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.

4, 9, 14	Outlook = Sunny Humidity = Normal Wind = Strong	Перспектива = Сонячно Вологість = Нормальна Вітер = Сильний

Маємо наступну таблицю з даними (рис. 1):

play_tennis.csv

1 to 14 of 14 entries Filter

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

Show 50 per page

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) = 1/5

- P(Wind= Weak|No) = 3/5

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 * 1/5 * 3/5 * 5/14 ≈ 0,0257

Тепер, коли ми нормалізуємо значення, ми отримуємо:

P(Yes) = 0,0317 / (0,0317 + 0,0574) ≈ 55.2%

P(No) = 0,0257 / (0,0317 + 0,0574) ≈ 44.8%

Модель передбачає, що ймовірність 55.2%, що завтра буде гра. Для виконання програми дані (рис. 1) було записано у файл 'play_tennis.csv.

Лістинг програми:

```
import pandas as pd

df = pd.read_csv('play_tennis.csv')

cnt = df['Day'].count()

p_yes = len(df[df['Play'] == 'Yes']) / cnt
p_no = len(df[df['Play'] == 'No']) / cnt

print(f"P(Yes): {p_yes}")
print(f"P(No): {p_no}")

P_Sunny_Yes = len(df[(df['Outlook'] == 'Sunny') & (df['Play'] == 'Yes')]) /
len(df[df['Play'] == 'Yes'])
P_Sunny_No = len(df[(df['Outlook'] == 'Sunny') & (df['Play'] == 'No')]) /
len(df[df['Play'] == 'No'])

P_Normal_Yes = len(df[(df['Humidity'] == 'Normal') & (df['Play'] == 'Yes')]) /
len(df[df['Play'] == 'Yes'])
P_Normal_No = len(df[(df['Humidity'] == 'Normal') & (df['Play'] == 'No')]) /
len(df[df['Play'] == 'No'])
```

```
P_Strong_Yes = len(df[(df['Wind'] == 'Strong') & (df['Play'] == 'Yes')]) /
len(df[df['Play'] == 'Yes'])
P_Strong_No = len(df[(df['Wind'] == 'Strong') & (df['Play'] == 'No')]) /
len(df[df['Play'] == 'No'])

P_yes_final = P_Sunny_Yes * P_Normal_Yes * P_Strong_Yes * p_yes
P_no_final = P_Sunny_No * P_Normal_No * P_Strong_No * p_no

print(f"\nЙмовірність гри (Yes): {P_yes_final:.4f}")
print(f"Ймовірність відсутності гри (No): {P_no_final:.4f}")

if P_yes_final > P_no_final:
    total_score = P_yes_final + P_no_final

real_percent_yes = P_yes_final / total_score
real_percent_no = P_no_final / total_score

print(f"\n--- Нормалізовані ймовірності ---")
print(f"Ймовірність 'Yes': {real_percent_yes:.4f} (або
{real_percent_yes*100:.1f}%)")
print(f"Ймовірність 'No':  {real_percent_no:.4f} (або
{real_percent_no*100:.1f}%)")

if real_percent_yes > real_percent_no:
    print("Результат: Матч відбудеться")
else:
    print("Результат: Матч не відбудеться")
```

Результат виконаня програми:

```
Ймовірність гри (Yes): 0.0317
Ймовірність відсутності гри (No): 0.0257

--- Нормалізовані ймовірності ---
Ймовірність 'Yes': 0.5525 (або 55.2%)
Ймовірність 'No':  0.4475 (або 44.8%)
Результат: Матч відбудеться
```

Рис. 2

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')

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

    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()

    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)

        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))

    ticket_data = {
        "origin": "MADRID",
        "destination": "BARCELONA",
        "train_type": "AVE",
        "train_class": "Turista",
        "fare": "Promo"
    }

    new_ticket = pd.DataFrame(0, index=[0], columns=df_encoded.columns)

    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)

    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

    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:

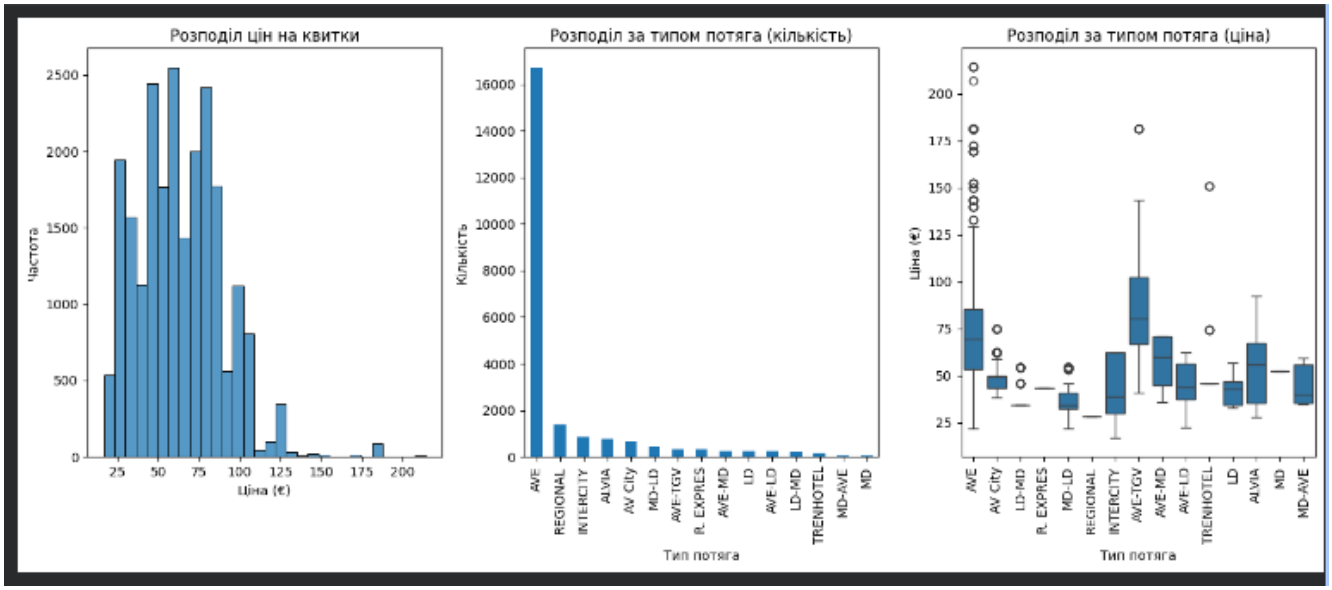

Рис. 3. Графіки з розподілами квитків.

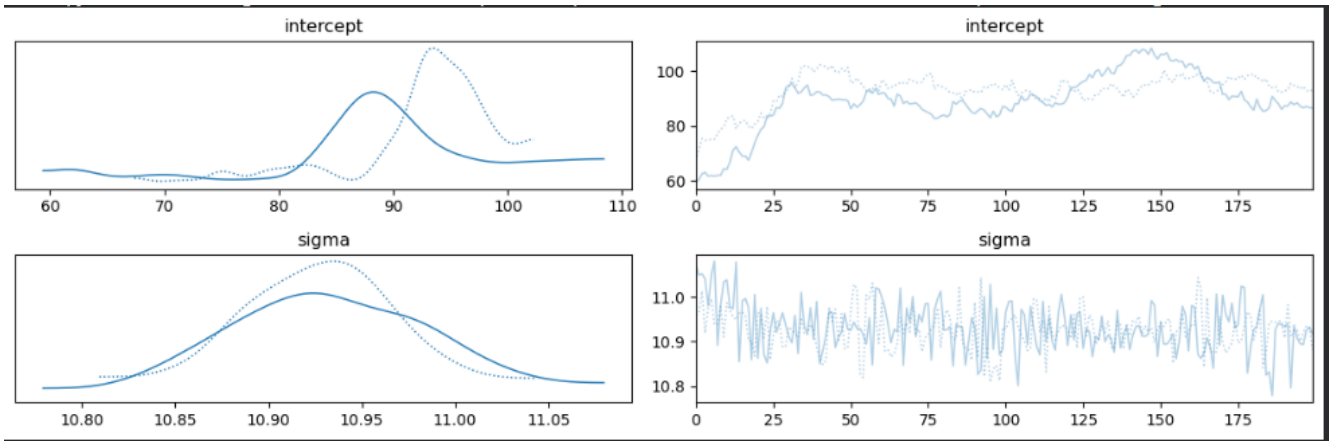

Рис. 4. Графіки вибірки MCMC і апостеріорного розподілу.

