Лабораторна робота №6

Наївний Байєс в Python

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

Література

Supervised learning - https://scikit-learn.org/stable/user_guide.html
Naive Bayes Tutorial: Naive Bayes Classifier in Python - https://dzone.com/articles/naive-bayes-tutorial-naive-bayes-classifier-in-pyt
Наивный баейсовский классификатор - http://datascientist.one/naive-bayes/

Завдання 1. Ретельно опрацювати теоретичні відомості:

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

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

Розберемо приклад, будемо використовувати код з завдання 3(дані з прикладу було трохи відредаговано для більшої точності прогнозувань):

В прикладі маємо показники: Дощ, слабкий вітер та високу вологість.

```
Conditions: {'Outlook': 'rainy', 'Humidity': 'high', 'Wind': 'weak'}
Probability that the match will happen (Yes): 31.65%
Probability that the match will not happen (No): 68.35%
```

Ось що каже програма яку я написала.

Показники відрізняються від показаних в прикладі за рахунок відмінностей в табличці:

```
data = {
    "Outlook": {
        "rainy": {"Yes": 2, "No": 3},
        "sunny": {"Yes": 3, "No": 2},
        "overcast": {"Yes": 4, "No": 0}
    },
    "Humidity": {
        "high": {"Yes": 3, "No": 4},
        "normal": {"Yes": 6, "No": 1}
    },
    "Wind": {
        "weak": {"Yes": 6, "No": 2},
        "strong": {"Yes": 3, "No": 3}
    }
}
```

Але якщо їх обрахувати вручну то результат буде таким самим як видає програма, можна зробити висновок що вона працює вірно.

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

2, 7, 12	Outlook = Overcast Humidity = High Wind = Strong	Перспектива = Похмуро Вологість = Висока Вітер = Сильний

Результат виконання:

```
[Running] python -u "d:\work\ksushenkaAI\Lab6\Task3.py"
Conditions: {'Outlook': 'overcast', 'Humidity': 'high', 'Wind': 'strong'}
Probability that the match will happen (Yes): 100.00%
Probability that the match will not happen (No): 0.00%
```

Лістинг:

```
def calculate_probability(data, total_yes, total_no, conditions):
    total = total_yes + total_no
    p_yes = total_yes / total
```

```
    p_no = total_no / total

    def get_conditional_probability(feature, value, outcome):
        try:
            return data[feature][value][outcome] / (total_yes if outcome == "Yes"
else total_no)
        except KeyError:
            raise ValueError(f"Invalid value '{value}' for feature '{feature}'.")

    p_rain_yes = get_conditional_probability("Outlook", conditions["Outlook"],
"Yes")
    p_rain_no = get_conditional_probability("Outlook", conditions["Outlook"], "No")

    p_humidity_yes = get_conditional_probability("Humidity",
conditions["Humidity"], "Yes")
    p_humidity_no = get_conditional_probability("Humidity", conditions["Humidity"],
"No")

    p_wind_yes = get_conditional_probability("Wind", conditions["Wind"], "Yes")
    p_wind_no = get_conditional_probability("Wind", conditions["Wind"], "No")

    p_yes_given_conditions = p_rain_yes * p_humidity_yes * p_wind_yes * p_yes
    p_no_given_conditions = p_rain_no * p_humidity_no * p_wind_no * p_no

    total_probability = p_yes_given_conditions + p_no_given_conditions
    p_yes_final = p_yes_given_conditions / total_probability
    p_no_final = p_no_given_conditions / total_probability

    return p_yes_final, p_no_final

data = {
    "Outlook": {
        "rainy": {"Yes": 2, "No": 3},
        "sunny": {"Yes": 3, "No": 2},
        "overcast": {"Yes": 4, "No": 0}
    },
    "Humidity": {
        "high": {"Yes": 3, "No": 4},
        "normal": {"Yes": 6, "No": 1}
    },
    "Wind": {
        "weak": {"Yes": 6, "No": 2},
        "strong": {"Yes": 3, "No": 3}
    }
}

total_yes = 9
```

```
total_no = 5

test_conditions = {
    "Outlook": "overcast",
    "Humidity": "high",
    "Wind": "strong"
}

try:
    p_yes_final, p_no_final = calculate_probability(data, total_yes, total_no,
test_conditions)
    print(f"Conditions: {test_conditions}")
    print(f"Probability that the match will happen (Yes): {p_yes_final:.2%}")
    print(f"Probability that the match will not happen (No): {p_no_final:.2%}")
except ValueError as e:
    print(f"Error: {e}")
```

