MACHINE LEARNING CLASSIFICATION ASSIGNMENT

1. Problem Statement:

a. Machine Learning
b. Supervised Learing
c. Classification

2. Basic info of Dataset:

No of Rows = 399
No of Columns = 25

3. Pre-Processing Method:

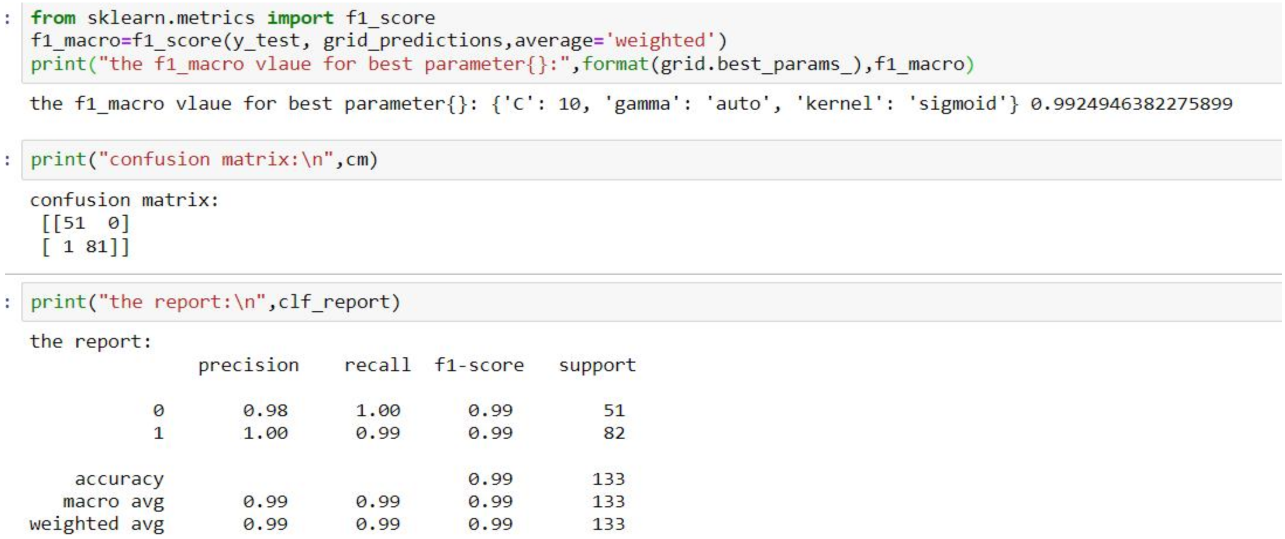
Odinal - Mapping Label Encoder
Standard scaler

4. Final model:

Random Forest or SVM or Logistic Regression

5. Models with Confusion matrix:

A).Support Vector Machine:



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions,average='weighted')
print("the f1_macro vlaue for best parameter{}:",format(grid.best_params_),f1_macro)
```

```
the f1_macro vlaue for best parameter{}: {'C': 10, 'gamma': 'auto', 'kernel': 'sigmoid'} 0.9924946382275899
```

