**Лабораторна робота №6**

**НАЇВНИЙ БАЙЄС В PYTHON**

**Мета роботи:** набути навичок працювати з даними і опонувати роботу у Python з використанням теореми Байэса.

Хід роботи:

**Завдання 6.1** Ретельно опрацювати теоретичні відомості: (Опрацьовано)

* теорему Байєса;
* які типи наївного байєсівського класифікатора є;
* де використовується Наївний Байєс.

**Завдання 6.2** Ретельно розібрати приклад: прогнозування з використанням теореми Байєса. (Опрацьовано)

**Завдання 6.3** Використовуя данні з пункту 2 визначити відбудеться матч при наступних погодних умовах чи ні: Розрахунки провести з використанням Python.

|  |  |
| --- | --- |
| Варіант | Умова |
| 7 | Outlook = Overcast, Humidity = High, Wind = Strong |

Лістинг коду:

class NaiveBayes:  
 def \_\_init\_\_(self, likelihoods, class\_probs):  
 self.likelihoods = likelihoods  
 self.class\_probs = class\_probs  
  
 def calculate\_probability(self, features):  
 *"""Обчислює ймовірність кожного класу для заданих ознак."""* probs = {}  
 for class\_name, class\_prob in self.class\_probs.items():  
 prob = class\_prob  
 for feature, value in features.items():  
 prob \*= self.likelihoods[feature][value][class\_name]  
 probs[class\_name] = prob  
 return probs  
  
 def normalize\_probabilities(self, probs):  
 *"""Нормалізує ймовірності."""* total = sum(probs.values())  
 return {class\_name: prob/total for class\_name, prob in probs.items()}

# Дані  
likelihoods = {  
 "Outlook": {"Overcast": {"Yes": 4/10, "No": 0/4}},  
 "Humidity": {"High": {"Yes": 3/9, "No": 4/5}},  
 "Wind": {"Strong": {"Yes": 3/9, "No": 3/5}}  
}

class\_probs = {"Yes": 10/14, "No": 4/14}  
  
# Умова  
features = {"Outlook": "Overcast", "Humidity": "High", "Wind": "Strong"}  
  
# Обчислення  
model = NaiveBayes(likelihoods, class\_probs)  
probs = model.calculate\_probability(features)  
normalized\_probs = model.normalize\_probabilities(probs)  
  
# Виведення результату  
print("Ймовірності:")  
for class\_name, prob in normalized\_probs.items():  
 print(f" {class\_name}: {prob}")

Результат виконання

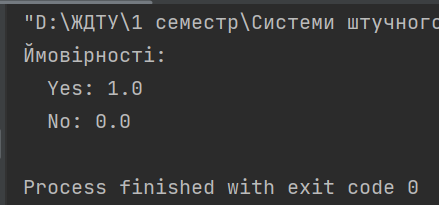


Рисунок 1 - Результат виконання програми

**Завдання 6.4** Застосуєте методи байєсівського аналізу до набору даних про ціни на квитки на іспанські високошвидкісні залізниці.

Лістинг коду:

import pandas as pd  
import matplotlib.pyplot as plt  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import BayesianRidge  
from sklearn.metrics import mean\_squared\_error  
from sklearn.preprocessing import StandardScaler  
  
# Завантаження даних  
url = "https://raw.githubusercontent.com/susanli2016/Machine-Learning-with-Python/master/data/renfe\_small.csv"  
df = pd.read\_csv(url)  
  
# Перевірка на пропущені значення  
df['price'] = df['price'].fillna(df['price'].mean()) # Заповнюємо пропущені ціни середнім значенням  
  
# Перетворення дати в тип datetime  
df['start\_date'] = pd.to\_datetime(df['start\_date'])

df['insert\_date'] = pd.to\_datetime(df['insert\_date'])  
  
# Створення нової колонки 'days\_to\_start' для кількості днів до поїздки  
df['days\_to\_start'] = (df['start\_date'] - df['insert\_date']).dt.days  
  
