Лабораторна робота 6

Наївний Байєс в Python

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

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

Варіант 14

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

```
=== Частотні таблиці (Frequency Tables) ===

--- Outlook ---
Play      No  Yes
Outlook
Overcast   0    4
Rain       2    3
Sunny      3    2

--- Humidity ---
Play      No  Yes
Humidity
High       4    3
Normal     1    6

--- Wind ---
Play    No  Yes
Wind
Strong   3    3
Weak     2    6

=== Апріорні ймовірності ===
P(Yes) = 0.643
P(No)  = 0.357
```

Рис. 1 Розрахунок частотних та апріорних ймовірностей

					ДУ «Житомирська політехніка».25.121.14.003 – Лр6		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Кольцова Н.О.			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів 9
Перевір.		Маєвський О.В.					
Керівник						ФІКТ Гр. ІПЗ-22-4[1]	
Н. контр.							
Зав. каф.							

```
=== Таблиці правдоподібності (Likelihood Tables) ===

Outlook:
          P(Outlook=value | Yes)  P(Outlook=value | No)
Outlook
Overcast                   0.444                    0.0
Rain                       0.333                    0.4
Sunny                      0.222                    0.6

Humidity:
          P(Humidity=value | Yes)  P(Humidity=value | No)
Humidity
High                        0.333                     0.8
Normal                      0.667                     0.2

Wind:
         P(Wind=value | Yes)  P(Wind=value | No)
Wind
Strong                0.333                0.6
Weak                  0.667                0.4

=== Вхідні умови ===
Outlook = Sunny
Humidity = High
Wind = Weak

=== Результат (Posterior Probabilities) ===
P(Yes | x) = 0.3161
P(No  | x) = 0.6839

=== Висновок ===
Матч НЕ відбудеться (No)
```

Рис. 2 Таблиці правдоподібностей та результати класифікації

Висновок: На основі частотних таблиць, апріорних ймовірностей та розрахованих таблиць правдоподібності було побудовано модель наївного байєсівського класифікатора для прогнозування можливості проведення матчу. Аналіз показує, що деякі ознаки мають значний вплив на кінцеве рішення. Зокрема, сонячна погода частіше пов’язана з відсутністю гри, оскільки в наборі даних значення Outlook = Sunny має більше негативних випадків, ніж позитивних. Висока вологість також є несприятливою умовою, оскільки вона частіше зустрічається у прикладах класу “No”. Слабкий вітер, навпаки, дещо більше асоціюється з проведенням гри, однак його вплив недостатній, щоб змінити загальний результат.

Після обчислення апостеріорних ймовірностей для заданих умов (Sunny, High, Weak) отримано P(Yes|x) = 0.3161 та P(No|x) = 0.6839. Це означає, що ймовірність того, що матч не відбудеться, суттєво перевищує ймовірність проведення зустрічі. Таким чином, відповідно до моделі наївного Байєса, за заданих погодних умов матч, найімовірніше, не відбудеться.

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

```
Початкові дані:
           insert_date  origin destination           start_date             end_date train_type  price train_class      fare
0  2019-04-22 08:00:25  MADRID     SEVILLA  2019-04-28 08:30:00  2019-04-28 11:14:00      ALVIA    NaN     Turista  Flexible
1  2019-04-22 10:03:24  MADRID    VALENCIA  2019-05-20 06:45:00  2019-05-20 08:38:00        AVE  21.95     Turista     Promo
2  2019-04-25 19:19:46  MADRID     SEVILLA  2019-05-29 06:20:00  2019-05-29 09:16:00    AV City  38.55     Turista     Promo

=== GaussianNB: Матриця плутанини ===
[[ 651  857    0]
 [ 379 2529  816]
 [   1  110 1472]]

=== GaussianNB: Звіт класифікації ===
              precision    recall  f1-score   support

       Cheap       0.63      0.43      0.51      1508
      Medium       0.72      0.68      0.70      3724
   Expensive       0.64      0.93      0.76      1583

    accuracy                           0.68      6815
   macro avg       0.67      0.68      0.66      6815
weighted avg       0.68      0.68      0.67      6815


=== MultinomialNB: Матриця плутанини ===
[[1409   68    0]
 [ 398 2528  814]
 [   0   62 1536]]

=== MultinomialNB: Звіт класифікації ===
              precision    recall  f1-score   support

       Cheap       0.78      0.95      0.86      1477
      Medium       0.95      0.68      0.79      3740
   Expensive       0.65      0.96      0.78      1598

    accuracy                           0.80      6815
   macro avg       0.79      0.86      0.81      6815
weighted avg       0.84      0.80      0.80      6815
```