Progress	Draws	Divergences	Step size	Grad evals	Sampling Speed	Elapsed	Remaining
———	300	0	0.006	1023	2.79 s/draws	0:13:53	0:00:00
———	300	0	0.005	1023	5.67 s/draws	0:28:15	0:00:00

Рис. 5.

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	\
beta[0]	17.587	4.961	9.465	27.141	1.825	0.554	8.0	
beta[1]	-32.370	0.772	-33.628	-30.841	0.037	0.030	423.0	
beta[2]	-29.089	0.273	-29.593	-28.628	0.013	0.011	435.0	
beta[3]	-37.403	0.294	-37.932	-36.851	0.015	0.012	385.0	
beta[4]	17.258	4.956	8.084	25.911	1.825	0.560	8.0	
beta[5]	-36.265	0.634	-37.391	-35.090	0.024	0.029	686.0	
beta[6]	-29.712	0.282	-30.204	-29.173	0.013	0.013	484.0	
beta[7]	-38.094	0.277	-38.602	-37.633	0.013	0.009	441.0	
beta[8]	6.676	0.671	5.398	7.850	0.032	0.030	446.0	
beta[9]	12.377	0.488	11.509	13.264	0.023	0.024	477.0	
beta[10]	5.395	2.391	1.090	10.216	0.604	0.299	16.0	
beta[11]	6.883	2.349	2.435	11.321	0.611	0.305	15.0	
beta[12]	9.202	0.782	7.734	10.510	0.032	0.032	615.0	
beta[13]	-2.558	0.615	-3.738	-1.474	0.026	0.026	554.0	
beta[14]	-3.249	2.351	-7.967	0.506	0.584	0.286	17.0	
beta[15]	-8.309	2.378	-12.644	-3.789	0.620	0.304	16.0	
beta[16]	-12.794	2.975	-18.429	-7.003	0.665	0.429	21.0	
beta[17]	-4.322	2.521	-9.183	-0.164	0.600	0.247	18.0	
beta[18]	-7.820	2.285	-12.128	-3.893	0.590	0.284	16.0	
beta[19]	-46.477	4.595	-54.510	-40.154	1.035	1.777	19.0	
beta[20]	-23.801	4.595	-32.731	-16.242	1.043	1.765	18.0	
beta[21]	-16.655	1.170	-19.133	-14.739	0.113	0.106	125.0	
beta[22]	9.616	4.379	2.756	18.120	1.611	0.323	7.0	
beta[23]	5.063	4.403	-2.587	13.002	1.892	0.572	5.0	
beta[24]	-18.566	4.370	-26.572	-10.909	1.873	0.561	5.0	
beta[25]	-8.971	4.382	-17.217	-1.359	1.886	0.561	5.0	
beta[26]	-20.743	4.592	-29.299	-13.068	1.888	0.469	6.0	
beta[27]	7.348	4.455	-0.805	14.740	1.017	1.710	18.0	
beta[28]	61.456	10.219	44.242	75.109	1.981	3.677	27.0	
beta[29]	54.027	7.990	42.301	65.553	2.112	2.701	28.0	
beta[30]	-23.441	4.457	-31.553	-15.910	1.019	1.712	18.0	
beta[31]	-12.301	4.484	-20.486	-5.853	1.014	1.666	19.0	
intercept	91.074	8.706	74.697	107.954	2.969	2.110	11.0	
sigma	10.930	0.051	10.826	11.021	0.004	0.005	154.0	

Рис. 6. Зведена таблиця параметрів моделі після семплювання.

```
Передбачення ціни для квитка:
{'origin': 'MADRID', 'destination': 'BARCELONA', 'train_type': 'AVE', 'train_class': 'Turista', 'fare': 'Promo'}
Sampling ... ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00 / 0:00:01

Очікувана ціна: 63.44011037717469 €, std: 25.87993393029359 €
```

Рис. 7. Прогнозована ціна.

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/Vanchik21/AI