ЛАБОРАТОРНА РОБОТА № 6

**Наївний Байєс в Python**

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

Git: https://github.com/flekXD/SAI

**Завдання 1.**

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



import numpy as np

# Функція для розрахунку ймовірності для кожної таблиці (Outlook, Humidity, Wind)

def table\_operations(smt, table, row\_index):

    total\_sum = np.sum(table)

    # Сума чисел в рядку

    row\_sum = np.sum(table[row\_index])

    # Сума чисел в стовпці для Yes та No

    column\_Y = np.sum(table[:, 0])  # Сума стовпця "Yes"

    column\_N = np.sum(table[:, 1])  # Сума стовпця "No"

    # Обчислення ймовірності для Yes та No

    res\_Y = (smt / column\_Y) \* (column\_Y / total\_sum) / (row\_sum / total\_sum)

    res\_N = (smt / column\_N) \* (column\_N / total\_sum) / (row\_sum / total\_sum)

    return res\_Y, res\_N

# Функція для розрахунку ймовірностей для матчу

def calculate\_probability\_for\_match(outlook, humidity, wind):

    # Таблиці ймовірностей для Outlook, Humidity, Wind

    outlook\_table = np.array([

        [3, 2],  # Sunny (Yes, No)

        [4, 0],  # Overcast (Yes, No)

        [3, 2]   # Rainy (Yes, No)

    ])

    humidity\_table = np.array([

        [3, 4],  # High (Yes, No)

        [6, 1]   # Normal (Yes, No)

    ])

    wind\_table = np.array([

        [6, 2],  # Strong (Yes, No)

        [3, 3]   # Weak (Yes, No)

    ])

    # Задано умови для Outlook, Humidity, Wind

    if outlook == 'Sunny':

        outlook\_row\_index = 0

    elif outlook == 'Overcast':

        outlook\_row\_index = 1

    elif outlook == 'Rainy':

        outlook\_row\_index = 2

    else:

        raise ValueError("Invalid outlook condition")

    if humidity == 'High':

        humidity\_row\_index = 0

    elif humidity == 'Normal':

        humidity\_row\_index = 1

    else:

        raise ValueError("Invalid humidity condition")

    if wind == 'Strong':

        wind\_row\_index = 0

    elif wind == 'Weak':

        wind\_row\_index = 1

    else:

        raise ValueError("Invalid wind condition")

    # Розрахунок ймовірностей для Outlook, Humidity, Wind

    outlook\_Y, outlook\_N = table\_operations(3, outlook\_table, outlook\_row\_index)  # Потрібно передати правильне значення smt

    humidity\_Y, humidity\_N = table\_operations(3, humidity\_table, humidity\_row\_index)  # Аналогічно

    wind\_Y, wind\_N = table\_operations(6, wind\_table, wind\_row\_index)  # І для вітру, передаємо правильне число smt

    # Розрахунок P(Yes) та P(No) для всіх умов

    P\_yes\_result = outlook\_Y \* humidity\_Y \* wind\_Y \* 9/14  # P(Yes) (наприклад, для конкретного загального числа)

    P\_no\_result = outlook\_N \* humidity\_N \* wind\_N \* 5/14  # P(No)

    P\_yes= P\_yes\_result/(P\_yes\_result+P\_no\_result)

    P\_no = P\_no\_result/(P\_yes\_result+P\_no\_result)

    return P\_yes, P\_no

# Задано умови

outlook = 'Overcast'

humidity = 'High'

wind = 'Strong'

# Розрахунок ймовірностей для Yes і No

P\_yes, P\_no = calculate\_probability\_for\_match(outlook, humidity, wind)

# Виведення результатів

print(f"P(Yes|Outlook={outlook}, Humidity={humidity}, Wind={wind}) = {P\_yes:.6f}")

print(f"P(No|Outlook={outlook}, Humidity={humidity}, Wind={wind}) = {P\_no:.6f}")



З результатів прогнозування бачимо що ймовірність складає 64% на те чи буде гра при таких погодних умовах

**Завдання 2.**

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

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score

# Завантаження даних

url = "https://raw.githubusercontent.com/susanli2016/Machine-Learning-with-Python/refs/heads/master/data/renfe\_small.csv"

df = pd.read\_csv(url)

# Видаляємо рядки, де ціна відсутня або не є числом

df = df.dropna(subset=['price'])

df['price'] = pd.to\_numeric(df['price'], errors='coerce')

df = df.dropna(subset=['price'])

# Переводимо ціни в категорії (наприклад, дешево і дорого)

df['price\_category'] = df['price'].apply(lambda x: 1 if x > 100 else 0)  # Створення стовпця price\_category

# Вибір ознак для моделювання

features = ['origin', 'destination', 'train\_type', 'train\_class']  # Використовуємо ці стовпці для моделювання

df\_dummies = pd.get\_dummies(df[features], drop\_first=True)  # Перетворюємо категоріальні змінні в числові

# Додаємо стовпець price\_category назад в df\_dummies

df\_dummies['price\_category'] = df['price\_category'].astype(int)

# Розділяємо дані на тренувальні та тестові

X = df\_dummies.drop('price\_category', axis=1)

y = df\_dummies['price\_category']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Створення і тренування моделі наївного баєсівського класифікатора

model = GaussianNB()

model.fit(X\_train, y\_train)

# Оцінка моделі

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

# Фільтрація даних для цін менше 25 євро

low\_price\_count = df[df['price'] < 25].shape[0]

# Загальна кількість записів

total\_count = df.shape[0]

# Ймовірність отримати ціну менше 25 євро

probability = low\_price\_count / total\_count

print(f"Ймовірність отримати ціну менше 25 євро: {probability:.2%}")

print(f"Точність моделі: {accuracy:.2f}")



У цій роботі проаналізовано ціни на залізничні квитки та виконано їх класифікацію на "дорогі" й "дешеві" за допомогою наївного баєсівського класифікатора. Точність моделі склала 52%, що свідчить про її обмежену ефективність. Ймовірність отримати квиток за ціною менше 25 євро склала лише 2.37%, що вказує на рідкість бюджетних квитків. Для покращення результатів рекомендується використовувати більш складні моделі та додаткові ознаки, як-от сезонність або дати бронювання.

**Висновок**

У ході виконання лабораторної роботи №6 було досягнуто мету, поставлену на початку. Я здобув практичні навички роботи з даними та опанував використання теореми Байєса в Python. Було розглянуто основні принципи роботи Наївного Байєса, його переваги та обмеження.

Також реалізовано алгоритм Наївного Байєса для класифікації даних, що дозволило закріпити теоретичні знання на практиці. Аналіз отриманих результатів показав, що Наївний Байєс є ефективним інструментом для розв'язання задач класифікації, особливо для великих наборів даних.