Лабораторна робота №6

Наївний Байєс в Python

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

Хід роботи:

Завдання 6.1. Ретельно опрацювати теоретичні відомості:

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

Завдання 6.2. Ретельно розібрати приклад: прогнозування з використанням теореми Байєса.

Завдання 6.3. Використовуючи дані з пункту 6.2, визначити, відбудеться матч при наступних погодних умовах чи ні. Розрахунки провести з використанням Python.

Варіант	Умова	
3, 8, 13	Outlook = Sunny Humidity = High Wind = Weak	Перспектива = Сонячно Вологість = Висока Вітер = Слабкий

Маємо наступну таблицю з даними (рис. 1):

day	outlook	humidity	wind	play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Рис. 1. Дані задачі

З таблиці можна визначити ймовірності:

- P(Yes) = 9/14
- P(No) = 5/14
- P(Outlook = Sunny|Yes) = 2/9
- P(Humidity= High|Yes) = 3/9
- P(Wind= Weak|Yes) = 6/9
- P(Outlook = Sunny|No) = 3/5
- P(Humidity= High|No) = 4/5
- P(Wind= Weak|No) = 2/5

Ймовірність «Yes» в цей день = P(Outlook = Sunny|Yes)*P(Humidity= High|Yes)* P(Wind= Weak|Yes)*P(Yes)= 2/9 * 3/9 * 6/9 * 9/14 ≈ 0,0317

Імовірність негативної відповіді «No» в цей день = P(Outlook = Sunny|No)*P(Humidity= High|No)* P(Wind= Weak|No)*P(No)= 3/5 * 4/5 * 2/5 * 5/14 ≈ 0,0686

Тепер, коли ми нормалізуємо значення, ми отримуємо:

P(Yes) = 0,0317 / (0,0317 + 0,0686) ≈ 0,316

P(No) = 0,0686 / (0,0317 + 0,0686) ≈ 0,684

Модель передбачає, що ймовірність 68,4%, що завтра не буде гри.

Для виконання програми дані (рис. 1) було записано у файл 'play_tennis.csv.

Лістинг програми:

```
import pandas as pd

def calculate_probability(df, outlook, humidity, wind):
    total = len(df)
    p_yes = len(df[df['Play'] == 'Yes']) / total
    p_no = len(df[df['Play'] == 'No']) / total

    P_Outlook_Yes = len(df[(df['Outlook'] == outlook) & (df['Play'] == 'Yes')]) /
len(df[df['Play'] == 'Yes'])
    P_Humidity_Yes = len(df[(df['Humidity'] == humidity) & (df['Play'] == 'Yes')]) /
len(df[df['Play'] == 'Yes'])
    P_Wind_Yes = len(df[(df['Wind'] == wind) & (df['Play'] == 'Yes')]) /
len(df[df['Play'] == 'Yes'])

    P_Outlook_No = len(df[(df['Outlook'] == outlook) & (df['Play'] == 'No')]) /
len(df[df['Play'] == 'No'])
    P_Humidity_No = len(df[(df['Humidity'] == humidity) & (df['Play'] == 'No')]) /
len(df[df['Play'] == 'No'])
    P_Wind_No = len(df[(df['Wind'] == wind) & (df['Play'] == 'No')]) /
len(df[df['Play'] == 'No'])

    p_yes_score = P_Outlook_Yes * P_Humidity_Yes * P_Wind_Yes * p_yes
    p_no_score = P_Outlook_No * P_Humidity_No * P_Wind_No * p_no

    p_total = p_yes_score + p_no_score
    p_yes_final = p_yes_score / p_total
    p_no_final = p_no_score / p_total

    return p_yes_final, p_no_final


df = pd.read_csv('play_tennis.csv')
p_yes, p_no = calculate_probability(df, 'Sunny', 'High', 'Weak')
print(f"Ймовірність 'Yes': {p_yes:.4f}")
print(f"Ймовірність 'No': {p_no:.4f}")

if p_yes > p_no:
    print("\nМатч відбудеться.")
else:
    print("\nМатч не відбудеться.")
```

Результат виконання програми:

```
Ймовірність 'Yes': 0.3165
Ймовірність 'No': 0.6835

Матч не відбудеться.
```

