ЗАДАНИЕ 1

Данные:

$\mathbf{x}_i$	a_1	a_2	a_3	Class
$\mathbf{x}_1$	T	T	5.0	Y
$\mathbf{x}_2$	T	T	7.0	Y
$\mathbf{x}_3$	T	F	8.0	N
$\mathbf{x}_4$	F	F	3.0	Y
$\mathbf{x}_5$	F	T	7.0	N
$\mathbf{x}_6$	F	T	4.0	N
$\mathbf{x}_7$	F	F	5.0	N
$\mathbf{x}_8$	T	F	6.0	Y
$\mathbf{x}_9$	F	T	1.0	N

Используя наивный байесовский классификатор определите класс точки (T,F,1.0)

1. Загрузим данные в датафрейм

	a1	a2	a3	Class
0	T	T	5	Y
1	T	T	7	Y
2	T	F	8	N
3	F	F	3	Y
4	F	T	7	N
5	F	T	4	N
6	F	F	5	N
7	T	F	6	Y
8	F	T	1	N

2. Заменим категориальные данные на численные.

Пусть Класс «Y» будет True, а класс «N» будет False

```
data['Class'] = data['Class'].map(lambda x : True if x == 'Y' else False)

for col in ['a1', 'a2']:
    data[col] = data[col].map(lambda x : True if x == 'T' else False)
data
```

	a1	a2	a3	Class
0	True	True	5	True
1	True	True	7	True
2	True	False	8	False
3	False	False	3	True
4	False	True	7	False
5	False	True	4	False
6	False	False	5	False
7	True	False	6	True
8	False	True	1	False

3. Загрузим необходимые для прогнозирования данные в датафрейм

```
to_predict = pd.DataFrame({'a1':[True], 'a2':[False], 'a3':[1]})
to_predict
```

	a1	a2	a3
0	True	False	1

1 способ: воспользоваться готовым решением

4. Воспользуемся классификатором GaussianNB

```
clf = GaussianNB()
clf.fit(data.drop('Class', axis=True), data['Class'])

clf.predict(to_predict)
```

Ответ: array([False]) – **принадлежит к классу N**

2 способ: вычислить «руками»

1.
```
n = data.shape[0]
n1 = np.count_nonzero(data['Class'])
n2 = n - n1
```

```
n=9 n1=4 n2=5
```

2.
```
p1 = n1 / n
p2 = n2 / n
```

p1= 0.4444444444444444 p2=0.5555555555555556

3.
```
m1 = np.mean(data[data['Class']==True].drop('Class', axis=True), axis=0)
m2 = np.mean(data[data['Class']==False].drop('Class', axis=True), axis=0)
```

```
m1 =
a1    0.75
a2    0.50
a3    5.25
dtype: float64
```

m2 =
```
a1    0.2
a2    0.6
a3    5.0
```

4.
```
cov1 = data[data['Class']==True].drop('Class', axis=True).cov()
cov2 = data[data['Class']==False].drop('Class', axis=True).cov()
```

cov1
```
          a1        a2        a3
a1  0.250000  0.166667  0.750000
a2  0.166667  0.333333  0.500000
a3  0.750000  0.500000  2.916667
```

cov2
```
      a1    a2    a3
a1  0.20 -0.15  0.75
a2 -0.15  0.30 -0.75
a3  0.75 -0.75  7.50
```

5.
```
print(stats.multivariate_normal.pdf(to_predict, m1, cov1) * p1)
print(stats.multivariate_normal.pdf(to_predict, m2, cov2) * p2)
```

```
3.4247331186367196e-10
7.48983846562541e-05
```

у второго класса вероятность выше, значит **принадлежит к классу N**

ЗАДАНИЕ 2

Даны два класса c1 and c2 со следующими мат. ожиданиями и матрицами ковариации:

$$\boldsymbol{\mu}_1 = (1, 3) \qquad \boldsymbol{\mu}_2 = (5, 5)$$

$$\boldsymbol{\Sigma}_1 = \begin{pmatrix} 5 & 3 \\ 3 & 2 \end{pmatrix} \qquad \boldsymbol{\Sigma}_2 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

Классифицируйте точку (3,4) используя Байесовский вывод, предположив, что классы распределены по нормальному закону, и P(c1) = P(c2) = 0.5

```
m1 = [1, 3]
m2 = [5, 5]

cov1 = [[5, 3], [3, 2]]
cov2 = [[2, 0], [0, 1]]

p1 = p2 = 0.5

to_predict = [3, 4]


print(stats.multivariate_normal.pdf(to_predict, m1, cov1)  * p1)
print(stats.multivariate_normal.pdf(to_predict, m2, cov2)  * p2)

0.048266176315027005
0.012555482738023718
```

Вероятность попасть в класс c1 выше — **принадлежит классу c1**

ЗАДАНИЕ 3

1. Загрузим данные в датасет

```
age =  pd.Series([25, 20, 25, 45, 20, 25])
car = pd.Series(['Sports', 'Vintage', 'Sports', 'SUV', 'Sports', 'SUV'])
risk = pd.Series(['L', 'H', 'L', 'H', 'H', 'H'])

data = pd.DataFrame({'Age':age, 'Car':car, 'Risk':risk})
```

	Age	Car	Risk
0	25	Sports	L
1	20	Vintage	H
2	25	Sports	L
3	45	SUV	H
4	20	Sports	H
5	25	SUV	H

2. Разобьем категориальные данные

```
df = data.join(pd.get_dummies(data['Car']))
df.drop('Car', axis=True, inplace=True)
df.head()
```

	Age	Risk	SUV	Sports	Vintage
0	25	L	0	1	0
1	20	H	0	0	1
2	25	L	0	1	0
3	45	H	1	0	0
4	20	H	0	1	0

3. Заменим категориальные данные на численные

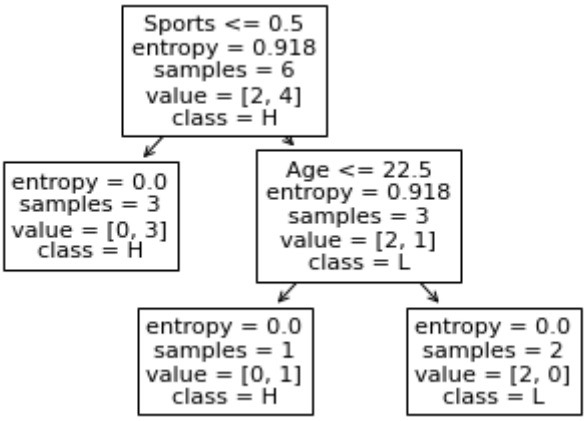
```
df['Risk'] = df['Risk'] == 'H'
```

	Age	Risk	SUV	Sports	Vintage
0	25	False	0	1	0
1	20	True	0	0	1
2	25	False	0	1	0
3	45	True	1	0	0
4	20	True	0	1	0
5	25	True	1	0	0

4. Воспользуемся классификатором DecisionTreeClassifier

```
clf = DecisionTreeClassifier(criterion='entropy')
clf.fit(df.drop('Risk', axis=True), df['Risk'])
```

5. Получим график дерева



6. Получим результат

```
clf.predict([[27, 0, 0, 1]])

array([ True])

с ответом True соотносится класс H
```

ответ: **принадлежит классу H**