ЛАБОРАТОРНА РОБОТА № 6
НАЇВНИЙ БАЙЄС В PYTHON

Лукашевич Влада ІПЗ-21-1

https://github.com/vladalukashevych/artificial-intelligence-systems

Завдання №3. Використовуючи дані 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

```
from collections import Counter
import pandas as pd

data = [
    {"Day": "D1", "Outlook": "Sunny", "Humidity": "High", "Wind": "Weak",
"Play": "No"},
    {"Day": "D2", "Outlook": "Sunny", "Humidity": "High", "Wind": "Strong",
"Play": "No"},
    {"Day": "D3", "Outlook": "Overcast", "Humidity": "High", "Wind": "Weak",
"Play": "Yes"},
    {"Day": "D4", "Outlook": "Rain", "Humidity": "High", "Wind": "Weak", "Play":
"Yes"},
    {"Day": "D5", "Outlook": "Rain", "Humidity": "Normal", "Wind": "Weak",
"Play": "Yes"},
    {"Day": "D6", "Outlook": "Rain", "Humidity": "Normal", "Wind": "Strong",
"Play": "No"},
    {"Day": "D7", "Outlook": "Overcast", "Humidity": "Normal", "Wind": "Strong",
"Play": "Yes"},
    {"Day": "D8", "Outlook": "Sunny", "Humidity": "High", "Wind": "Weak",
"Play": "No"},
    {"Day": "D9", "Outlook": "Sunny", "Humidity": "Normal", "Wind": "Weak",
"Play": "Yes"},
    {"Day": "D10", "Outlook": "Rain", "Humidity": "Normal", "Wind": "Weak",
"Play": "Yes"},
    {"Day": "D11", "Outlook": "Sunny", "Humidity": "Normal", "Wind": "Strong",
"Play": "Yes"},
    {"Day": "D12", "Outlook": "Overcast", "Humidity": "High", "Wind": "Strong",
"Play": "Yes"},
    {"Day": "D13", "Outlook": "Overcast", "Humidity": "Normal", "Wind": "Weak",
"Play": "Yes"},
    {"Day": "D14", "Outlook": "Rain", "Humidity": "High", "Wind": "Strong",
"Play": "No"},
]

print(pd.DataFrame(data))


play_yes_count = Counter(row["Play"] for row in data if row["Play"] ==
"Yes").total()
play_count = len(data)
play_yes_prob = play_yes_count / play_count
print("\n\nProbability of the game being played: {0}/{1} = {2}"
      .format(play_yes_count, play_count, round(play_yes_prob, 3)))

overcast_yes_count = Counter(
    row["Outlook"] for row in data if row["Play"] == "Yes" and row["Outlook"] ==
```

```
"Overcast").total()
overcast_yes_prob = overcast_yes_count / play_yes_count
print("Probability of overcast during the game: {0}/{1} = {2}"
      .format(overcast_yes_count, play_yes_count, round(overcast_yes_prob, 3)))

humidity_high_yes_count = Counter(
    row["Humidity"] for row in data if row["Play"] == "Yes" and row["Humidity"]
== "High").total()
humidity_high_yes_prob = humidity_high_yes_count / play_yes_count
print("Probability of high humidity during the game: {0}/{1} = {2}"
      .format(humidity_high_yes_count, play_yes_count,
round(humidity_high_yes_prob, 3)))

wind_strong_yes_count = Counter(
    row["Wind"]for row in data if row["Play"] == "Yes" and row["Wind"] ==
"Strong").total()
wind_strong_yes_prob = wind_strong_yes_count / play_yes_count
print("Probability of strong wind during the game: {0}/{1} = {2}"
      .format(wind_strong_yes_count, play_yes_count, round(wind_strong_yes_prob,
3)))


overall_probability = play_yes_prob * overcast_yes_prob * humidity_high_yes_prob
* wind_strong_yes_prob
print("\nProbability of the game being played with conditions overcast, high
humidity, strong wind:"
      " \n{:.3f} * {:.3f} * {:.3f} * {:.3f} = {:.4f}"
      .format(play_yes_prob, overcast_yes_prob, humidity_high_yes_prob,
wind_strong_yes_prob, overall_probability))
```

