Lab Assignment 6
Chaudhary Hamdan
1905387
Date: 09-03-2022

1. On ‘Income dataset’

```
# coding: utf-8

# In[1]:


import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import seaborn as sns

from sklearn import svm
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score


# In[2]:


import warnings
warnings.filterwarnings('ignore')
```

```
# In[3]:


df = pd.read_excel('income.xlsx')


# In[4]:


df.head()


# In[5]:


df.drop(columns=['capitalgain', 'capitalloss'], inplace=True)


# In[6]:


df.head()


# In[7]:


cols = ['JobType', 'EdType', 'maritalstatus', 'occupation', 'relationship',
'race',
        'gender', 'nativecountry', 'SalStat']

for col in cols:
    le = preprocessing.LabelEncoder()
    df[col] = le.fit_transform(df[col])


# In[8]:


df.head()
```

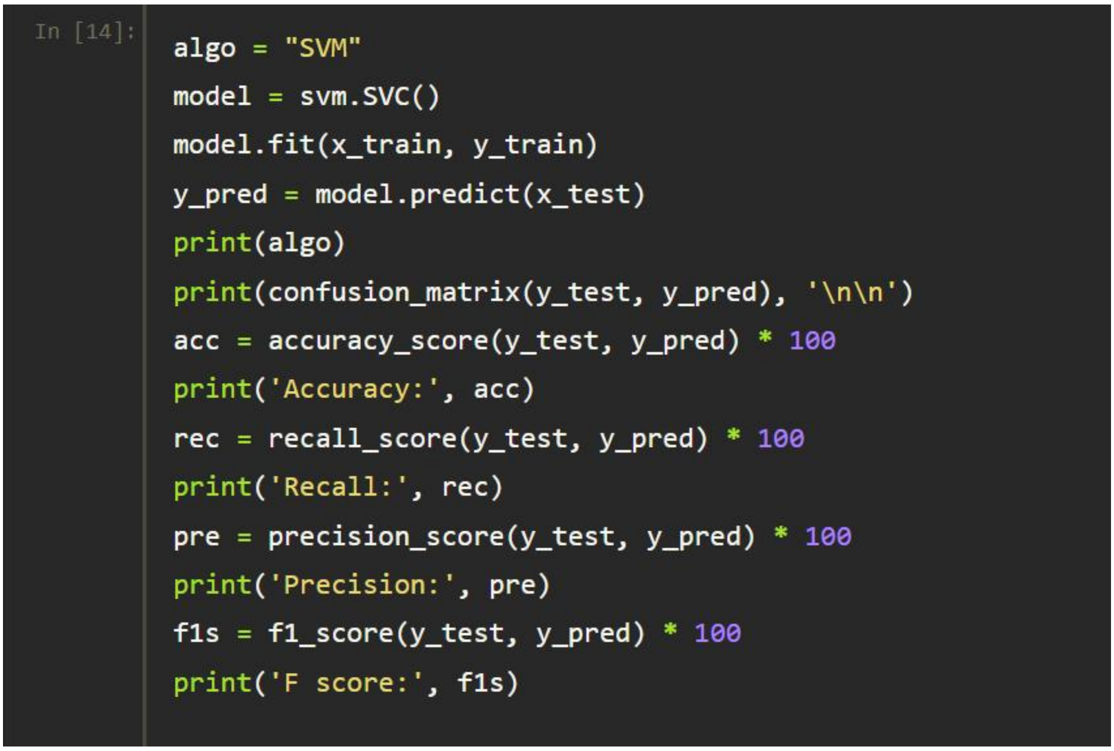
```
# In[9]:


df.SalStat.value_counts()


# 1 : Less than or equal to 50k, 0 means less than 50k

# In[10]:


df.head()


# In[11]:


corr = df.corr()
corr.style.background_gradient(cmap='coolwarm')


# In[12]:


x_train, x_test, y_train, y_test = train_test_split(df.drop(columns =
['SalStat']), df['SalStat'], test_size = 0.2)
x_train.shape, y_train.shape, x_test.shape, y_test.shape


# In[13]:


algos = []
accuracy = []
recall = []
precision = []
f1Score = []


# In[14]:


algo = "SVM"
```

```
model = svm.SVC()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)


# In[15]:


algo = "Gradient Boost"
model = GradientBoostingClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
```