Рис. 3 Межі класифікації Random Forest на тестових даних

Висновок: Використовуючи дані про ціни на квитки на іспанські високошвидкісні залізниці, було побудовано два наївні байєсовські класифікатори — GaussianNB та MultinomialNB. Для категоризації цін у три класи (дешеві, середні, дорогі) обчислено матриці плутанини та звіти класифікації. Результати показують, що MultinomialNB забезпечує кращу точність класифікації (accuracy = 0.80)

порівняно з GaussianNB (accuracy = 0.68). Основний внесок у точність моделі MultinomialNB зробили категоріальні ознаки (початкова та кінцева станції, тип поїзда, клас обслуговування та тариф), які кодуються за допомогою one-hot енкодера. Модель добре відрізняє дешеві та дорогі квитки, з високим recall для класу Expensive, а також демонструє задовільну здатність класифікувати середні ціни. Результати свідчать, що наївний байєсовський підхід ефективний для прогнозування категорій цін, особливо коли ознаки є дискретними або закодованими як частотні категорії.

Висновки: в ході лабораторної роботи було набуто навичи працювання з даними і опонувати роботу у Python з використанням теореми Байэса.

Посилання на github: https://github.com/KoltcovaNadiia/Artificial-intelligence-systems-2025

Лістинг програми:

LR_6_task_1.py

```
import pandas as pd

# 1. Вхідні дані — набір Play Tennis
data = [
    {"Outlook": "Sunny",    "Humidity": "High",    "Wind": "Weak",   "Play": "No"},
    {"Outlook": "Sunny",    "Humidity": "High",    "Wind": "Strong", "Play": "No"},
    {"Outlook": "Overcast", "Humidity": "High",    "Wind": "Weak",   "Play": "Yes"},
    {"Outlook": "Rain",     "Humidity": "High",    "Wind": "Weak",   "Play": "Yes"},
    {"Outlook": "Rain",     "Humidity": "Normal",  "Wind": "Weak",   "Play": "Yes"},
    {"Outlook": "Rain",     "Humidity": "Normal",  "Wind": "Strong", "Play": "No"},
    {"Outlook": "Overcast", "Humidity": "Normal",  "Wind": "Strong", "Play": "Yes"},
    {"Outlook": "Sunny",    "Humidity": "High",    "Wind": "Weak",   "Play": "No"},
    {"Outlook": "Sunny",    "Humidity": "Normal",  "Wind": "Weak",   "Play": "Yes"},
    {"Outlook": "Rain",     "Humidity": "Normal",  "Wind": "Weak",   "Play": "Yes"},
    {"Outlook": "Sunny",    "Humidity": "Normal",  "Wind": "Strong", "Play": "Yes"},
    {"Outlook": "Overcast", "Humidity": "High",    "Wind": "Strong", "Play": "Yes"},
    {"Outlook": "Overcast", "Humidity": "Normal",  "Wind": "Weak",   "Play": "Yes"},
    {"Outlook": "Rain",     "Humidity": "High",    "Wind": "Strong", "Play": "No"}
]

df = pd.DataFrame(data)

# 2. Частотні таблиці
print("\n=== Частотні таблиці (Frequency Tables) ===\n")

for feature in ["Outlook", "Humidity", "Wind"]:
    print(f"--- {feature} ---")
    print(pd.crosstab(df[feature], df["Play"]), "\n")

# 3. Апріорні ймовірності класів
p_yes = (df["Play"] == "Yes").mean()
p_no = (df["Play"] == "No").mean()

print("=== Апріорні ймовірності ===")
print(f"P(Yes) = {p_yes:.3f}")
print(f"P(No)  = {p_no:.3f}\n")

# 4. Функція побудови таблиць правдоподібності
def likelihood_table(feature):
    freq = pd.crosstab(df[feature], df["Play"])
    p_given_yes = (freq["Yes"] / freq["Yes"].sum()).round(3)
    p_given_no = (freq["No"] / freq["No"].sum()).round(3)

    return pd.DataFrame({
        f"P({feature}=value | Yes)": p_given_yes,
        f"P({feature}=value | No)": p_given_no
    })
```

```
# Таблиці правдоподібності
print("=== Таблиці правдоподібності (Likelihood Tables) ===\n")

outlook_lh = likelihood_table("Outlook")
humidity_lh = likelihood_table("Humidity")
wind_lh = likelihood_table("Wind")

print("Outlook:\n", outlook_lh, "\n")
print("Humidity:\n", humidity_lh, "\n")
print("Wind:\n", wind_lh, "\n")

# 5. Умови, для яких треба визначити результат
X_outlook = "Sunny"
X_humidity = "High"
X_wind = "Weak"

print("=== Вхідні умови ===")
print(f"Outlook = {X_outlook}")
print(f"Humidity = {X_humidity}")
print(f"Wind = {X_wind}\n")

# 6. Вибір ймовірностей P(feature=value | class)
p1_yes = outlook_lh.loc[X_outlook, "P(Outlook=value | Yes)"]
p1_no  = outlook_lh.loc[X_outlook, "P(Outlook=value | No)"]

p2_yes = humidity_lh.loc[X_humidity, "P(Humidity=value | Yes)"]
p2_no  = humidity_lh.loc[X_humidity, "P(Humidity=value | No)"]

p3_yes = wind_lh.loc[X_wind, "P(Wind=value | Yes)"]
p3_no  = wind_lh.loc[X_wind, "P(Wind=value | No)"]

# 7. Байєсівське обчислення
unnorm_yes = p1_yes * p2_yes * p3_yes * p_yes
unnorm_no  = p1_no  * p2_no  * p3_no  * p_no

evidence = unnorm_yes + unnorm_no

P_yes_x = unnorm_yes / evidence
P_no_x = unnorm_no / evidence

print("=== Результат (Posterior Probabilities) ===")
print(f"P(Yes | x) = {P_yes_x:.4f}")
print(f"P(No  | x) = {P_no_x:.4f}\n")

# 8. Висновок
print("=== Висновок ===")
if P_yes_x > P_no_x:
    print("Матч ВІДБУДЕТЬСЯ (Yes)")
else:
```

```
    print("Матч НЕ відбудеться (No)")
```

