Лабораторная работа №1 по курсу "Интеллектуальный анализ данных"

выполнил Сорокин Никита, М8О-403Б-20

1.1 Наивный байесовский классификатор. Номинальный случай

Имеем пространство признаков и ответов $X = \{(x_1, \ldots, x_n) : x_i -$ значения i -ого признака $\}, y = \{y_i : y_i \in C\}$

Теорема Байеса:

$$P(y = c|x) = \frac{P(x|y = c)P(y = c)}{P(x)}$$

Сделаем допущение о том, что условные вероятности независимы, то есть:

$$P(x|y = c) = \prod_{i=1}^{n} P(x_i|y = c)$$

Воспользуемся этим для создания простейшего классификатора: будем принимать решение по значению вероятностей величины $P(y = c|x)$:

$$c^* = argmax_{c \in C} P(y = c|x) = argmax_{c \in C} \left[P(y = c) \prod_{i=1}^{n} P(x_i|y = c) \right]$$

Рассмотрим датасет погоды для игры в гольф:

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
df = pd.read_csv('csv_result-weather.csv')
df
```

Out[2]:

	id	outlook	temperature	humidity	windy	play
0	1	sunny	hot	high	False	no
1	2	sunny	hot	high	True	no
2	3	overcast	hot	high	False	yes
3	4	rainy	mild	high	False	yes
4	5	rainy	cool	normal	False	yes
5	6	rainy	cool	normal	True	no
6	7	overcast	cool	normal	True	yes
7	8	sunny	mild	high	False	no
8	9	sunny	cool	normal	False	yes
9	10	rainy	mild	normal	False	yes
10	11	sunny	mild	normal	True	yes
11	12	overcast	mild	high	True	yes
12	13	overcast	hot	normal	False	yes
13	14	rainy	mild	high	True	no

Спрогнозируем подходит ли следующая погода для игры в гольф по представленным данным

In [3]:
```
new_weather = {
    'id': [15],
    'outlook': 'rainy',
    'temperature': 'hot',
    'humidity': 'normal',
    'windy': True
}
new_weather_df = pd.DataFrame(data=new_weather)
new_weather_df
```

Out[3]:

	id	outlook	temperature	humidity	windy
0	15	rainy	hot	normal	True

In [4]:
```
prob_yes = (
    df[(df['outlook'] == 'rainy') & (df['play'] == 'yes')].count()[0] / df[df['play
  * df[(df['temperature'] == 'hot') & (df['play'] == 'yes')].count()[0] / df[df['pl
  * df[(df['outlook'] == 'rainy') & (df['play'] == 'yes')].count()[0] / df[df['play
  * df[(df['outlook'] == 'rainy') & (df['play'] == 'yes')].count()[0] / df[df['play
  * df[(df['play'] == 'yes')].count()[0]
)
prob_yes
```

Out[4]:
```
0.07407407407407407
```

In [5]:
```
prob_no = (
    df[(df['outlook'] == 'rainy') & (df['play'] == 'no')].count()[0] / df[df['play'
  * df[(df['temperature'] == 'hot') & (df['play'] == 'no')].count()[0] / df[df['pla
  * df[(df['outlook'] == 'rainy') & (df['play'] == 'no')].count()[0] / df[df['play'
  * df[(df['outlook'] == 'rainy') & (df['play'] == 'no')].count()[0] / df[df['play'
  * df[(df['play'] == 'no')].count()[0]
)
prob_no
```

Out[5]:
```
0.128
```

То есть для такой погоды классификатор считает, что идти на гольф не стоит, поскольку $P(y = \text{'yes'} \mid x) < P(y = \text{'no'} \mid x)$

Далее, построим классификатор с помощью библиотеки sklearn:

In [6]:
```
from sklearn.naive_bayes import CategoricalNB
from sklearn.preprocessing import OrdinalEncoder, LabelEncoder
```

In [7]:
```
enc_ordinal = OrdinalEncoder()

X_train = df.drop(['id', 'play'], axis=1)
X_train = enc_ordinal.fit_transform(X_train)
X_train[0]
```

Out[7]:
```
array([2., 1., 0., 0.])
```

In [8]:
```
enc_label = LabelEncoder()

y_train = df['play'].values
y_train = enc_label.fit_transform(y_train)
```

In [9]:
```
clf = CategoricalNB(alpha=1)