# Вибір лише числових та категоріальних змінних для аналізу  
df\_encoded = pd.get\_dummies(df, columns=['train\_type', 'train\_class', 'fare'])  
  
# Вибір фіч та цільової змінної (ціна)  
X = df\_encoded.drop(['insert\_date', 'origin', 'destination', 'start\_date', 'end\_date', 'price'], axis=1)  
y = df\_encoded['price']  
  
# Масштабування даних (особливо важливо для регресії)  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# Розбиття даних на навчальну та тестову вибірки  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
# Створення моделі байєсівської регресії  
model = BayesianRidge()  
  
# Навчання моделі  
model.fit(X\_train, y\_train)  
  
# Прогнозування  
y\_pred = model.predict(X\_test)  
  
# Оцінка моделі  
mse = mean\_squared\_error(y\_test, y\_pred)  
r2 = model.score(X\_test, y\_test)

# Виведення результатів  
print(f'Mean Squared Error: {mse}')  
print(f'R-squared: {r2}')  
  
# Побудова графіків  
# 1. Графік розсіяння між реальними і передбаченими цінами  
plt.figure(figsize=(10, 6))  
plt.scatter(y\_test, y\_pred, color='blue', alpha=0.5)  
plt.plot([min(y\_test), max(y\_test)], [min(y\_test), max(y\_test)], color='red', linestyle='--') # Лінія ідеального прогнозу  
plt.title('Real vs Predicted Prices')  
plt.xlabel('Real Prices')  
plt.ylabel('Predicted Prices')  
plt.show()  
  
# 2. Гістограма різниці між реальними та передбаченими цінами  
plt.figure(figsize=(10, 6))  
plt.hist(y\_test - y\_pred, bins=20, color='green', edgecolor='black', alpha=0.7)  
plt.title('Difference between Real and Predicted Prices')  
plt.xlabel('Difference')  
plt.ylabel('Frequency')  
plt.show()

Результат виконання

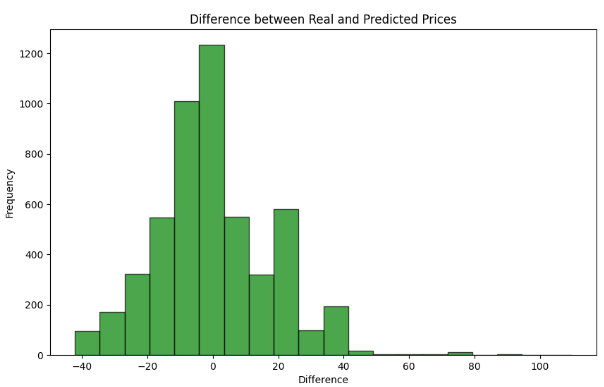


Рисунок 2 - Результат виконання програми

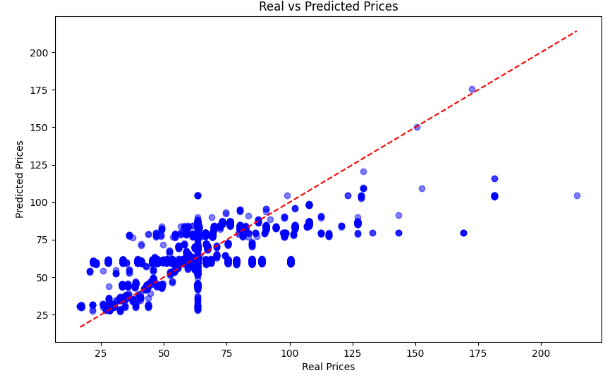


Рисунок 3 - Результат виконання програми

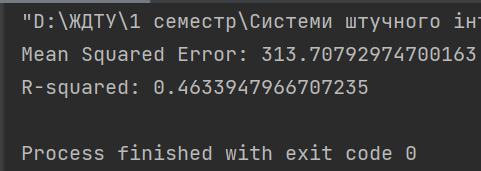


Рисунок 4 - Результат виконання програми

**Висновок:** набув практичних навичок роботи з даними і опонував роботу у Python з використанням теореми Байэса.