```
import pandas as pd
from imblearn.over_sampling import SMOTE
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix,f1_score, roc_auc_score,
accuracy_score
from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score

from sklearn.neural_network import MLPClassifier
from sklearn.metrics import precision_recall_fscore_support
from collections import Counter

train =pd.read_csv('train_unb.csv')
test = pd.read_csv('test.csv')
train_b = pd.read_csv('train_b.csv')

# split x y
train_x = train.drop(['Revenue'], axis=1)
train_y=train['Revenue']


test_x = test.drop(['Revenue'], axis=1)
test_y=test['Revenue']


train_b_x = train_b.drop(['Revenue'], axis=1)
train_b_y=train_b['Revenue']

# tuning
parameters1 = {'kernel':['rbf'], 'gamma': [1e-2, 1e-3,1e-4], 'C':[1,
5,10,50]}
parameters2 = {'kernel':['linear'], 'C':[0.25, 0.5,1,2]}

svc = SVC()
clf_rbf = GridSearchCV(svc, parameters1)
clf_rbf.fit(train_x,train_y)

GridSearchCV(estimator=SVC(),
             param_grid={'C': [1, 5, 10, 50], 'gamma': [0.01, 0.001,
0.0001],
                         'kernel': ['rbf']})

clf_rbf.cv_results_['params'][clf_rbf.best_index_]

{'C': 1, 'gamma': 0.01, 'kernel': 'rbf'}

# SVM model: kernel: rbf, gamma0.01: , C:1
SVM_selected = SVC(kernel = "rbf", gamma=0.01,C=1)
SVM_selected.fit(train_x,train_y)
pred_y_rbf = SVM_selected.predict(test_x)
```

```python
auc = accuracy_score(test_y, pred_y_rbf)
ROC_auc = roc_auc_score(test_y, pred_y_rbf)
print('accuracy score: ', auc)
print('roc_auc score: ', ROC_auc)
```

```
accuracy score:  0.8448229251148959
roc_auc score:  0.5
```

```python
macro_F1 = f1_score(test_y, pred_y_rbf, average='macro')
micro_F1 = f1_score(test_y, pred_y_rbf, average='micro')
print('macroaveraged F1: ', macro_F1)
print('microaveraged F1: ', micro_F1)
```

```
macroaveraged F1:  0.4579425556858147
microaveraged F1:  0.8448229251148958
```

```python
# SVM model kernel linear, C=2
SVM_lin = SVC(kernel = "linear", C=2)
SVM_lin.fit(train_x,train_y)
pred_y_lin = SVM_lin.predict(test_x)
```

```python
auc = accuracy_score(test_y, pred_y_lin)
ROC_auc = roc_auc_score(test_y, pred_y_lin)
print('accuracy score: ', auc)
print('roc_auc score: ', ROC_auc)
```

```
accuracy score:  0.8732089753987564
roc_auc score:  0.7784774912891985
```

```python
macro_F1 = f1_score(test_y, pred_y_lin, average='macro')
micro_F1 = f1_score(test_y, pred_y_lin, average='micro')
print('macroaveraged F1: ', macro_F1)
print('microaveraged F1: ', micro_F1)
```

```
macroaveraged F1:  0.7675288590451138
microaveraged F1:  0.8732089753987564
```

```python
# polynomial
SVM_poly = SVC(kernel = "poly", C=1,gamma=0.01,degree=2)
SVM_poly.fit(train_x,train_y)
pred_y_poly = SVM_poly.predict(test_x)
```

```python
auc = accuracy_score(test_y, pred_y_poly)
ROC_auc = roc_auc_score(test_y, pred_y_poly)
print('accuracy score: ', auc)
print('roc_auc score: ', ROC_auc)
```

```
accuracy score:  0.8848337388483374
roc_auc score:  0.6936281533101045
```

```python
macro_F1 = f1_score(test_y, pred_y_poly, average='macro')
micro_F1 = f1_score(test_y, pred_y_poly, average='micro')
```

```python
print('macroaveraged F1: ', macro_F1)
print('microaveraged F1: ', micro_F1)
```

```
macroaveraged F1:  0.7315810237360014
microaveraged F1:  0.8848337388483374
```

```python
# Balanced data
# SVM model kernel linear, C=2
SVM_linb = SVC(kernel = "linear", C=2)
