

Naiwny klasyfikator bayesowski

Celem ćwiczenia jest zapoznanie się z technikami konstrukcji naiwnego klasyfikatora bayesowskiego.

Ćwiczenie oparte jest o artykuł z [linku](#)

```
1 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
```

Na początku proszę wczytać plik **data.csv** używając jako separatora ',':

```
2 df = pd.read_csv("data.csv", delimiter=',')
df
```

	age	workclass	fnlwgt	education	education_num	marital_status	occupation
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty
...
32556	27	Private	257302	Assoc-acdm	12	Married-civ-spouse	Tech-support
32557	40	Private	154374	HS-grad	9	Married-civ-spouse	Machine-op-inspct
32558	58	Private	151910	HS-grad	9	Widowed	Adm-clerical
32559	22	Private	201490	HS-grad	9	Never-married	Adm-clerical
32560	52	Self-emp-inc	287927	HS-grad	9	Married-civ-spouse	Exec-managerial

32561 rows × 15 columns

W następnie sprawdź które z kolumn zawierają dane tekstowe lub liczbowe. W tym celu sprawdź wartość argumentu *dtype* z odpowiednim warunkiem:

```
4 types = df.dtypes
  columns_names = df.columns.values
  categorical = []
  for i in range(len(columns_names)):
      if types[i] == 'object':
          categorical.append(columns_names[i])

print('Dane tekstowe zawierają kolumny :\n\n', categorical)

Dane tekstowe zawierają kolumny :

['workclass', 'education', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'native_co
```

Sprawdź czy dane w kolumnach nie zawierają braków a jeżeli tak uzupełnij je według znanych Ci metod:

```
8 df = df.replace(" ?", 'Not a Country')
  print("\nLista wszystkich narodowości:\n\n",df['native_country'].value_counts())

  print("\nLista ras:\n\n",df['race'].value_counts())
  SetToData = pd.get_dummies(df,columns = ['income'])
```

Lista wszystkich narodowości:

United-States	29170
Mexico	643
Not a Country	583
Philippines	198
Germany	137
Canada	121
Puerto-Rico	114
El-Salvador	106
India	100
Cuba	95
England	90
Jamaica	81
South	80
China	75
Italy	73
Dominican-Republic	70
Vietnam	67
Guatemala	64
Japan	62
Poland	60
Columbia	59
Taiwan	51
Haiti	44
Iran	43
Portugal	37
Nicaragua	34
Peru	31
France	29
Greece	29
Ecuador	28
Ireland	24
Hong	20
Cambodia	19

Trinidad&Tobago	19
Laos	18
Thailand	18
Yugoslavia	16
Outlying-US(Guam-USVI-etc)	14
Honduras	13
Hungary	13
Scotland	12
Holand-Netherlands	1

Name: native_country, dtype: int64

Lista ras:

```

White      27816
Black      3124
Asian-Pac-Islander  1039
Amer-Indian-Eskimo  311
Other      271
Name: race, dtype: int64

```

Na przygotowanych danych przeprowadź proces tworzenia zbiorów uczących i testowych, tak by klasyfikator rozpoznawał do której z grup w kolumnie 'income' należy opisywana osoba

```
6 columns_names = SetToData.columns.values

X = SetToData[columns_names[0:-2]].to_numpy()
y = SetToData[columns_names[-1]].to_numpy().reshape(-1,1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

Przy pomocy biblioteki `category_encoders` przeprowadź proces kodowania zmiennych tekstowych z pozostałych kategorii na wartości liczbowe:

[illegible]

```
X_train = encoder.fit_transform(X_train)
```

```
X_test = encoder.transform(X_test)
```

[illegible]

```
C:\Users\Sebastian\AppData\Local\Programs\Python\Python39\lib\site-packages\category_encoders\one_hot
  for cat_name, class_ in values.iteritems():
C:\Users\Sebastian\AppData\Local\Programs\Python\Python39\lib\site-packages\category_encoders\one_hot
  for cat_name, class_ in values.iteritems():
C:\Users\Sebastian\AppData\Local\Programs\Python\Python39\lib\site-packages\category_encoders\one_hot
  for cat_name, class_ in values.iteritems():
```

Używając [GaussianNB](#) przeprowadź klasyfikację danych ze zbiorów testowych i treningowych. Podaj dokładność modelu i macierz błędów wraz z jej wykresem i interpretacją.