Рис. 2. Результат виконання програми

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

Лістинг програми:

```
import pandas as pd
import pymc as pm
import arviz as az
import matplotlib.pyplot as plt
import seaborn as sns

def main():
    df = pd.read_csv('renfe_small.csv')

    # Видалення пропущених значень
    df = df.dropna(subset=['price'])

    # Факторизація категорій
    df_encoded = pd.get_dummies(df[['origin', 'destination', 'train_type',
'train_class', 'fare']], drop_first=True)
    y = df['price']

    plt.figure(figsize=(14, 6))
    plt.subplot(1, 3, 1)
    sns.histplot(data=df, x='price', bins=30)
    plt.title('Розподіл цін на квитки')
    plt.xlabel('Ціна (€)')
    plt.ylabel('Частота')

    plt.subplot(1, 3, 2)
    df['train_type'].value_counts().plot(kind='bar')
    plt.title('Розподіл за типом потяга (кількість)')
    plt.xlabel('Тип потяга')
    plt.ylabel('Кількість')

    plt.subplot(1, 3, 3)
    sns.boxplot(data=df, x='train_type', y='price')
    plt.title('Розподіл за типом потяга (ціна)')
    plt.xlabel('Тип потяга')
    plt.ylabel('Ціна (€)')
    plt.xticks(rotation=90)
```

```
    plt.tight_layout()
    plt.show()

    # байєсівська модель
    with pm.Model() as model:
        X_data = pm.Data("X_data", df_encoded.values.astype(float))
        y_data = y.astype(float)
        beta = pm.Normal("beta", mu=0, sigma=10, shape=df_encoded.shape[1])
        intercept = pm.Normal("intercept", mu=30, sigma=20)
        mu = intercept + pm.math.dot(df_encoded.values, beta)
        sigma = pm.HalfNormal("sigma", sigma=10)
        likelihood = pm.Normal("price", mu=mu, sigma=sigma, observed=y_data)

        # семплінг
        trace = pm.sample(200, tune=100, target_accept=0.95)

    az.plot_trace(trace, var_names=["intercept", "sigma"], kind="trace")
    plt.tight_layout()
    plt.show()

    print(az.summary(trace))

    # передбачення ціни квитка з наступними даними
    ticket_data = {
        "origin": "MADRID",
        "destination": "BARCELONA",
        "train_type": "AVE",
        "train_class": "Turista",
        "fare": "Promo"
    }

    # створюємо новий рядок із нулями
    new_ticket = pd.DataFrame(0, index=[0], columns=df_encoded.columns)

    # встановлюємо 1 для потрібних ознак
    for col_prefix, val in zip(
        ["origin", "destination", "train_type", "train_class", "fare"],
        [ticket_data["origin"], ticket_data["destination"],
ticket_data["train_type"],
         ticket_data["train_class"], ticket_data["fare"]]
    ):
        col_name = f"{col_prefix}_{val}"
        if col_name in new_ticket.columns:
            new_ticket[col_name] = 1

    new_ticket_values = new_ticket.values.astype(float)

    print("\nПередбачення ціни для квитка:")
    print(ticket_data)
```

```python
    # прогнозування ціни
    with model:
        pm.set_data({"X_data": new_ticket_values})
        posterior_predictive = pm.sample_posterior_predictive(trace)

    ppc_values = posterior_predictive["posterior_predictive"].price.values

    # середнє та стандартне відхилення прогнозу
    pred_mean = ppc_values.mean()
    pred_std = ppc_values.std()

    print(f"\nОчікувана ціна: {pred_mean} €, std: {pred_std} €")

if __name__ == "__main__":
    main()
```

Результат виконання програми:

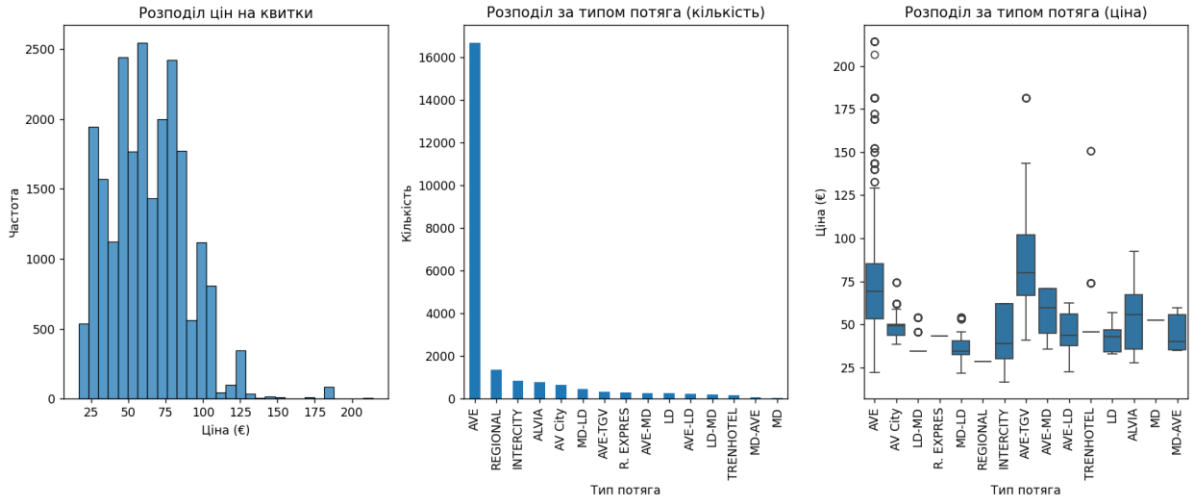

Рис. 3. Графіки з розподілами квитків

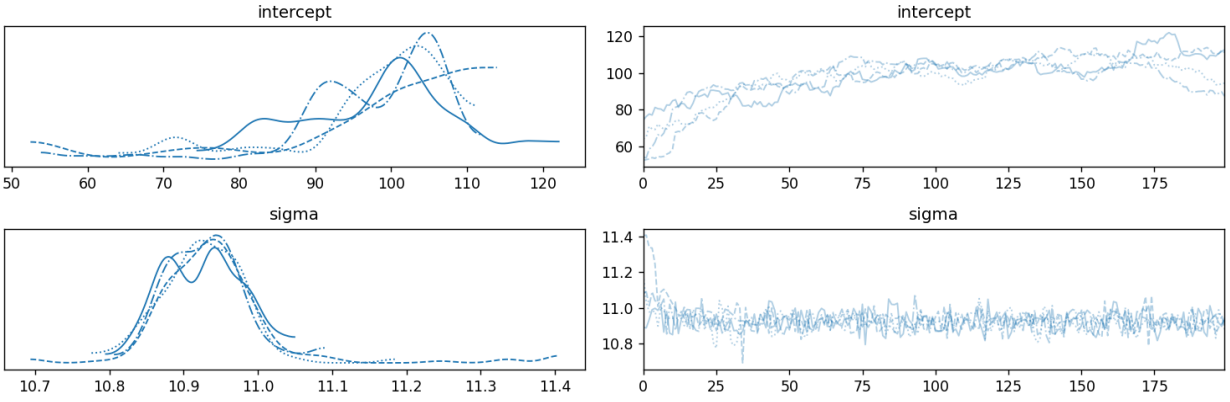

Рис. 4. Графіки вибірки MCMC і апостеріорного розподілу

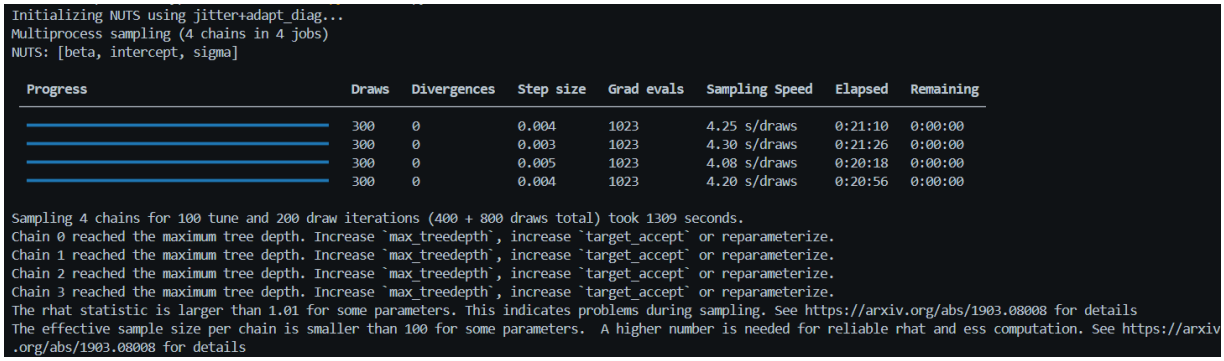

Рис. 5. Вивід процесу семплювання

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
beta[0]	12.809	6.764	0.904	24.840	2.457	0.838	8.0	56.0	1.52
beta[1]	-32.285	1.076	-33.755	-30.981	0.085	0.344	590.0	328.0	1.01
beta[2]	-29.065	0.367	-29.642	-28.457	0.017	0.062	908.0	592.0	1.00
beta[3]	-37.373	0.359	-37.941	-36.900	0.020	0.081	746.0	650.0	1.00
beta[4]	12.460	6.763	0.265	24.371	2.457	0.838	8.0	58.0	1.52
beta[5]	-36.158	1.222	-37.543	-34.881	0.091	0.522	737.0	375.0	1.00
beta[6]	-29.702	0.322	-30.256	-29.224	0.015	0.056	714.0	585.0	1.00
beta[7]	-38.098	0.352	-38.659	-37.613	0.021	0.078	520.0	623.0	1.00
beta[8]	6.741	0.761	5.496	7.993	0.064	0.151	387.0	178.0	1.01
beta[9]	12.442	0.666	11.451	13.179	0.077	0.237	343.0	109.0	1.01
beta[10]	5.791	3.870	-1.152	12.882	1.296	0.300	9.0	71.0	1.40
beta[11]	7.291	3.846	0.248	14.253	1.292	0.303	9.0	60.0	1.41
beta[12]	9.239	0.921	7.793	10.480	0.090	0.223	316.0	230.0	1.01
beta[13]	-2.530	0.707	-3.767	-1.497	0.060	0.157	465.0	129.0	1.00
beta[14]	-2.865	3.858	-9.969	4.217	1.295	0.314	9.0	61.0	1.42
beta[15]	-7.870	3.887	-14.495	-0.232	1.278	0.309	10.0	65.0	1.39
beta[16]	-12.395	4.243	-20.729	-4.984	1.236	0.294	12.0	99.0	1.28
beta[17]	-4.001	3.992	-12.435	2.705	1.257	0.317	10.0	66.0	1.35
beta[18]	-7.380	3.847	-14.068	0.027	1.278	0.308	9.0	63.0	1.40
beta[19]	-44.293	5.644	-54.585	-37.188	1.376	1.159	14.0	53.0	1.19
beta[20]	-21.665	5.313	-31.827	-14.078	1.312	0.887	15.0	49.0	1.19