clf.fit(X_train, y_train)
```

Out[9]:
```
CategoricalNB(alpha=1)
```

In [10]:
```
new_weather_df = new_weather_df.drop(['id'], axis=1)
new_weather_enc = enc_ordinal.transform(new_weather_df)
```

In [11]:
```
clf.predict_proba(new_weather_enc)
```

Out[11]:
```
array([[0.39811362, 0.60188638]])
```

In [12]:
```
clf.get_params()
```

Out[12]:
```
{'alpha': 1,
 'class_prior': None,
 'fit_prior': True,
 'force_alpha': 'warn',
 'min_categories': None}
```

sklearn получает такой же результат

1.2 Наивный байесовский классификатор. Непрерывный случай

На самом деле баесовская классификация - это метод максимального правдоподобия. Таким образом при переходе к непрерывному признаковому пространству функция вероятности представляет собой не произведение вероятностей а произведение **плотностей**.

Пользуемся предположением о том, что

$$p(x = v|c) \sim N(\mu_c, \sigma_c^2), \quad p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

In [16]:
```
from sklearn.naive_bayes import GaussianNB
```

In [17]:
```
df = pd.read_csv('csv_result-weather_numeric.csv')
df
```

Out[17]:

	id	outlook	temperature	humidity	windy	play
0	1	sunny	85	85	False	no
1	2	sunny	80	90	True	no
2	3	overcast	83	86	False	yes
3	4	rainy	70	96	False	yes
4	5	rainy	68	80	False	yes
5	6	rainy	65	70	True	no
6	7	overcast	64	65	True	yes
7	8	sunny	72	95	False	no
8	9	sunny	69	70	False	yes
9	10	rainy	75	80	False	yes
10	11	sunny	75	70	True	yes
11	12	overcast	72	90	True	yes
12	13	overcast	81	75	False	yes
13	14	rainy	71	91	True	no

In [18]:
```
new_weather = {
    'id': [15],
    'outlook': 'rainy',
    'temperature': 85,
    'humidity': 90,
    'windy': True
}
new_weather_df = pd.DataFrame(data=new_weather)
new_weather_df
```

Out[18]:

	id	outlook	temperature	humidity	windy
0	15	rainy	85	90	True

Подготовка данных:

In [19]:
```
X_train_categorical = df[['outlook', 'windy']]
X_train_categorical = enc_ordinal.fit_transform(X_train_categorical)
X_train_categorical
```

Out[19]:
```
array([[2., 0.],
       [2., 1.],
       [0., 0.],
       [1., 0.],
       [1., 0.],
       [1., 1.],
       [0., 1.],
       [2., 0.],
       [2., 0.],
       [1., 0.],
       [2., 1.],
       [0., 1.],
       [0., 0.],
       [1., 1.]])
```

In [20]:
```
X_train_numerical = df[['temperature', 'humidity']]
X_train_numerical
```

Out[20]:

	temperature	humidity
0	85	85
1	80	90
2	83	86
3	70	96
4	68	80
5	65	70
6	64	65
7	72	95
8	69	70
9	75	80
10	75	70
11	72	90
12	81	75
13	71	91

In [21]:
```
y_train = df['play'].values
y_train = enc_label.fit_transform(y_train)
y_train
```

Out[21]:
```
array([0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0])
```

In [22]:
```
new_weather_df
```

Out[22]:

	id	outlook	temperature	humidity	windy
0	15	rainy	85	90	True

In [23]:
```
new_weather_categorical = new_weather_df[['outlook', 'windy']]
new_weather_categorical = enc_ordinal.transform(new_weather_categorical)
new_weather_categorical
```

Out[23]:
```
array([[1., 1.]])
```

In [24]:
```
new_weather_numerical = new_weather_df[['temperature', 'humidity']]
new_weather_numerical
```

Out[24]:

	temperature	humidity
0	85	90

Модели

In [25]:
```
clf_categorical = CategoricalNB()
clf_categorical.fit(X_train_categorical, y_train)
```

Out[25]:
```
▾ CategoricalNB
CategoricalNB()
```

In [26]:
```
clf_numerical = GaussianNB()
clf_numerical.fit(X_train_numerical, y_train)
```

Out[26]:
```
▾ GaussianNB
GaussianNB()
```

Предсказания

In [27]:
```
pred_categorical = clf_categorical.predict_proba(new_weather_categorical)
pred_categorical
```

Out[27]:
```
array([[0.4954955, 0.5045045]])
```

In [28]:
```
pred_numerical = clf_numerical.predict_proba(new_weather_numerical)
pred_numerical
```

Out[28]:
```
array([[0.71284406, 0.28715594]])
```

In [29]:
```
pred = pred_categorical * pred_numerical
pred
```

Out[29]:
```
array([[0.35321102, 0.14487147]])
```

Таким образом, классификатор предсказывает 'yes', то есть при такой погоде играть можно

1.2 Применение метода K ближайших соседей (kNN) в

задаче классификации

In [30]:
```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
```

Загружаем датасет из библиотеки sklearn

In [31]:
```
data = load_iris()
data.target
```

Out[31]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

Используем pandas для представления датасета в виде удобной таблице:

In [32]:
```
df = pd.DataFrame(data=data.data, columns=data.feature_names)
df['target'] = pd.Series(data.target)
df.iloc[:, :]
```

Out[32]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

Делим датасет на тренировочную и валидационную выборку:

In [37]:
```
X_train, X_val, y_train, y_val = train_test_split(
    df.iloc[:, :-1], df.iloc[:, -1], test_size=0.3, random_state=17
)
```

'Тренируем' knn для тестового датасета:

In [40]:
```
clf = KNeighborsClassifier()

clf.fit(X_train, y_train)
```

Out[40]:
```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

Предсказание:

In [43]:
```
y_pred = clf.predict(X_val)
```

Полученные метрики:

In [44]:
```
print(classification_report(y_val, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       0.90      1.00      0.95        19
           2       1.00      0.86      0.92        14

    accuracy                           0.96        45
   macro avg       0.97      0.95      0.96        45
weighted avg       0.96      0.96      0.95        45
```