```
64 from sklearn.naive_bayes import GaussianNB
   from sklearn import metrics
   # instantiate the model
   gnb = GaussianNB()

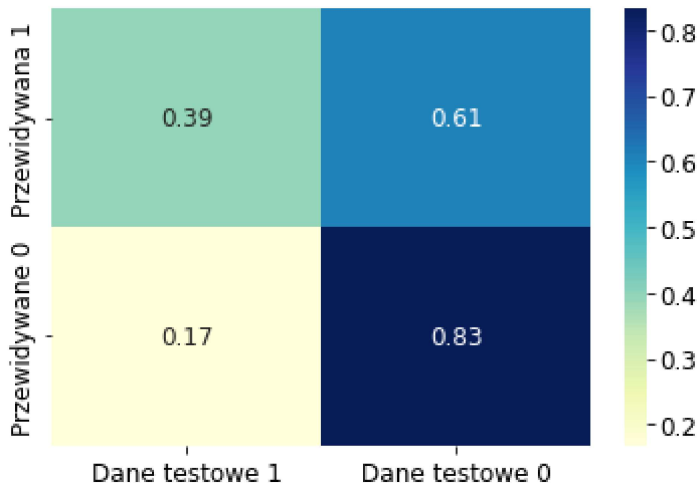
65 # fit the model
   gnb.fit(X_train,y_train)
   # gnb
   y_pred = gnb.predict(X_test)
   cm = metrics.confusion_matrix(y_test,y_pred,normalize='true')

   cm_matrix = pd.DataFrame(data=cm, columns=['Dane testowe 1', 'Dane testowe 0'],
                           index=['Przewidywana 1', 'Przewidywane 0'])

   sns.heatmap(cm_matrix, annot=True, cmap='YlGnBu')

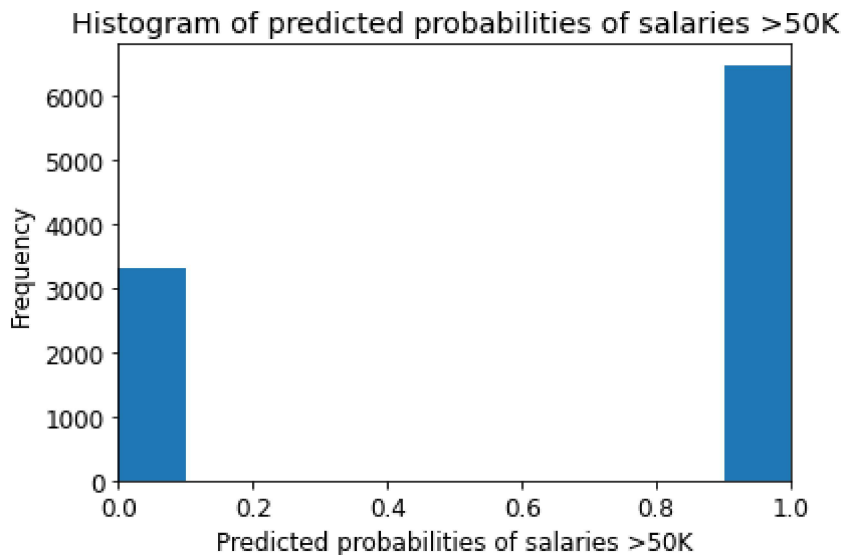
C:\Users\Sebastian\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\utils\validation.py
  y = column_or_1d(y, warn=True)
```

65 <AxesSubplot:>



```
75 y_prob = gnb.predict_proba(X_test)
   plt.rcParams['font.size'] = 12
   plt.hist(y_prob[:,1], bins = 10)
   plt.title('Histogram of predicted probabilities of salaries >50K')
   plt.xlim(0,1)
   plt.xlabel('Predicted probabilities of salaries >50K')
   plt.ylabel('Frequency')
```

75 Text(0, 0.5, 'Frequency')



```

25 correct = []
   uncorrect = []
   for i in range(len(y_pred)):
       if y_pred[i] != y_test[i]:
           uncorrect.append(i)
       else:
           correct.append(i)
   print(len(correct)/len(y_pred)*100, '% poprawnych typowań')
49.79015252328795 % poprawnych typowań

```

Przeprowadź uczenie klasyfikatora dla kolumn *race* i *native_country*. Podaj dokładność modeli i macierze błędów wraz z ich wykresami i interpretacją. Przedstaw wnioski od czego zależą otrzymane wyniki

