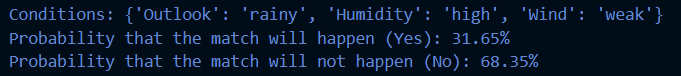


Розберемо приклад, будемо використовувати код з завдання 3(дані з прикладу було трохи відредаговано для більшої точності прогнозувань):

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

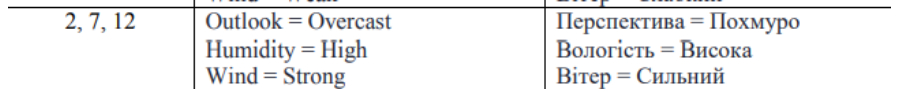
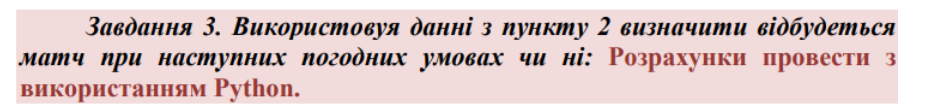


Ось що каже програма яку я написала.

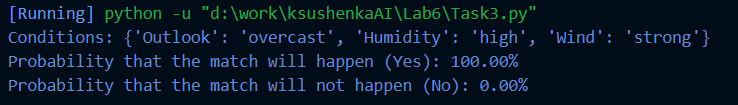
Показники відрізняються від показаних в прикладі за рахунок відмінностей в табличці:



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



Результат виконання:



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)

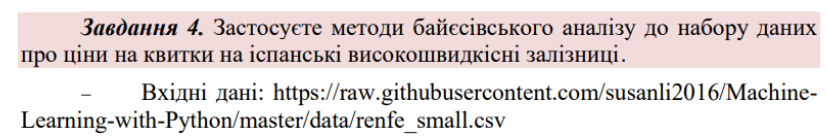
    print(*f*"Conditions: {test\_conditions}")

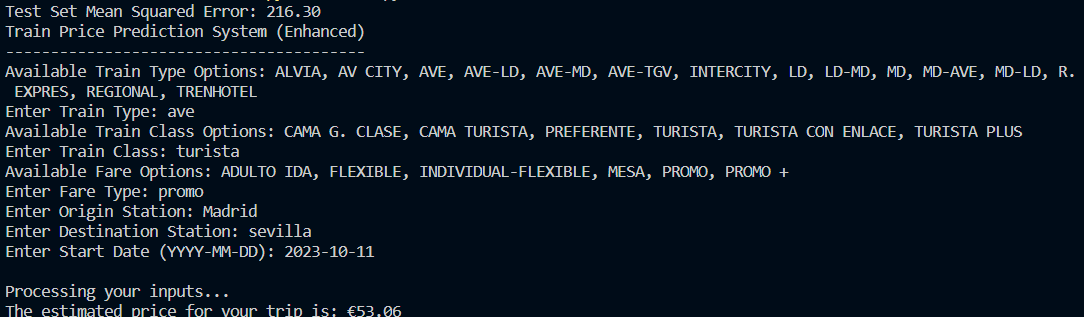
    print(*f*"Probability that the match will happen (Yes): {p\_yes\_final*:.2%*}")

    print(*f*"Probability that the match will not happen (No): {p\_no\_final*:.2%*}")

except ValueError as e:

    print(*f*"Error: {e}")





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>