```
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[51  0]
 [ 1 81]]
```

```
print("the report:\n",clf_report)
```

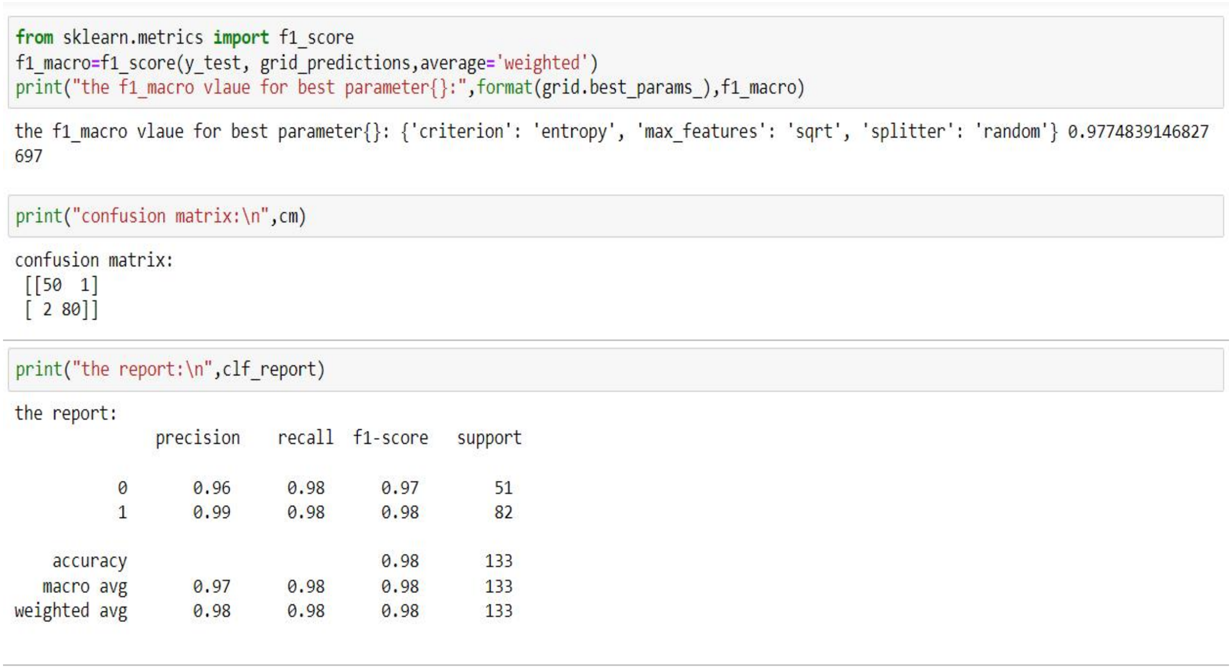
```
the report:
               precision    recall  f1-score   support

           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
```

MACHINE LEARNING CLASSIFICATION ASSIGNMENT

B).Decision Tree:



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions,average='weighted')
print("the f1_macro vlaue for best parameter{}:",format(grid.best_params_),f1_macro)
```

```
the f1_macro vlaue for best parameter{}: {'criterion': 'entropy', 'max_features': 'sqrt', 'splitter': 'random'} 0.9774839146827
697
```

```
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[50  1]
 [ 2 80]]
```

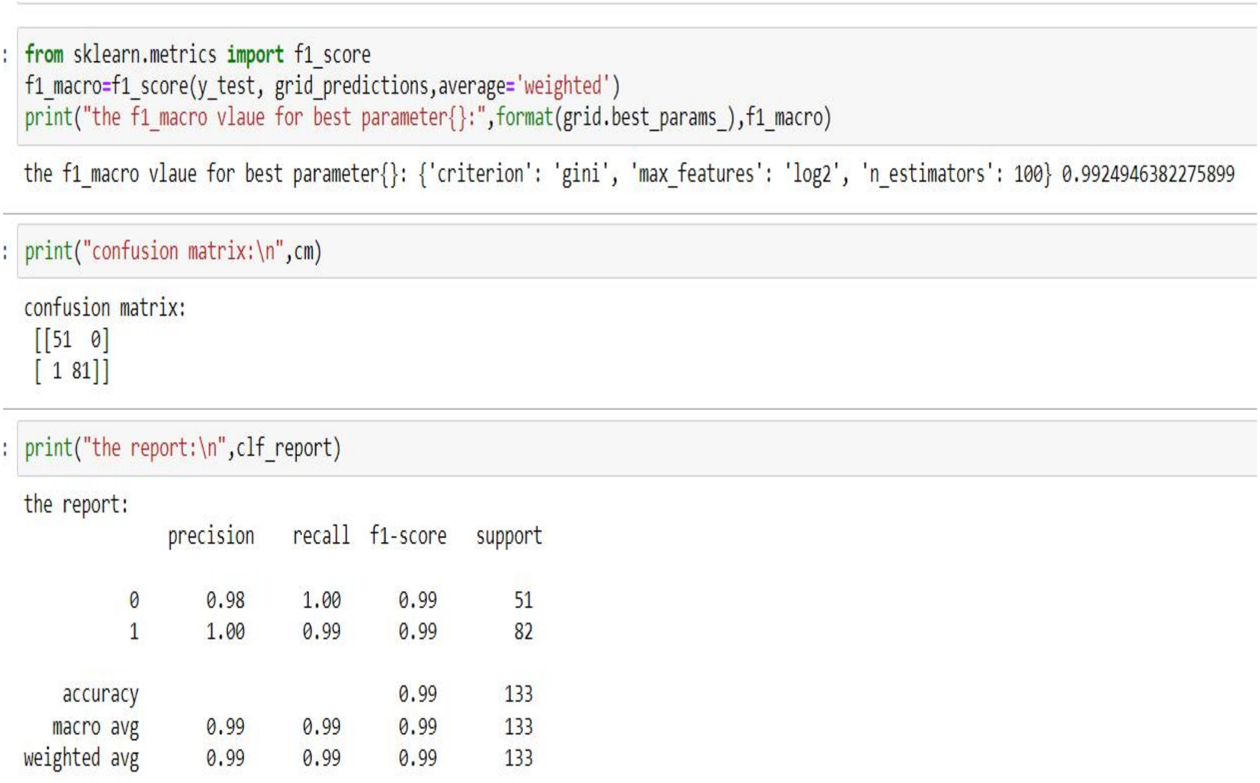
```
print("the report:\n",clf_report)
```

