

Sprawozdanie z Laboratorium uczenia Maszynowego

Bartłomiej Kieżun

Nr.Albumu 79535

Klasyfikacja e-maili jako SPAM	1
Przewidywanie przeżycia pasażerów Titanica	2
Klasyfikacja odmian wina.....	3

Klasyfikacja e-maili jako SPAM

Celem algorytmu jest klasyfikacja wiadomości, która automatycznie rozpoznaje czy dana wiadomość jest wiadomością SPAM czy zwykłą.

Zbiór danych: SMS Spam Collection Dataset

Link: <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset>

Plik posiada dwie kolumny V1 oraz V2, które odpowiadają Spam lub Ham oraz treść wiadomości. W kodzie zostały użyte zmienne label oraz message.

```
df.columns = ['label', 'message']
```

Skonwertowano etykiety do wartości 1 i 0

```
df['label_num'] = df['label'].map({'ham': 0, 'spam': 1})
```

Podzielono dane na zbiór treningowy oraz testowy

```
# zbiór treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y, test_size=0.2, random_state=42)
```

Wytrenowano klasyfikator Native Bayes przy użyciu MultinomialNB

```
model = MultinomialNB()
model.fit(X_train_counts, y_train)
```

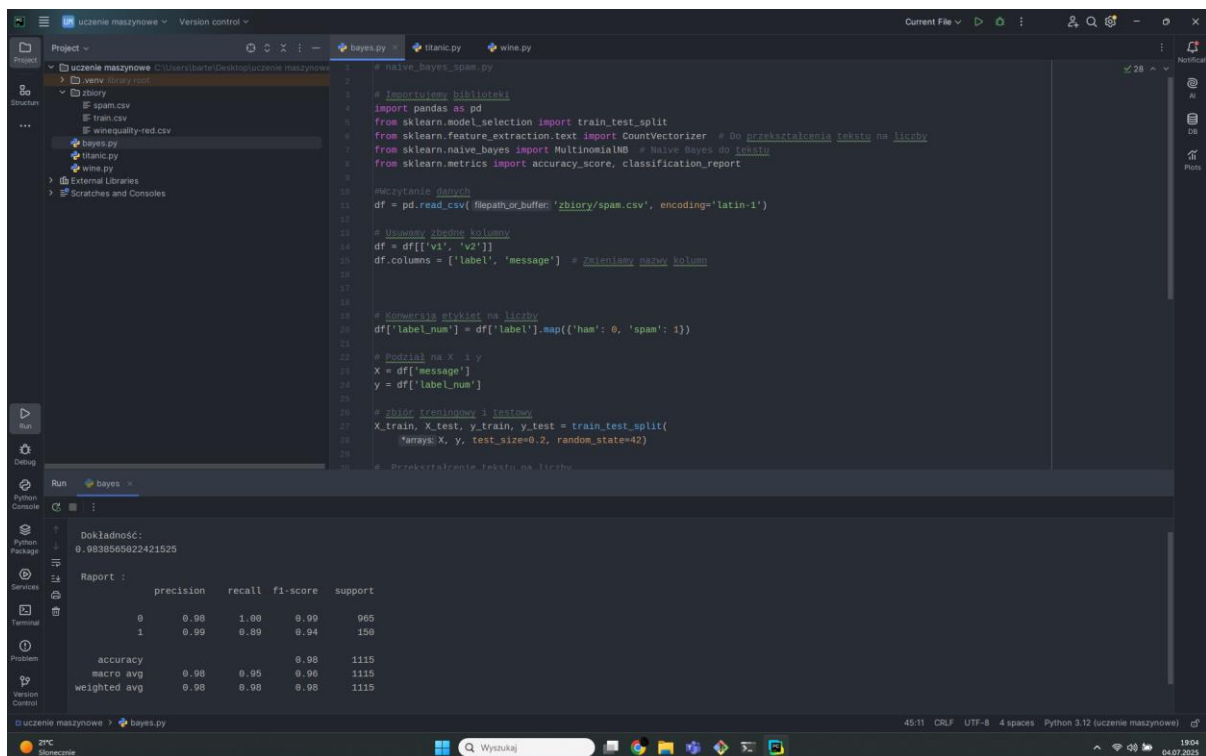
Obliczono dokładność i macierz konfuzji.

```
print("\n Dokładność:")
print(accuracy_score(y_test, y_pred))

print("\nRaport klasyfikacji:")
print(classification_report(y_test, y_pred))
```

Model osiągnął dokładność na poziomie **0.98**.

Klasyfikator skutecznie odróżnia wiadomości SPAM



```
# naive_bayes_spam.py
# Importujemy biblioteki
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report

# Wczytanie danych
df = pd.read_csv('zbiory/spam.csv', encoding='latin-1')

# Rozdzielamy kolumny
df[['v1', 'v2']]
df.columns = ['label', 'message']

# Konwersja stylów na liczby
df['label_num'] = df['label'].map({'ham': 0, 'spam': 1})

# Podział na X i y
X = df['message']
y = df['label_num']

# Podział na treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)

# Przekształcenie tekstu na liczby
```

Run

Dokładność:
0.989565622421525

Raport :

	precision	recall	f1-score	support
0	0.98	1.00	0.99	965
1	0.99	0.89	0.94	150
accuracy			0.98	1115
macro avg	0.98	0.95	0.96	1115
weighted avg	0.98	0.98	0.98	1115

Przewidywanie przeżycia pasażerów Titanica

Celem algorytmu jest klasyfikacja osób na podstawie klasy podróży, płci oraz wieku.

