gaussian-naive-bayes

December 12, 2023

[90]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import RobustScaler
import category_encoders as ce

from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

[91]:
```
df = pd.read_csv("/kaggle/input/adult-census-dataset/adult.csv")
df.head()
```

[91]:
```
   age         workclass  fnlwgt  education  education-num  \
0   39         State-gov   77516  Bachelors             13
1   50  Self-emp-not-inc   83311  Bachelors             13
2   38           Private  215646    HS-grad              9
3   53           Private  234721       11th              7
4   28           Private  338409  Bachelors             13

       marital-status          occupation   relationship   race     sex  \
0       Never-married        Adm-clerical  Not-in-family  White    Male
1  Married-civ-spouse     Exec-managerial        Husband  White    Male
2            Divorced   Handlers-cleaners  Not-in-family  White    Male
3  Married-civ-spouse   Handlers-cleaners        Husband  Black    Male
4  Married-civ-spouse      Prof-specialty           Wife  Black  Female

   capital-gain  capital-loss  hours-per-week        country  salary
0          2174             0              40  United-States   <=50K
1             0             0              13  United-States   <=50K
2             0             0              40  United-States   <=50K
3             0             0              40  United-States   <=50K
4             0             0              40           Cuba   <=50K
```

[92]:
```
df.columns
```

[92]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'country', 'salary'],
      dtype='object')
```

[93]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education-num   32561 non-null  int64
 5   marital-status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64
 11  capital-loss    32561 non-null  int64
 12  hours-per-week  32561 non-null  int64
 13  country         32561 non-null  object
 14  salary          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

[94]:
```
df.describe()
```

[94]:
```
                age        fnlwgt  education-num  capital-gain  capital-loss  \
count  32561.000000  3.256100e+04   32561.000000  32561.000000  32561.000000
mean      38.581647  1.897784e+05      10.080679   1077.648844     87.303830
std       13.640433  1.055500e+05       2.572720   7385.292085    402.960219
min       17.000000  1.228500e+04       1.000000      0.000000      0.000000
25%       28.000000  1.178270e+05       9.000000      0.000000      0.000000
50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000
75%       48.000000  2.370510e+05      12.000000      0.000000      0.000000
max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000

       hours-per-week
count    32561.000000
mean        40.437456
```

```
std          12.347429
min           1.000000
25%          40.000000
50%          40.000000
75%          45.000000
max          99.000000
```

[95]:
```python
#find categorical variables
categorical = [var for var in df.columns if df[var].dtype=='O']
print('There are {} categorical variables\n'.format(len(categorical)))
print('The categorical variables are :\n\n', categorical)
```

```
There are 9 categorical variables

The categorical variables are :

 ['workclass', 'education', 'marital-status', 'occupation', 'relationship',
'race', 'sex', 'country', 'salary']
```

[96]:
```python
df[categorical].head()
```

[96]:
```
          workclass  education       marital-status         occupation  \
0         State-gov  Bachelors        Never-married       Adm-clerical
1  Self-emp-not-inc  Bachelors   Married-civ-spouse    Exec-managerial
2           Private    HS-grad             Divorced  Handlers-cleaners
3           Private       11th   Married-civ-spouse  Handlers-cleaners
4           Private  Bachelors   Married-civ-spouse     Prof-specialty

    relationship   race     sex        country  salary
0  Not-in-family  White    Male  United-States   <=50K
1        Husband  White    Male  United-States   <=50K
2  Not-in-family  White    Male  United-States   <=50K
3        Husband  Black    Male  United-States   <=50K
4           Wife  Black  Female           Cuba   <=50K
```

[97]:
```python
df[categorical].columns
```

[97]:
```
Index(['workclass', 'education', 'marital-status', 'occupation',
       'relationship', 'race', 'sex', 'country', 'salary'],
      dtype='object')
```

[98]:
```python
df[categorical].isnull().sum()
```

[98]:
```
workclass         0
education         0
marital-status    0
occupation        0
```

```
relationship      0
race              0
sex               0
country           0
salary            0
dtype: int64
```

[99]:
```
df.workclass.unique()
```

[99]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' ?', ' Self-emp-inc', ' Without-pay',
       ' Never-worked'], dtype=object)
```

[100]:
```
df.workclass.value_counts()
```

[100]:
```
workclass
 Private             22696
 Self-emp-not-inc     2541
 Local-gov            2093
 ?                    1836
 State-gov            1298
 Self-emp-inc         1116
 Federal-gov           960
 Without-pay            14
 Never-worked            7
Name: count, dtype: int64
```