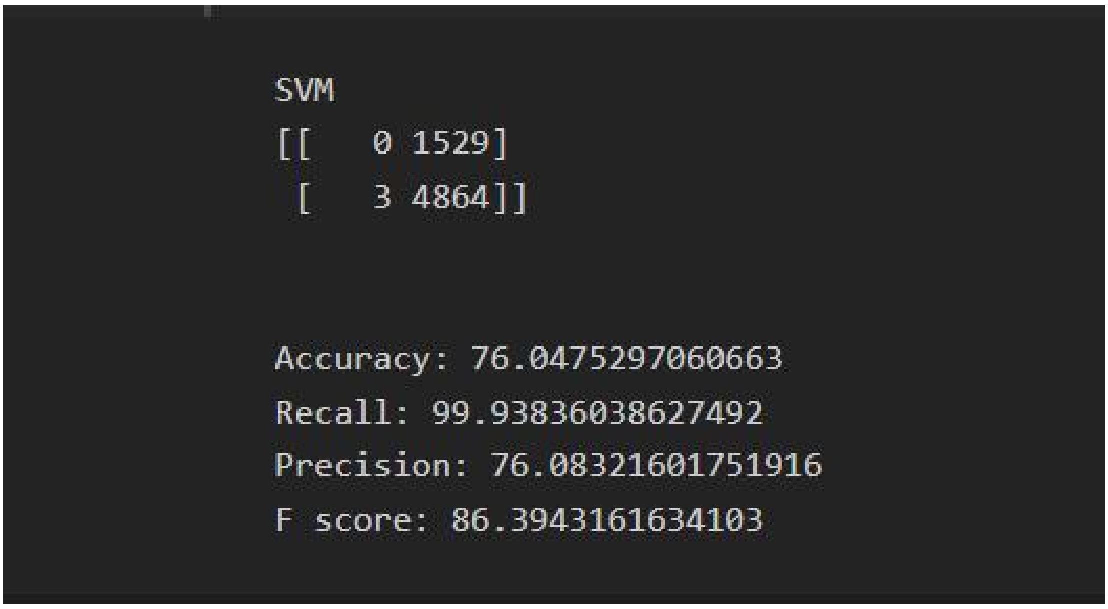
```
f1Score.append(f1s)


# In[16]:


algo = "Ada Boost"
model = AdaBoostClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)

algos.append(algo)
accuracy.append(acc)
recall.append(rec)
precision.append(pre)
f1Score.append(f1s)


# In[17]:


for i in range(3):
    print(algos[i], ':     ', accuracy[i],',  ', recall[i],',  ', precision[i],',  ',
f1Score[i])


# In[18]:


plt.bar(algos, accuracy)
plt.show()
```

A)apply Support vector machine algorithm and find out the accuarcy in predicting whether the salary status is less than or equal to 5000 or it is greater tan 50000.

i) Confusion Matrix
ii) Accuracy score
iii)recall, precision, f- score



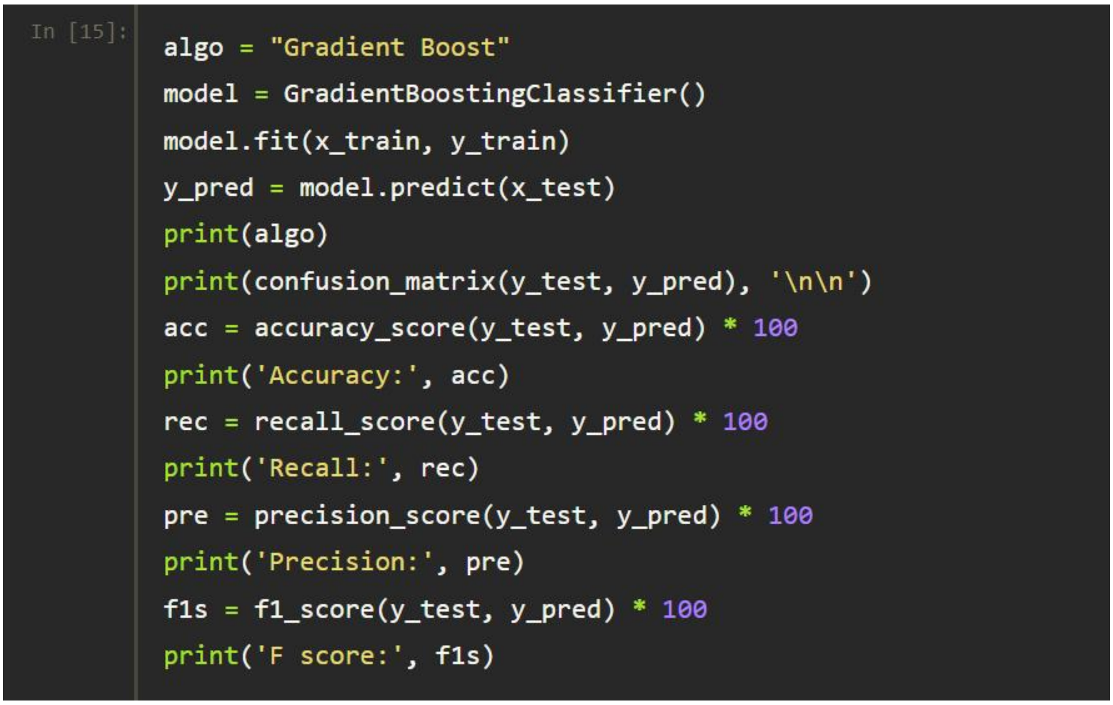
```
algo = "SVM"
model = svm.SVC()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
```



```
SVM
[[   0 1529]
 [   3 4864]]


Accuracy: 76.0475297060663
Recall: 99.93836038627492
Precision: 76.08321601751916
F score: 86.3943161634103
```