```

59 X = SetToData[['native_country', 'race']].to_numpy()
   y = SetToData['income_ >50K'].to_numpy()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)
gnb = GaussianNB()
# fit the model
gnb.fit(X_train, y_train)
# gnb
y_pred = gnb.predict(X_test)
y_prob = gnb.predict_proba(X_test)

cm = metrics.confusion_matrix(y_test, y_pred, normalize='true')

cm_matrix = pd.DataFrame(data=cm, columns=['Dane testowe 1', 'Dane testowe 0'],
                        index=['Przewidywana 1', 'Przewidywane 0'])

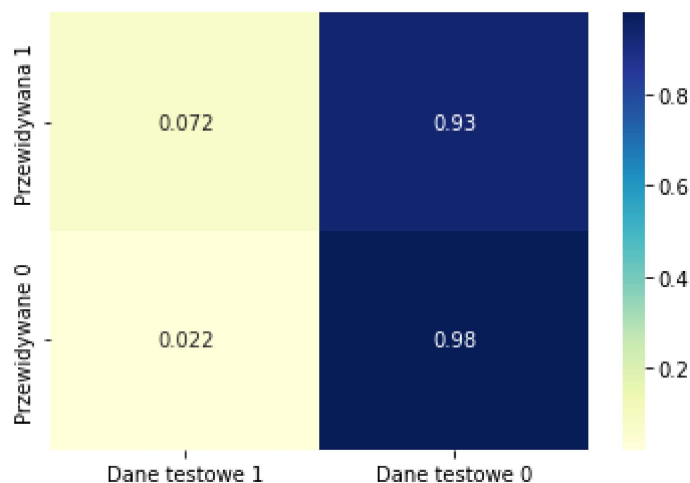
sns.heatmap(cm_matrix, annot=True, cmap='YlGnBu')
print(metrics.classification_report(y_test, y_pred))

C:\Users\Sebastian\AppData\Local\Programs\Python\Python39\lib\site-packages\category_encoders\one_hot
  for cat_name, class_ in values.iteritems():
C:\Users\Sebastian\AppData\Local\Programs\Python\Python39\lib\site-packages\category_encoders\one_hot
  for cat_name, class_ in values.iteritems():

precision    recall  f1-score   support

```

	0	0.91	0.07	0.13	7407
	1	0.25	0.98	0.40	2362
accuracy				0.29	9769
macro avg		0.58	0.52	0.27	9769
weighted avg		0.75	0.29	0.20	9769



```

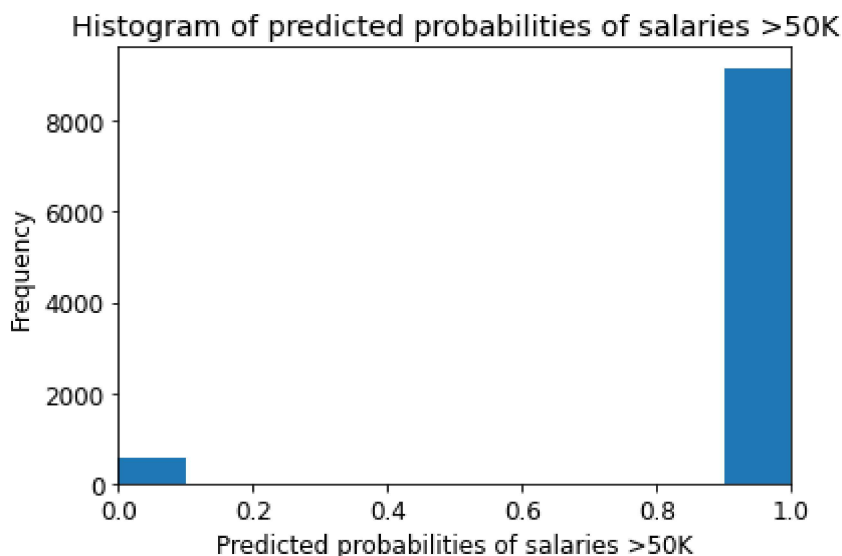
60 plt.rcParams['font.size'] = 12
plt.hist(y_prob[:,1], bins = 10)
plt.title('Histogram of predicted probabilities of salaries >50K')
plt.xlim(0,1)
plt.xlabel('Predicted probabilities of salaries >50K')
plt.ylabel('Frequency')

```

```

60 Text(0, 0.5, 'Frequency')

```



Wnioski:

W porównaniu z metodami stosowanymi na poprzednich zajęciach klasyfikator bayesowski daje większy procent poprawnych typowań wyniku, rozkład prawdopodobieństwa wyników jest zdecydowanie bardziej skrajny, praktycznie nie występują dane testowe dla których przewidywane prawdopodobieństwo wystąpienia jedynki nie jest liczbą bliską 100% lub 0%.

