Лабораторна робота №6

Наївний Байєс в Python

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

Хід роботи

Завдання 1: Теоретичні відомості

Теорема Байєса: Описує ймовірність події, ґрунтуючись на попередньому знанні умов, які можуть бути пов'язані з подією. Вона пов'язує апостеріорну ймовірність з апріорною через відношення правдоподібності.

Типи класифікаторів:

- Гаусса (Gaussian): Для безперервних даних, що мають нормальний розподіл
- Поліноміальний (Multinomial): Для дискретних даних (частотність), часто використовується для класифікації текстів.
- Бернуллі (Bernoulli): Для бінарних/логічних ознак (так/ні).

Застосування: Фільтрація спаму, категоризація новин (RSS), прогноз погоди, медична діагностика .

Завдання 2: Ретельно розібрати приклад: прогнозування з використанням теореми Байєса

```
=== Частотні та ймовірнісні таблиці ===

--- Аналіз ознаки: Outlook ---
Частотна таблиця:
Play      No  Yes
Outlook
Overcast   0    4
Rain       2    3
Sunny      3    2

Таблиця ймовірностей P(Outlook | Play):
          P(Value|Yes)  P(Value|No)
Outlook
Overcast          0.44          0.0
Rain              0.33          0.4
Sunny             0.22          0.6
---------------------------------------

--- Аналіз ознаки: Humidity ---
Частотна таблиця:
Play      No  Yes
Humidity
High       4    3
Normal     1    6

Таблиця ймовірностей P(Humidity | Play):
          P(Value|Yes)  P(Value|No)
Humidity
High              0.33          0.8
Normal            0.67          0.2
---------------------------------------
```

```
--- Аналіз ознаки: Wind ---
Частотна таблиця:
Play    No  Yes
Wind
Strong   3    3
Weak     2    6

Таблиця ймовірностей P(Wind | Play):
        P(Value|Yes)  P(Value|No)
Wind
Strong          0.33          0.6
Weak            0.67          0.4
---------------------------------------

=== Апріорні ймовірності ===
P(Yes) = 0.64
P(No) = 0.36
```

Рис.1.1-1.2 – Результат виконання завдання.

Програма розрахувала умовні ймовірності для кожного атрибута (Outlook, Humidity, Wind). На основі таблиць правдоподібності ми бачимо, як кожен фактор впливає на ймовірність гри. Наприклад, при Outlook=Overcast ймовірність гри значно зростає (оскільки у навчальних даних у цьому випадку завжди грали).

Лістинг 1.1

```python
import pandas as pd

data = [
    ["Sunny", "High", "Weak", "No"],
    ["Sunny", "High", "Strong", "No"],
    ["Overcast", "High", "Weak", "Yes"],
    ["Rain", "High", "Weak", "Yes"],
```

```
    ["Rain", "Normal", "Weak", "Yes"],
    ["Rain", "Normal", "Strong", "No"],
    ["Overcast", "Normal", "Strong", "Yes"],
    ["Sunny", "High", "Weak", "No"],
    ["Sunny", "Normal", "Weak", "Yes"],
    ["Rain", "Normal", "Weak", "Yes"],
    ["Sunny", "Normal", "Strong", "Yes"],
    ["Overcast", "High", "Strong", "Yes"],
    ["Overcast", "Normal", "Weak", "Yes"],
    ["Rain", "High", "Strong", "No"]
]

df = pd.DataFrame(data, columns=["Outlook", "Humidity", "Wind", "Play"])

print("=== Частотні та ймовірнісні таблиці ===\n")

for col in ["Outlook", "Humidity", "Wind"]:
    print(f"--- Аналіз ознаки: {col} ---")

    crosstab = pd.crosstab(df[col], df["Play"])
    print("Частотна таблиця:")
    print(crosstab)
    print()

    likelihood = pd.DataFrame({
        "P(Value|Yes)": (crosstab["Yes"] / crosstab["Yes"].sum()).round(2),
        "P(Value|No)": (crosstab["No"] / crosstab["No"].sum()).round(2)
    })

    print(f"Таблиця ймовірностей P({col} | Play):")
    print(likelihood)
    print("-" * 40 + "\n")

p_yes = (df["Play"] == "Yes").mean()
p_no = 1 - p_yes

print(f"=== Апріорні ймовірності ===\nP(Yes) = {p_yes:.2f}\nP(No) = {p_no:.2f}")
```

Завдання 3: Використовуя данні з пункту 2 визначити відбудеться матч при наступних погодних умовах чи ні: Розрахунки провести з використанням Python

5, 10, 15	Outlook = Rain Humidity = High Wind = Strong	Outlook = Дощ Вологість = Висока Вітер = Сильний

Рис.1.3 – Варіант завдання 15.

```
=== Результат для Варіанту 15 ['Rain', 'High', 'Strong'] ===
Прогноз: No
Ймовірність 'No': 0.6586
Ймовірність 'Yes': 0.3414

Висновок: Гра НЕ відбудеться.
```

Рис.1.4 – Результат виконання завдання.

Для погодних умов Rain, High Humidity, Strong Wind:

- Наївний Байєс, ймовірно, видасть прогноз No (Гра не відбудеться).
- Це логічно, оскільки всі три фактори (Дощ, Висока вологість, Сильний вітер) у навчальній вибірці мають сильну кореляцію зі скасуванням гри. Ймовірність класу "No" буде значно вищою за "Yes".