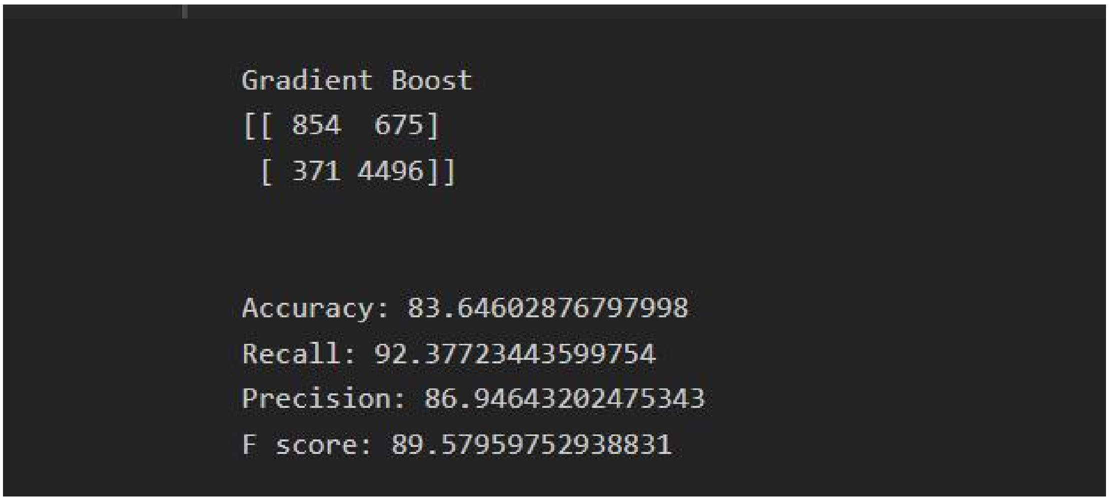
B)apply Gradient Boost algorithm and find out the accuarcy in predicting whether the salary status is less than or equal to 5000 or it is greater tan 50000.

I)Confusion Matrix
ii)Accuracy score
iii)recall, precision, f- score



In [15]:

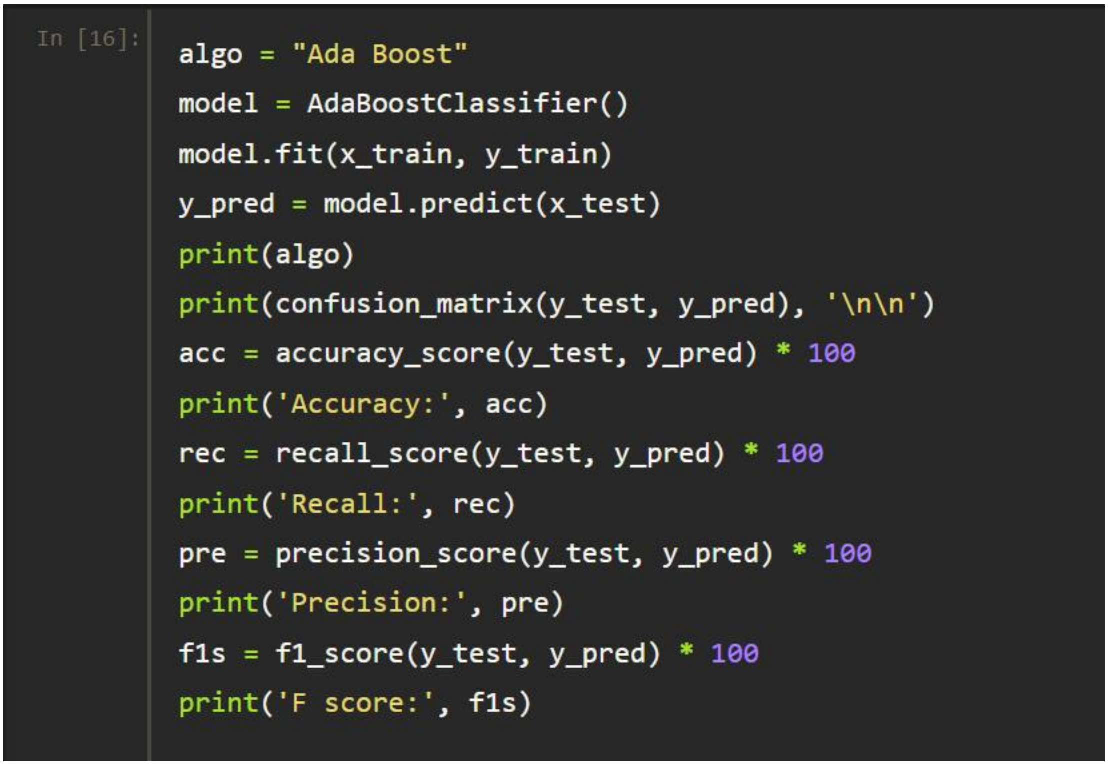
```
algo = "Gradient Boost"
model = GradientBoostingClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
```



