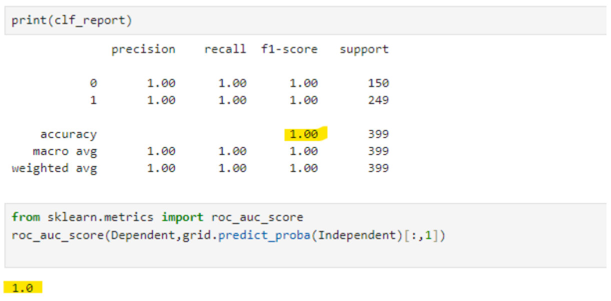
Data pre-processing

The data is a mix of categorical and numerical value. Here the categorical data is converted into numerical data using get dummies function from pandas.

Creating a good model

The goal is to create a model with good accuracy and roc_auc_curve using machine learning algorithm. Here the output is categorical so, “classifier” is used. It’s a supervised learning as the input and outputs is clearly defined.

1.SUPPORT VECTOR MACHINE

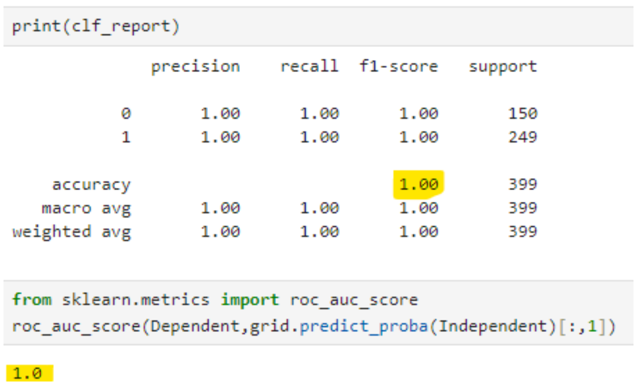


```
print(clf_report)
              precision    recall  f1-score   support

           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
```

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

```
1.0
```

2.DECISION TREE

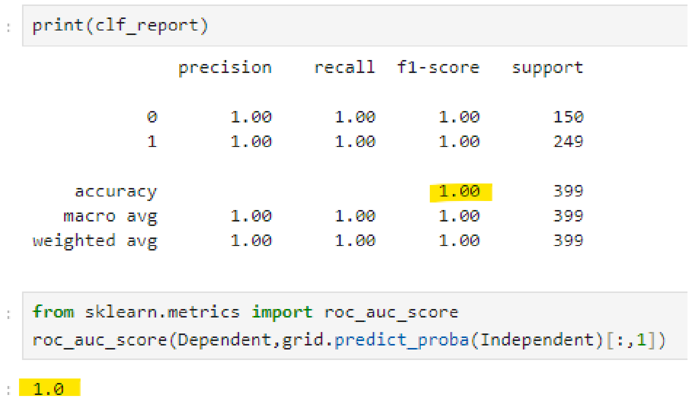


```
print(clf_report)
              precision    recall  f1-score   support

           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
```

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

```
1.0
```

3.Random Forest



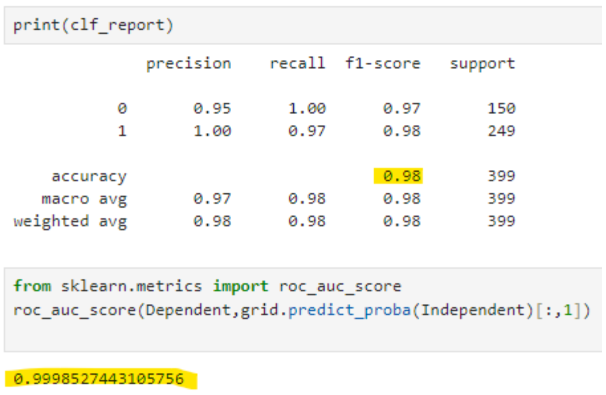
```
print(clf_report)
```

```
              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
```