[101]:
```
df.country.unique()
```

[101]:
```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' ?', ' Mexico',
       ' South', ' Puerto-Rico', ' Honduras', ' England', ' Canada',
       ' Germany', ' Iran', ' Philippines', ' Italy', ' Poland',
       ' Columbia', ' Cambodia', ' Thailand', ' Ecuador', ' Laos',
       ' Taiwan', ' Haiti', ' Portugal', ' Dominican-Republic',
       ' El-Salvador', ' France', ' Guatemala', ' China', ' Japan',
       ' Yugoslavia', ' Peru', ' Outlying-US(Guam-USVI-etc)', ' Scotland',
       ' Trinadad&Tobago', ' Greece', ' Nicaragua', ' Vietnam', ' Hong',
       ' Ireland', ' Hungary', ' Holand-Netherlands'], dtype=object)
```

[102]:
```
# find numerical variables
numerical = [var for var in df.columns if df[var].dtype!='O']
print('There are {} numerical variables\n'.format(len(numerical)))
print('The numerical variables are :', numerical)
```

```
There are 6 numerical variables

The numerical variables are : ['age', 'fnlwgt', 'education-num', 'capital-gain',
```

```
'capital-loss', 'hours-per-week']
```

[103]:
```python
df[numerical].head()
```

[103]:

	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
0	39	77516	13	2174	0	40
1	50	83311	13	0	0	13
2	38	215646	9	0	0	40
3	53	234721	7	0	0	40
4	28	338409	13	0	0	40

[104]:
```python
df[numerical].isnull().sum()
```

[104]:
```
age               0
fnlwgt            0
education-num     0
capital-gain      0
capital-loss      0
hours-per-week    0
dtype: int64
```

[105]:
```python
X = df.drop(['salary'], axis=1)
y = df['salary']
```

[106]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
↪random_state = 0)
```

[107]:
```python
X_train.shape, X_test.shape
```

[107]:
```
((22792, 14), (9769, 14))
```

[108]:
```python
X_train.dtypes
```

[108]:
```
age                int64
workclass         object
fnlwgt             int64
education         object
education-num      int64
marital-status    object
occupation        object
relationship      object
race              object
sex               object
capital-gain       int64
capital-loss       int64
hours-per-week     int64
```

```
country           object
dtype: object
```

[109]:
```
X_train.isnull().sum()
```

[109]:
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
country           0
dtype: int64
```

[110]:
```
X_test.isnull().sum()
```

[110]:
```
age               0
workclass         0
fnlwgt            0
education         0
education-num     0
marital-status    0
occupation        0
relationship      0
race              0
sex               0
capital-gain      0
capital-loss      0
hours-per-week    0
country           0
dtype: int64
```

[111]:
```
categorical
```

[111]:
```
['workclass',
 'education',
 'marital-status',
 'occupation',
 'relationship',
 'race',
```

```
'sex',
 'country',
 'salary']
```

[112]:

```
print(X_train.columns)
```

```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capital-gain', 'capital-loss', 'hours-per-week', 'country'],
      dtype='object')
```

[113]:

```
import category_encoders as ce

encoder = ce.OneHotEncoder(['workclass', 'education', 'marital_status',␣
 ↪'occupation', 'relationship',
                                 'race', 'sex', 'country'])

X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)
X_train.head()
```

[113]:

```
       age  workclass_1  workclass_2  workclass_3  workclass_4  workclass_5  \
32098   45            1            0            0            0            0
25206   47            0            1            0            0            0
23491   48            1            0            0            0            0
12367   29            1            0            0            0            0
7054    23            1            0            0            0            0

       workclass_6  workclass_7  workclass_8  workclass_9  …  country_33  \
32098            0            0            0            0  …           0
25206            0            0            0            0  …           0
23491            0            0            0            0  …           0
12367            0            0            0            0  …           0
7054             0            0            0            0  …           0

       country_34  country_35  country_36  country_37  country_38  country_39  \
32098           0           0           0           0           0           0
25206           0           0           0           0           0           0
23491           0           0           0           0           0           0
12367           0           0           0           0           0           0
7054            0           0           0           0           0           0

       country_40  country_41  country_42
32098           0           0           0
25206           0           0           0
23491           0           0           0
12367           0           0           0
```

```
7054           0           0           0

[5 rows x 108 columns]
```