Zbiór danych: S Titanic - Machine Learning from Disaster

Link: <https://www.kaggle.com/competitions/titanic/data>

Wartość tabeli Sex została podzielona na wartości liczbowe 0-Mężczyzna oraz 1-Kobieta.

```
df = df[['Survived', 'Pclass', 'Sex', 'Age']].dropna()
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
```

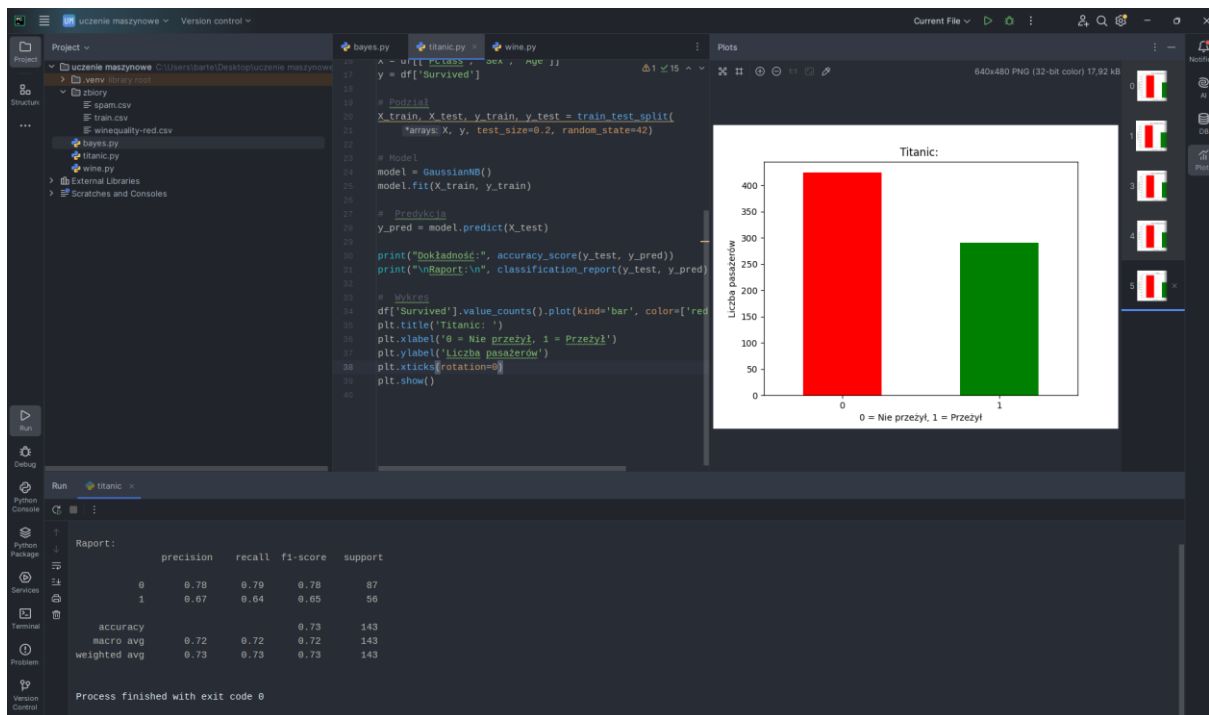
Zbiór podzielony na treningowy oraz testowy.

```
X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y, test_size=0.2, random_state=42)
```

Wytrenowany został klasyfikator GaussianNB

```
model = GaussianNB()
model.fit(X_train, y_train)
```

Model pokazano graficznie za pomocą biblioteki Matploib gdzie 0 oznacza, że nie przeżył, a 1, że przeżył.



Model osiągnął dokładność na poziomie **0.73**.

Klasyfikacja odmian wina

Celem algorytmu jest rozpoznawanie rodzaju wina na podstawie zestawu cech chemicznych.

Zbiór danych: Red Wine Quality

Link: <https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009>

Podzielono dane cechy i etykiety

```
X = df.drop(labels='quality', axis=1)
y = df['quality']
```

Podzielono zbiór na treningowy oraz testowy

```
X_train, X_test, y_train, y_test = train_test_split(
    *arrays: X, y, test_size=0.2, random_state=42)
```

Wytrenowany klasyfikator GaussianNB

```
model = GaussianNB()
model.fit(X_train, y_train)
```

Dokonano predykcji

```
y_pred = model.predict(X_test)

print("Dokładność:", accuracy_score(y_test, y_pred))
print("\n Raport:\n", classification_report(y_test, y_pred))
```

Model pokazano graficznie za pomocą biblioteki Matplotlib

