ЛАБОРАТОРНА РОБОТА № 6
Наївний Байєс в Python
Варіант 13
Хід роботи:

Завдання 1: Ретельно опрацювати теоретичні відомості.

Виконання:

1. Типи наївного байєвського класифікатора:
   a. Гаусівський
   b. Бернулійський
   c. Поліноміальний
2. Де використовується:
   a. Фільтрація спаму за допомогою визначення частоти слів
   b. Визначення тону тексту
   c. Класифікація документів
   d. Створення систем рекомендацій
   e. Діагноз медіа

					ДУ «Житомирська політехніка».21.121.5.000 - Лр1		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Корнійчук В. В.			Звіт з лабораторної роботи	Літ.	Арк. 1 / Аркушів
Перевір.		Іванов Д. А.					
Керівник						ФІКТ Гр. ІПЗ-21-5[2]	
Н. контр.							
Зав. каф.							

Завдання 2: Використовуючи дані, визначити відбудеться матч при наступних погодних умовах чи ні.

Дані:

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

Варіант:

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

Код програми:

```
import pandas as pd


def get_match_likelihood(dataframe, key, value):
    yes = len(
        dataframe[(dataframe[key] == value) & (dataframe["Play"] == "Yes")]
    ) / len(dataframe[dataframe["Play"] == "Yes"])
    no = len(dataframe[(dataframe[key] == value) & (dataframe["Play"] == "No")]) /
len(
        dataframe[dataframe["Play"] == "No"]
    )
    return yes, no


def get_outlook_match_likelihood(dataframe, value):
    return get_match_likelihood(dataframe, "Outlook", value)


def get_humidity_match_likelihood(dataframe, value):
    return get_match_likelihood(dataframe, "Humidity", value)


def get_wind_match_likelihood(dataframe, value):
    return get_match_likelihood(dataframe, "Wind", value)


def get_probability(*conditions):
    result = 1
    for value in conditions:
        result *= value
```

```
    return result


def normalize(yes_probability, no_probability):
    total = yes_probability + no_probability
    yes = yes_probability / total
    no = no_probability / total

    return yes, no


data = pd.DataFrame(
    {
        "Day": [
            "D1",
            "D2",
            "D3",
            "D4",
            "D5",
            "D6",
            "D7",
            "D8",
            "D9",
            "D10",
            "D11",
            "D12",
            "D13",
            "D14",
        ],
        "Outlook": [
            "Sunny",
            "Sunny",
            "Overcast",
            "Rain",
            "Rain",
            "Rain",
            "Overcast",
            "Sunny",
            "Sunny",
            "Rain",
            "Sunny",
            "Overcast",
            "Overcast",
            "Rain",
        ],
        "Humidity": [
            "High",
            "High",
            "High",
            "High",
```

```
            "Normal",
            "Normal",
            "Normal",
            "High",
            "Normal",
            "Normal",
            "Normal",
            "High",
            "Normal",
            "High",
        ],
        "Wind": [
            "Weak",
            "Strong",
            "Weak",
            "Weak",
            "Weak",
            "Strong",
            "Strong",
            "Weak",
            "Weak",
            "Weak",
            "Strong",
            "Strong",
            "Weak",
            "Strong",
        ],
        "Play": [
            "No",
            "No",
            "Yes",
            "Yes",
            "Yes",
            "No",
            "Yes",
            "No",
            "Yes",
            "Yes",
            "Yes",
            "Yes",
            "Yes",
            "No",
        ],
    }
)

outlook = "Sunny"
humidity = "High"
wind = "Weak"
```

```
yes_outlook_likelihood, no_outlook_likelihood = get_outlook_match_likelihood(
    data,
    outlook,
)
yes_humidity_likelihood, no_humidity_likelihood = get_humidity_match_likelihood(
    data,
    humidity,
)
yes_wind_likelihood, no_wind_likelihood = get_wind_match_likelihood(
    data,
    wind,
)

yes_probability = get_probability(
    yes_outlook_likelihood,
    yes_humidity_likelihood,
    yes_wind_likelihood,
)
no_probability = get_probability(
    no_outlook_likelihood,
    no_humidity_likelihood,
    no_wind_likelihood,
)

yes, no = normalize(yes_probability, no_probability)
print(f"Match WILL happen: {yes:.2f};\nMatch will NOT happen: {no:.2f};")
```

Виконання:

```
Match WILL happen: 0.20;
Match will NOT happen: 0.80;
```

Рисунок 2.1 – Виконання програми

Завдання 3: Застосуєте методи байєсівського аналізу до набору даних про ціни на квитки на іспанські високошвидкісні залізниці.

Код програми:

```
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Читаємо файл
url = "https://raw.githubusercontent.com/susanli2016/Machine-Learning-with-
Python/master/data/renfe_small.csv"
data = pd.read_csv(url)

# Викидуємо пропущені значення
data = data.dropna(subset=["price"])

# Категоризуємо дані для роботи з класифікатором
data["price_category"] = pd.cut(
    data["price"],
    bins=[0, 30, 60, 100, float("inf")],
    labels=["low", "medium", "high", "very_high"],
)

# Викидуємо не категоризовані дані
data.drop(columns=["price", "insert_date", "start_date", "end_date"], inplace=True)

# Перетворення рядкових даних на числові
label_encoders = {}
for column in [
    "price_category",
    "origin",
    "destination",
    "train_type",
    "train_class",
    "fare",
]:
    instance = LabelEncoder()
    data[column] = instance.fit_transform(data[column])
    label_encoders[column] = instance

# Визначаємо ознаки та мітки
X = data.drop("price_category", axis=1)
y = data["price_category"]

# Створення даних для навчання і тестування
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.1, random_state=42
)

# Навчаємо класифікатор
```

```python
classifier = GaussianNB()
classifier.fit(X_train, y_train)

# Прогнозування на одиночному прикладі
datapoint = X_test.iloc[0]

decoded_datapoint = datapoint.copy()
for column in label_encoders:
    if column == "price_category":
        continue

    decoded_datapoint[column] = label_encoders[column].inverse_transform(
        [datapoint[column]]
    )[0]

print(f"Datapoint:\n{decoded_datapoint}")

prediction = classifier.predict([datapoint])
decoded_prediction = label_encoders["price_category"].inverse_transform(prediction)
print("\nPrice prediction:", decoded_prediction[0])
```

Виконання:

```
Datapoint:
origin         VALENCIA
destination      MADRID
train_type           AVE
train_class      Turista
fare               Promo
Name: 17858, dtype: object

Price prediction: high
```

Рисунок 3.1 – Виконання програми

Посилання на GitHub: https://github.com/ipz215kvv/artificial-inteligence-systems