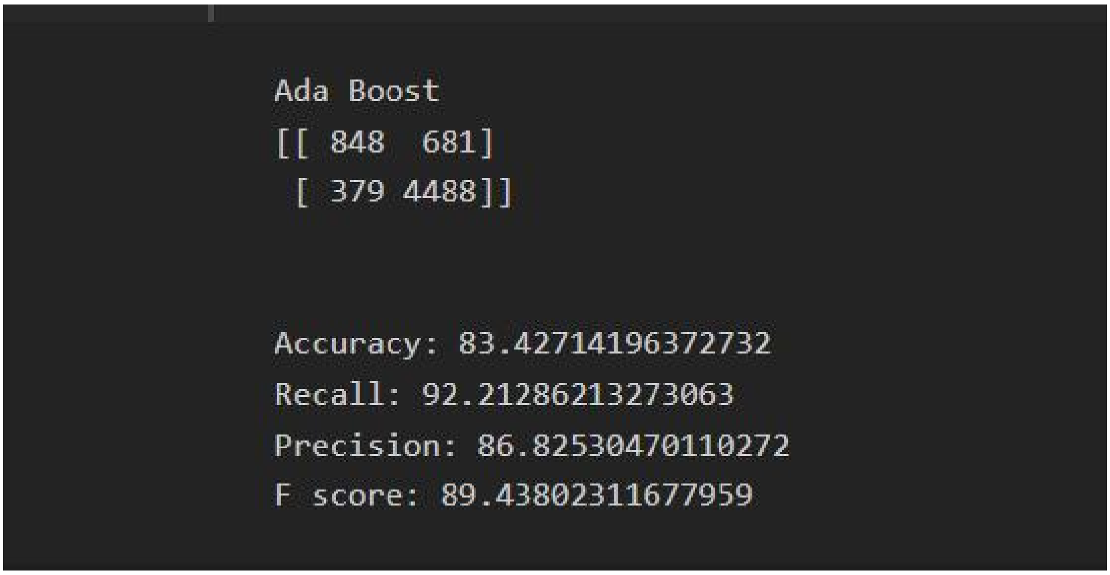
```
Gradient Boost
[[ 854  675]
 [ 371 4496]]


Accuracy: 83.64602876797998
Recall: 92.37723443599754
Precision: 86.94643202475343
F score: 89.57959752938831
```

C)apply Adaboost algorithm and find out the accuarcy in predicting whether the salary status is less than or equal to 5000 or it is greater tan 50000.

I)Confusion Matrix
ii)Accuracy score
iii)recall, precision, f- score



```
In [16]:
algo = "Ada Boost"
model = AdaBoostClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
print(algo)
print(confusion_matrix(y_test, y_pred), '\n\n')
acc = accuracy_score(y_test, y_pred) * 100
print('Accuracy:', acc)
rec = recall_score(y_test, y_pred) * 100
print('Recall:', rec)
pre = precision_score(y_test, y_pred) * 100
print('Precision:', pre)
f1s = f1_score(y_test, y_pred) * 100
print('F score:', f1s)
```



```
Ada Boost
[[ 848  681]
 [ 379 4488]]


Accuracy: 83.42714196372732
Recall: 92.21286213273063
Precision: 86.82530470110272
F score: 89.43802311677959
```