SVM_linb.fit(train_b_x,train_b_y)
pred_y_linb = SVM_linb.predict(test_x)
```

```python
auc = accuracy_score(test_y, pred_y_linb)
ROC_auc = roc_auc_score(test_y, pred_y_linb)
print('accuracy score: ', auc)
print('roc_auc score: ', ROC_auc)
```

```
accuracy score:  0.8799675587996756
roc_auc score:  0.7945658536585367
```

```python
macro_F1 = f1_score(test_y, pred_y_linb, average='macro')
micro_F1 = f1_score(test_y, pred_y_linb, average='micro')
print('macroaveraged F1: ', macro_F1)
print('microaveraged F1: ', micro_F1)
```

```
macroaveraged F1:  0.7812343487258397
microaveraged F1:  0.8799675587996756
```

```python
# Multi layers perceptron
# unbalanced single layer
un_single_mlp =
MLPClassifier(hidden_layer_sizes=(50),max_iter=300,random_state =
42,learning_rate_init=0.001).fit(train_x,train_y.values.ravel())
```

```python
y_un_smlppred = un_single_mlp.predict(test_x)
macro_F11 = f1_score(test_y, y_un_smlppred, average='macro')
print('macroaveraged F1 of Single MLP: ', macro_F11)
```

```
macroaveraged F1 of Single MLP:  0.7552723074957566
```

```python
auc1 = accuracy_score(test_y, y_un_smlppred)
ROC_auc1 = roc_auc_score(test_y, y_un_smlppred)
print('accuracy score: ', auc1)
print('roc_auc score: ', ROC_auc1)
```

```
accuracy score:  0.8902406055690727
roc_auc score:  0.7210048780487806
```

```python
#unbalanced Multi layers
un_multi_mlp =
MLPClassifier(hidden_layer_sizes=(50,50,50,50),max_iter=300,random_sta
te = 42,learning_rate_init=0.001).fit(train_x,train_y.values.ravel())
```

```
y_un_mmlppred = un_multi_mlp.predict(test_x)
macro_F12 = f1_score(test_y, y_un_mmlppred, average='macro')
print('macroaveraged F1 of Multi MLP: ', macro_F12)
```

```
macroaveraged F1 of Multi MLP:  0.7387607393597695
```

```
auc2 = accuracy_score(test_y, y_un_mmlppred)
ROC_auc2 = roc_auc_score(test_y, y_un_mmlppred)
print('accuracy score: ', auc2)
print('roc_auc score: ', ROC_auc2)
```

```
accuracy score:  0.8878075155447418
roc_auc score:  0.6996546341463414
```

```
#balanced single layer
single_mlp =
MLPClassifier(hidden_layer_sizes=(50),max_iter=300,random_state =
42,learning_rate_init=0.001).fit(train_b_x,train_b_y.values.ravel())
```

```
y_smlppred = single_mlp.predict(test_x)
macro_F13 = f1_score(test_y, y_smlppred, average='macro')
print('macroaveraged F1 of Single MLP: ', macro_F13)
```

```
macroaveraged F1 of Single MLP:  0.7069512716839467
```

```
auc3 = accuracy_score(test_y, y_smlppred)
ROC_auc3 = roc_auc_score(test_y, y_smlppred)
print('accuracy score: ', auc3)
print('roc_auc score: ', ROC_auc3)
```

```
accuracy score:  0.8796972154636388
roc_auc score:  0.6678335888501743
```

```
#balanced multi layer
multi_mlp =
MLPClassifier(hidden_layer_sizes=(50,50,50,50),max_iter=300,random_sta
te =
42,learning_rate_init=0.001).fit(train_b_x,train_b_y.values.ravel())


y_mmlppred = multi_mlp.predict(test_x)
macro_F14 = f1_score(test_y, y_mmlppred, average='macro')
print('macroaveraged F1 of Single MLP: ', macro_F14)
```

```
macroaveraged F1 of Single MLP:  0.735611703621261
```

```
auc4 = accuracy_score(test_y, y_mmlppred)
ROC_auc4 = roc_auc_score(test_y, y_mmlppred)
print('accuracy score: ', auc4)
print('roc_auc score: ', ROC_auc4)
```

```
accuracy score:  0.8788861854555285
roc_auc score:  0.7078851567944251
```