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

           0       0.96      0.98      0.97        51
           1       0.99      0.98      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
```

C).Random Forest



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions,average='weighted')
print("the f1_macro vlaue for best parameter{}:",format(grid.best_params_),f1_macro)
```

```
the f1_macro vlaue for best parameter{}: {'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 100} 0.9924946382275899
```

```
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[51  0]
 [ 1 81]]
```

```
print("the report:\n",clf_report)
```

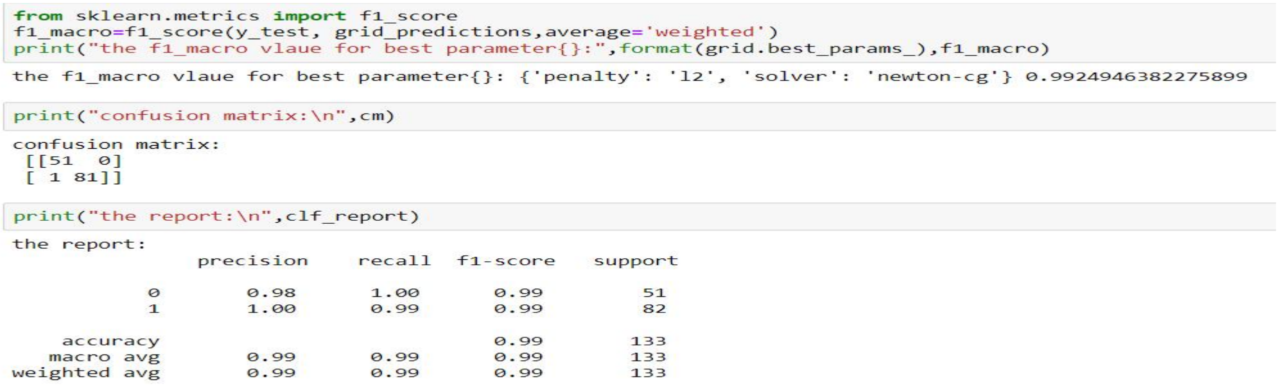
```
the report:
               precision    recall  f1-score   support

           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
```

MACHINE LEARNING CLASSIFICATION ASSIGNMENT

D) .Logistic Regression:



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions,average='weighted')
print("the f1_macro vlaue for best parameter{}:",format(grid.best_params_),f1_macro)
```