[114]:
```python
cols = X_train.columns
```

[115]:
```python
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[116]:
```python
X_train = pd.DataFrame(X_train, columns=[cols])
X_test = pd.DataFrame(X_test, columns=[cols])
X_train.head()
```

[116]:
```
    age workclass_1 workclass_2 workclass_3 workclass_4 workclass_5  \
0  0.40         0.0         0.0         0.0         0.0         0.0
1  0.50        -1.0         1.0         0.0         0.0         0.0
2  0.55         0.0         0.0         0.0         0.0         0.0
3 -0.40         0.0         0.0         0.0         0.0         0.0
4 -0.70         0.0         0.0         0.0         0.0         0.0

  workclass_6 workclass_7 workclass_8 workclass_9 … country_33 country_34  \
0         0.0         0.0         0.0         0.0 …        0.0        0.0
1         0.0         0.0         0.0         0.0 …        0.0        0.0
2         0.0         0.0         0.0         0.0 …        0.0        0.0
3         0.0         0.0         0.0         0.0 …        0.0        0.0
4         0.0         0.0         0.0         0.0 …        0.0        0.0

  country_35 country_36 country_37 country_38 country_39 country_40  \
0        0.0        0.0        0.0        0.0        0.0        0.0
1        0.0        0.0        0.0        0.0        0.0        0.0
2        0.0        0.0        0.0        0.0        0.0        0.0
3        0.0        0.0        0.0        0.0        0.0        0.0
4        0.0        0.0        0.0        0.0        0.0        0.0

  country_41 country_42
0        0.0        0.0
1        0.0        0.0
2        0.0        0.0
3        0.0        0.0
4        0.0        0.0

[5 rows x 108 columns]
```

Gaussian Naive Bayes classifier

[117]:
```python
# train a Gaussian Naive Bayes classifier on the training set
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
```

[117]:
```
GaussianNB()
```

[118]:
```python
y_pred = gnb.predict(X_test)
y_pred
```

[118]:
```
array([' <=50K', ' <=50K', ' >50K', …, ' >50K', ' <=50K', ' <=50K'],
      dtype='<U6')
```

[119]:
```python
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

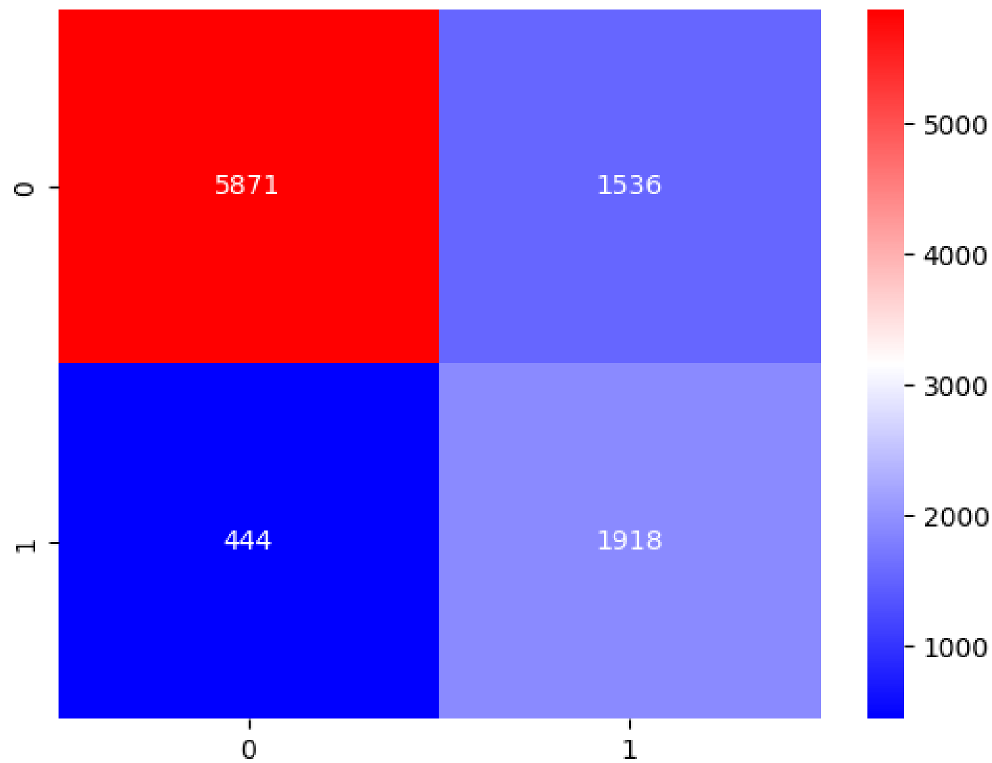
```
              precision    recall  f1-score   support

       <=50K       0.93      0.79      0.86      7407
        >50K       0.56      0.81      0.66      2362

    accuracy                           0.80      9769
   macro avg       0.74      0.80      0.76      9769
weighted avg       0.84      0.80      0.81      9769
```

[120]:
```python
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

accuracy = accuracy_score(y_test, y_pred)
print('Model accuracy score: {0:0.4f}'.format(accuracy))
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix\n\n', cm)
```

```
Model accuracy score: 0.7973
Confusion matrix

 [[5871 1536]
 [ 444 1918]]
```

[121]:
```python
sns.heatmap(cm, annot=True, fmt='d', cmap="bwr")
```

[121]:
```
<Axes: >
```


5871
1536
444
1918
0
1
0
1
5000
4000
3000
2000
1000