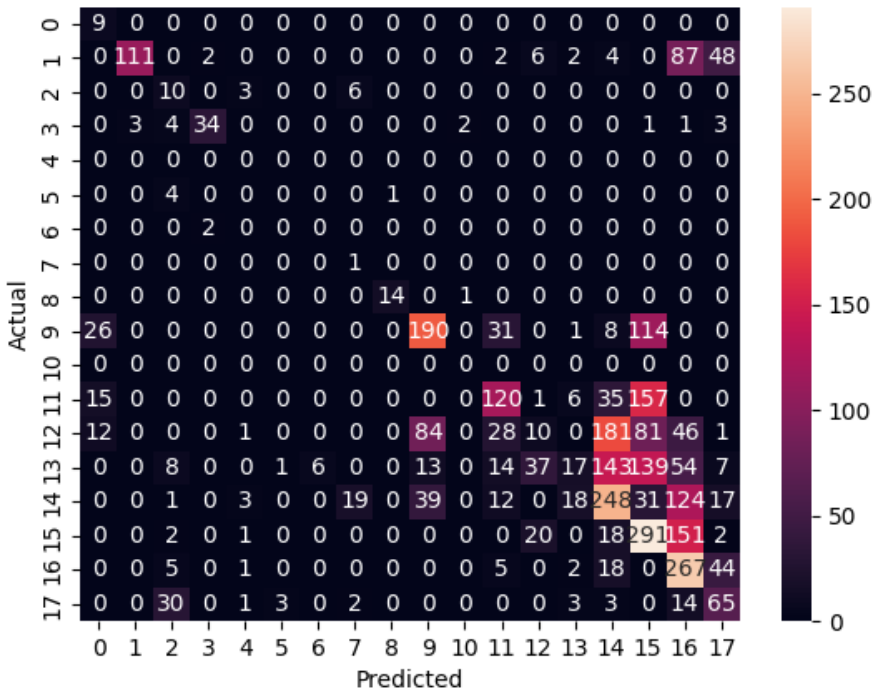
```
    Day   Outlook Humidity    Wind Play
0    D1     Sunny     High    Weak   No
1    D2     Sunny     High  Strong   No
2    D3  Overcast     High    Weak  Yes
3    D4      Rain     High    Weak  Yes
4    D5      Rain   Normal    Weak  Yes
5    D6      Rain   Normal  Strong   No
6    D7  Overcast   Normal  Strong  Yes
7    D8     Sunny     High    Weak   No
8    D9     Sunny   Normal    Weak  Yes
9   D10      Rain   Normal    Weak  Yes
10  D11     Sunny   Normal  Strong  Yes
11  D12  Overcast     High  Strong  Yes
12  D13  Overcast   Normal    Weak  Yes
13  D14      Rain     High  Strong   No


Probability of the game being played: 9/14 = 0.643
Probability of overcast during the game: 4/9 = 0.444
Probability of high humidity during the game: 3/9 = 0.333
Probability of strong wind during the game: 3/9 = 0.333

Probability of the game being played with conditions overcast, high humidity, strong wind:
0.643 * 0.444 * 0.333 * 0.333 = 0.0317
```

Завдання №4. Застосуєте методи байєсівського аналізу до набору даних про ціни на квитки на іспанські високошвидкісні залізниці.

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


df = pd.read_csv('ave_data.csv')
df = df.dropna()


max_val = round(df['price'].max()) + 10
bins = range(0, max_val, 10)
labels = [f"{i}-{i+10}" for i in range(0, max_val-10, 10)]
df['price_range'] = pd.cut(df['price'], bins=bins, right=False, labels=labels)
y = df['price_range']

label_encoders = {}
for column in ['origin', 'destination', 'train_type', 'train_class', 'fare']:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le

df['start_day_of_week'] = pd.to_datetime(df['start_date']).dt.dayofweek
df['duration'] = pd.to_datetime(df['end_date']) -
pd.to_datetime(df['start_date'])
df['duration_minutes'] = (df['duration'].dt.total_seconds()//60).astype(int)

# df['price'] = (df['price'] * 100).astype(int)


X = df[['origin', 'destination', 'start_day_of_week', 'duration_minutes',
'train_type', 'train_class', 'fare']]
# y = df['price']
y = df['price_range']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15,
random_state=5)

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

y_pred = model.predict(X_test)


print("Classification Report:\n", classification_report(y_test, y_pred,
zero_division=np.nan, ))

scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
print("Cross-validation scores:", scores)
print("Mean cross-validation score:", scores.mean())

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

```
Classification Report:
              precision    recall  f1-score   support

       10-20       0.15      1.00      0.25         9
     100-110       0.97      0.42      0.59       262
     110-120       0.16      0.53      0.24        19
     120-130       0.89      0.71      0.79        48
     130-140       0.00       nan      0.00         0
     140-150       0.00      0.00      0.00         5
     150-160       0.00      0.00      0.00         2
     160-170       0.04      1.00      0.07         1
     180-190       0.93      0.93      0.93        15
       20-30       0.58      0.51      0.55       370
     210-220       0.00       nan      0.00         0
       30-40       0.57      0.36      0.44       334
       40-50       0.14      0.02      0.04       444
       50-60       0.35      0.04      0.07       439
       60-70       0.38      0.48      0.42       512
       70-80       0.36      0.60      0.45       485
       80-90       0.36      0.78      0.49       342
      90-100       0.35      0.54      0.42       121

    accuracy                           0.41      3408
   macro avg       0.35      0.50      0.32      3408
weighted avg       0.43      0.41      0.37      3408
```

```
Cross-validation scores: [0.4159331  0.41316311 0.41052168 0.41294299 0.41624477]
Mean cross-validation score: 0.41376112995383707
```



З отриманих результатів, можемо побачити, що в деяких категоріях не вистачає даних, щоб натренувати модель, тож і метрики не можна розрахувати. Загалом, класифікатор справляється з роботою гірше середнього. Причиною такого може бути велика кількість різноманітних ознак з недостатньо великим набором даних для класифікації.

Висновок: виконуючи лабораторну роботу, я набула навичок роботи з даними і опонувала роботу у Python з використанням теореми Байєса.

Контрольні запитання

1. Де застосовується наївний Байєс?

Аналіз текстів і обробка природної мови, медична діагностика, рекомендаційні системи, фільтрація контенту.

2. Поясніть теорему Байєса.

У статистиці і теорії ймовірностей теорема Байєса описує ймовірність події, гуртуючись на попередньому знанні умов, які можуть бути пов'язані з подією, тобто служить способом визначення умовної ймовірності.

Теорема Байєса задається наступним математичним рівнянням:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)},$$

де P(A | B) – це ймовірність події A, за умови, що подія B відбулась.

3. Які типи наївного байєсівського класифікатора існують?

Мультиноміальний (тексти), бернуллійський (бінарні ознаки), гаусовий (безперервні ознаки).