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

```
1.0
```

4.KNN



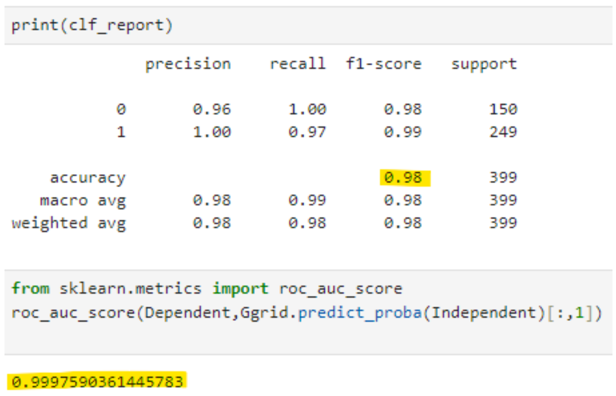
```
print(clf_report)
```

```
              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
```

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

```
0.9998527443105756
```

5.GausianNB



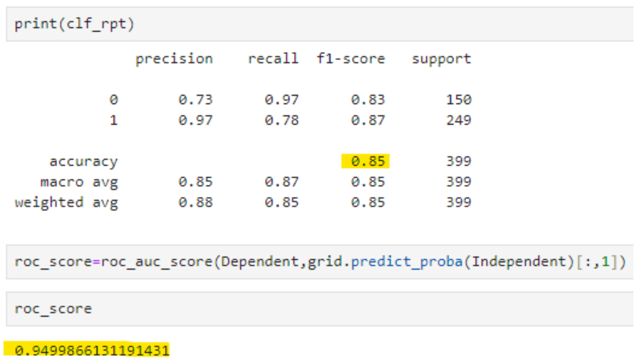
```
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       150
           1       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
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Dependent,Ggrid.predict_proba(Independent)[:,1])
```

```
0.9997590361445783
```

6.MultinomialNB



```
print(clf_rpt)
```

```
              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
```

```
roc_score=roc_auc_score(Dependent,grid.predict_proba(Independent)[:,1])
```

```
roc_score
```

```
0.9499866131191431
```

7.BernoulliNB

```
print(clf_report)
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97       150
           1       1.00      0.96      0.98       249

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

```
from sklearn.metrics import roc_auc_score
roc_auc_score(Dependent,Ggrid.predict_proba(Independent)[:,1])
```

```
0.9992235609103078
```

8.CategoricalNB

```
print(clf_rpt)
```

```
              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
```

```
roc_score=roc_auc_score(Dependent,grid.predict_proba(Independent)[:,1])
```

```
roc_score
```

```
0.9999196787148594
```

9.ComplementNB

```
print(clf_rpt)
```

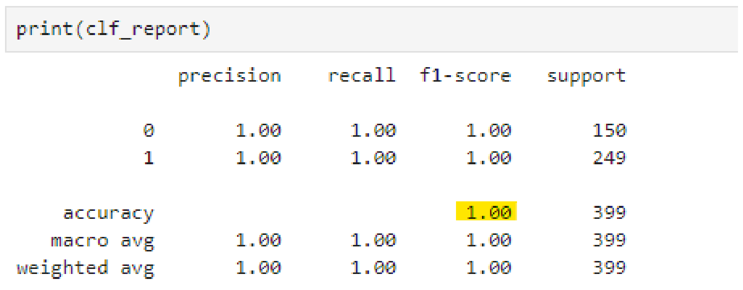
```
              precision    recall  f1-score   support

           0       0.72      0.97      0.83       150
           1       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
```

```
roc_score=roc_auc_score(Dependent,grid.predict_proba(Independent)[:,1])
```

```
roc_score
```

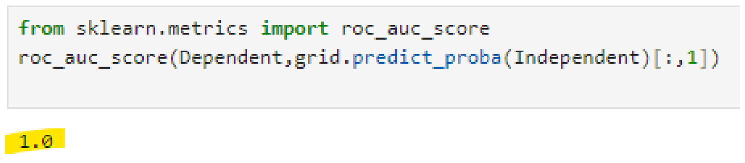
```
0.9499866131191431
```

10.Logistic Regression



```
print(clf_report)
```

```
              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
```



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

```
1.0
```

Final Model

The accuracy and roc_score are 0.1 for the following algorithms.

- Logistic Regression
- Random Forest
- Support Vector Machine
- Decision Tree

Among this best model selecting the “Decision Tree” algorithm for Deployment

The final model is **Machine learning>>Classification>>Decision Tree**

Justification: While using the Decision Tree algorithm, we get the accuracy and roc_score as **1.0**

The best parameters for Decision Tree are {'criterion': 'entropy', 'max_features': 'log2', 'splitter': 'random'}