```
the f1_macro vlaue for best parameter{}: {'penalty': 'l2', 'solver': 'newton-cg'} 0.9924946382275899
```

```
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[51  0]
 [ 1 81]]
```

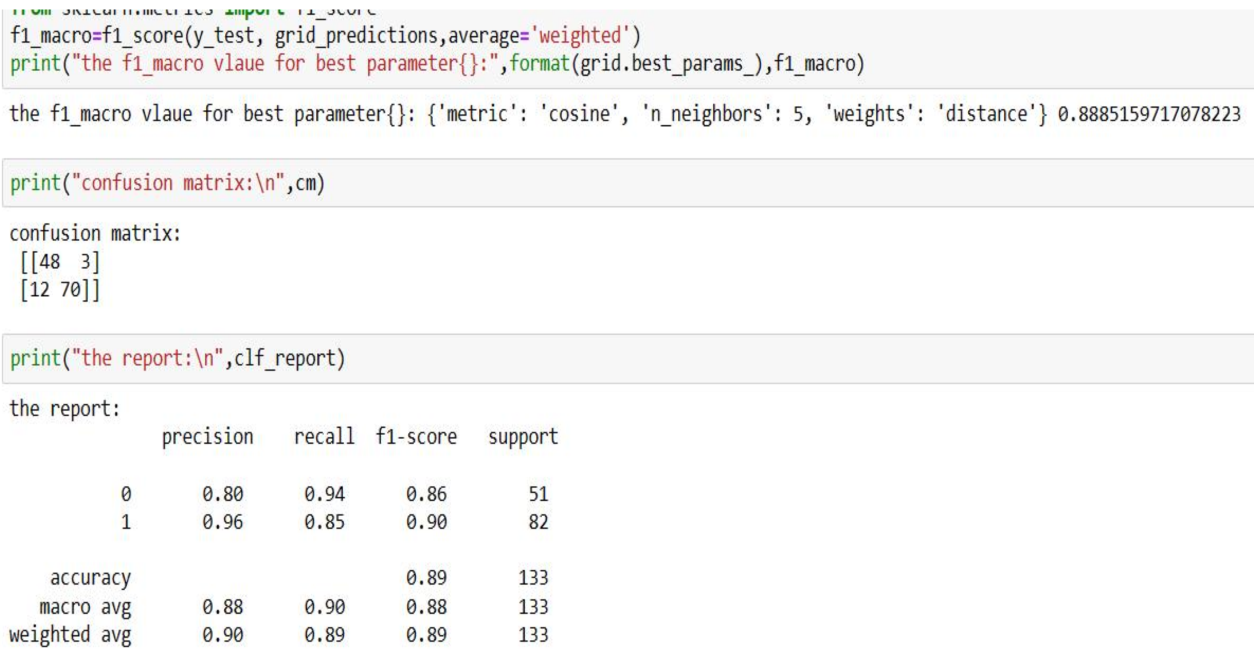
```
print("the report:\n",clf_report)
```

```
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
```

E) .KNN:



```
from sklearn.metrics import f1_score
f1_macro=f1_score(y_test, grid_predictions,average='weighted')
print("the f1_macro vlaue for best parameter{}:",format(grid.best_params_),f1_macro)
```

```
the f1_macro vlaue for best parameter{}: {'metric': 'cosine', 'n_neighbors': 5, 'weights': 'distance'} 0.8885159717078223
```

```
print("confusion matrix:\n",cm)
```

```
confusion matrix:
 [[48  3]
 [12 70]]
```

```
print("the report:\n",clf_report)
```

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

           0       0.80      0.94      0.86        51
           1       0.96      0.85      0.90        82

    accuracy                           0.89       133
   macro avg       0.88      0.90      0.88       133
weighted avg       0.90      0.89      0.89       133
```

6. Justification of Final model:

In this Requirement i created many models with evaluation metrics. All models giving superb results except KNN. but SVM,Random forest and Logistic Regression giving little bit nearer to 1 compared to Decision tree and KNN. so i choose Random forest,SVM and Logistic Regression are good model for this requirement.