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

– Вхідні дані: https://raw.githubusercontent.com/susanli2016/Machine-Learning-with-Python/master/data/renfe_small.csv

```
Train Price Prediction System (Enhanced)
----------------------------------------
Available Train Type Options: ALVIA, AV CITY, AVE, AVE-LD, AVE-MD, AVE-TGV, INTERCITY, LD, LD-MD, MD, MD-AVE, MD-LD, R.
 EXPRES, REGIONAL, TRENHOTEL
Enter Train Type: ave
Available Train Class Options: CAMA G. CLASE, CAMA TURISTA, PREFERENTE, TURISTA, TURISTA CON ENLACE, TURISTA PLUS
Enter Train Class: turista
Available Fare Options: ADULTO IDA, FLEXIBLE, INDIVIDUAL-FLEXIBLE, MESA, PROMO, PROMO +
Enter Fare Type: promo
Enter Origin Station: Madrid
Enter Destination Station: sevilla
Enter Start Date (YYYY-MM-DD): 2023-10-11

Processing your inputs...
The estimated price for your trip is: €53.06
```

Лістинг:

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import BayesianRidge
from sklearn.metrics import mean_squared_error
from datetime import datetime

data = pd.read_csv('data.txt')
data.dropna(subset=['price'], inplace=True)
data['start_date'] = pd.to_datetime(data['start_date'])
```

```
data['day_of_week'] = data['start_date'].dt.dayofweek

encoders = {}
for column in ['origin', 'destination', 'train_type', 'train_class', 'fare']:
    encoder = LabelEncoder()
    data[f'{column}_enc'] = encoder.fit_transform(data[column].str.lower())
    encoders[column] = encoder

X = data[['origin_enc', 'destination_enc', 'train_type_enc', 'train_class_enc',
'fare_enc', 'day_of_week']]
y = data['price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

model = BayesianRidge()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Test Set Mean Squared Error: {mse:.2f}")

def display_options(column_name):
    unique_values = sorted(data[column_name].str.upper().unique())
    print(f"Available {column_name.replace('_', ' ').title()} Options: {',
'.join(unique_values)}")

def predict_price(origin, destination, train_type, train_class, fare, start_date):
    try:
        day_of_week = pd.to_datetime(start_date).dayofweek
        input_data = pd.DataFrame([[
            encoders['origin'].transform([origin.lower()])[0],
            encoders['destination'].transform([destination.lower()])[0],
            encoders['train_type'].transform([train_type.lower()])[0],
            encoders['train_class'].transform([train_class.lower()])[0],
            encoders['fare'].transform([fare.lower()])[0],
            day_of_week
        ]], columns=['origin_enc', 'destination_enc', 'train_type_enc',
'train_class_enc', 'fare_enc', 'day_of_week'])
        predicted_price = model.predict(input_data)[0]
        return round(predicted_price, 2)
    except (KeyError, ValueError) as e:
        return f"Invalid input: {e}"

def main():
    print("Train Price Prediction System (Enhanced)")
    print("-" * 40)
```

```
    display_options('train_type')
    train_type = input("Enter Train Type: ")

    display_options('train_class')
    train_class = input("Enter Train Class: ")

    display_options('fare')
    fare = input("Enter Fare Type: ")

    origin = input("Enter Origin Station: ")
    destination = input("Enter Destination Station: ")
    start_date = input("Enter Start Date (YYYY-MM-DD): ")

    print("\nProcessing your inputs...")
    predicted_price = predict_price(origin, destination, train_type, train_class,
fare, start_date)
    if isinstance(predicted_price, str):
        print(f"Error: {predicted_price}")
    else:
        print(f"The estimated price for your trip is: €{predicted_price}")

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

Git: https://github.com/PavlenkoOks/AI