LR_6_task_2.py

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler, LabelEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import GaussianNB, MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix

# ------------------------------
# 1. Завантаження даних
# ------------------------------
url = "https://raw.githubusercontent.com/susanli2016/Machine-Learning-with-
Python/master/data/renfe_small.csv"
df = pd.read_csv(url)
print("Початкові дані:")
print(df.head(3))

# ------------------------------
# 2. Обробка дат та створення ознак
# ------------------------------
df['start_date'] = pd.to_datetime(df['start_date'])
df['end_date'] = pd.to_datetime(df['end_date'])
df['trip_duration'] = (df['end_date'] - df['start_date']).dt.total_seconds() / 3600
df['month'] = df['start_date'].dt.month
df['weekday'] = df['start_date'].dt.weekday

# Видаляємо непотрібні колонки
df = df.drop(columns=['insert_date', 'start_date', 'end_date'])

# Видаляємо рядки з пропущеними критичними значеннями
df = df.dropna(subset=['price', 'train_class', 'fare'])

# ------------------------------
# 3. Кодування категоріальних змінних
# ------------------------------
label_cols = ['origin', 'destination', 'train_type', 'train_class', 'fare']
label_encoders = {}
for col in label_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# ------------------------------
# 4. Категоризація цін
# ------------------------------
```

```python
def categorize_price(price):
    if price < 40:
        return 0  # cheap
    elif price < 80:
        return 1  # medium
    else:
        return 2  # expensive

df['price_category'] = df['price'].apply(categorize_price)

# ------------------------------
# 5. Підготовка ознак та цільової змінної
# ------------------------------
X = df.drop(columns=['price', 'price_category'])
y = df['price_category']

# ------------------------------
# 6. GaussianNB
# ------------------------------
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.3, random_state=42
)

gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred_gnb = gnb.predict(X_test)

print("\n=== GaussianNB: Матриця плутанини ===")
print(confusion_matrix(y_test, y_pred_gnb))
print("\n=== GaussianNB: Звіт класифікації ===")
print(classification_report(y_test, y_pred_gnb, target_names=['Cheap', 'Medium',
'Expensive']))

# ------------------------------
# 7. MultinomialNB з OneHotEncoder
# ------------------------------
numeric_cols = ['trip_duration', 'month', 'weekday']
cat_cols = ['origin', 'destination', 'train_type', 'train_class', 'fare']

ct = ColumnTransformer([
    ("onehot", OneHotEncoder(handle_unknown='ignore', sparse_output=False),
numeric_cols + cat_cols)
])

pipe = Pipeline([
    ("prep", ct),
    ("clf", MultinomialNB())
```

```
])

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)

pipe.fit(X_train, y_train)
y_pred_mnb = pipe.predict(X_test)

print("\n=== MultinomialNB: Матриця плутанини ===")
print(confusion_matrix(y_test, y_pred_mnb))
print("\n=== MultinomialNB: Звіт класифікації ===")
print(classification_report(y_test, y_pred_mnb, target_names=['Cheap', 'Medium',
'Expensive'], zero_division=0))
```