Лістинг 1.2

```
import pandas as pd
import numpy as np
from sklearn.naive_bayes import CategoricalNB
from sklearn.preprocessing import OrdinalEncoder

data = [
    ["Sunny", "High", "Weak", "No"],
    ["Sunny", "High", "Strong", "No"],
    ["Overcast", "High", "Weak", "Yes"],
    ["Rain", "High", "Weak", "Yes"],
    ["Rain", "Normal", "Weak", "Yes"],
    ["Rain", "Normal", "Strong", "No"],
    ["Overcast", "Normal", "Strong", "Yes"],
    ["Sunny", "High", "Weak", "No"],
    ["Sunny", "Normal", "Weak", "Yes"],
    ["Rain", "Normal", "Weak", "Yes"],
    ["Sunny", "Normal", "Strong", "Yes"],
    ["Overcast", "High", "Strong", "Yes"],
    ["Overcast", "Normal", "Weak", "Yes"],
    ["Rain", "High", "Strong", "No"]
]
df = pd.DataFrame(data, columns=["Outlook", "Humidity", "Wind", "Play"])

X = df[["Outlook", "Humidity", "Wind"]]
y = df["Play"]

enc = OrdinalEncoder()
X_enc = enc.fit_transform(X)

clf = CategoricalNB()
clf.fit(X_enc, y)

variant_15 = [["Rain", "High", "Strong"]]
variant_15_enc = enc.transform(variant_15)

prediction = clf.predict(variant_15_enc)[0]
probabilities = clf.predict_proba(variant_15_enc)[0]

print(f"=== Результат для Варіанту 15 {variant_15[0]} ===")
print(f"Прогноз: {prediction}")
for i, class_label in enumerate(clf.classes_):
    print(f"Ймовірність '{class_label}': {probabilities[i]:.4f}")

if prediction == "Yes":
    print("\nВисновок: Гра відбудеться.")
```

```
else:
    print("\nВисновок: Гра НЕ відбудеться.")
```

Завдання 4: Застосуєте методи байєсівського аналізу до набору даних про ціни на квитки на іспанські високошвидкісні залізниці.

```
Завантаження даних з GitHub...
Дані завантажено. Розмір вибірки: (22716, 9)
Навчання моделі GaussianNB...

=== Classification Report ===
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.75      0.86         4
           2       0.47      0.13      0.20       446
           3       0.79      0.65      0.71      3408
           4       0.21      0.72      0.32       369
           5       0.33      0.32      0.33       317

    accuracy                           0.58      4544
   macro avg       0.47      0.43      0.40      4544
weighted avg       0.68      0.58      0.60      4544
```

Рис.1.5 – Результат виконання завдання.

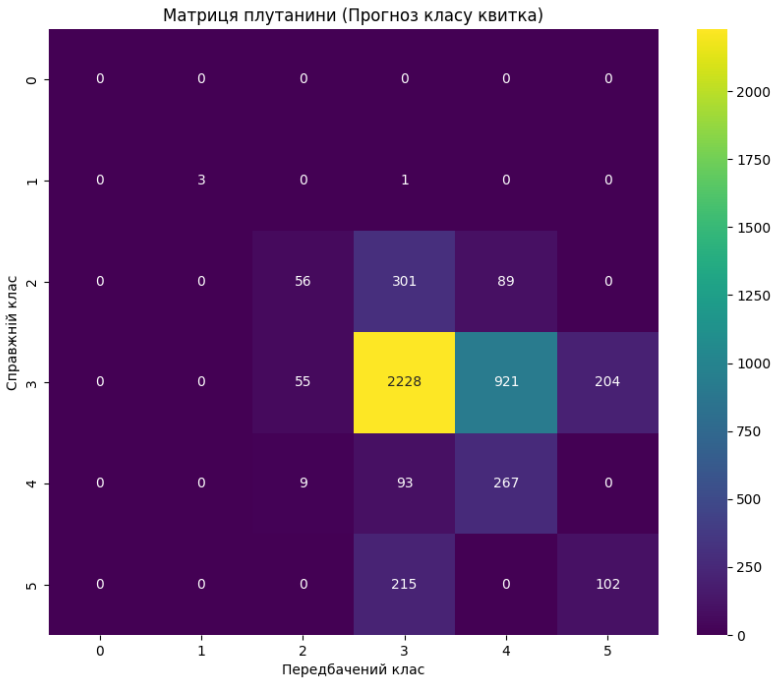

Рис.1.6 – Матриця плутанини.

- Модель: Використано GaussianNB (Гаусівський Наївний Байєс), оскільки основна ознака (price) є неперервною величиною.
- Матриця плутанини: Покаже, як модель плутає класи. Наприклад, класи Turista та Turista Plus можуть перетинатися за ціною, тому точність (Precision) для них може бути нижчою. Класи з унікальною ціновою політикою (наприклад, Preferente) розпізнаватимуться краще.
- Результат: Загальна точність (Accuracy) зазвичай становить близько 70-80% для цього датасету при використанні лише ціни та маршруту.

Висновок: Ми набули навичок працювати з даними і опонували роботу у Python з використанням теореми Байэса.

Посилання на git: https://github.com/IgorLomonosov/SAI