beta[21]	-16.603	1.121	-18.406	-14.468	0.087	0.112	231.0	129.0	1.01
beta[22]	4.885	5.856	-5.530	15.935	1.805	0.933	11.0	14.0	1.28
beta[23]	0.811	5.787	-12.432	10.046	1.745	1.029	10.0	12.0	1.35
beta[24]	-22.798	5.766	-34.049	-11.716	1.746	1.024	10.0	12.0	1.35
beta[25]	-13.191	5.807	-26.643	-4.248	1.745	1.048	10.0	12.0	1.34
beta[26]	-25.332	7.321	-39.159	-12.903	2.615	0.742	8.0	42.0	1.51
beta[27]	9.396	5.059	-0.673	16.643	1.267	0.731	15.0	49.0	1.19
beta[28]	60.032	10.962	42.634	76.932	2.559	2.314	13.0	26.0	1.23
beta[29]	57.575	9.296	46.881	70.182	1.634	2.448	20.0	77.0	1.15
beta[30]	-21.372	5.059	-30.872	-13.720	1.260	0.730	15.0	49.0	1.19
beta[31]	-10.280	5.056	-20.470	-2.682	1.239	0.671	15.0	46.0	1.19
intercept	97.958	11.821	71.555	113.816	3.539	2.009	9.0	36.0	1.37
sigma	10.931	0.063	10.826	11.020	0.005	0.011	324.0	172.0	1.01

Рис. 6. Зведена таблиця параметрів моделі після семплювання

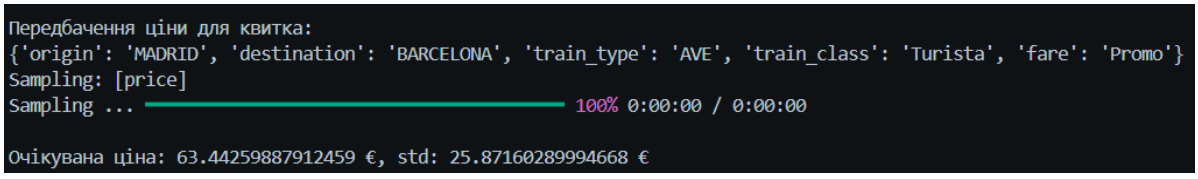

Рис. 7. Прогнозована ціна для тестового квитка

У завданні було використано багатовимірну лінійну регресію у байєсівському підході за допомогою бібліотеки PyMC. Вхідними ознаками є origin,

destination, train_type, train_class, fare, цільовою є price. Виконано семплінг методом NUTS (для тестування було використано 200 семплів і короткий прогрів 100, для отримання кращих результатів необхідно збільшити кількість семплів та прогрів). Було створено новий квиток та спрогнозовано його ціну.

Висновок: в ході виконання лабораторної роботи ми набули навички працювати з даними і опанували роботу у Python з використанням теореми Байєса.

Репозиторій: https://github.com/SofiiaKozlyk/AI-systems