Махмадзиеев Али 181-331

Задание 5 [до 10 баллов]3. По набору данных из задания 1 построить наивный классификатор Байеса.

1. Предобработать данные [до 2 баллов].

2. Построить и протестировать классификатор [до 2 баллов].

3. Реализовать метод, проверяющий значения признаков классифицируемого объекта на соответствие областям допустимых значений признаков и выявляющий

аномальные объекты [до 4 баллов].

4. Проиллюстрировать варианты эксплуатации классификатора [до 2 баллов].

Лабораторная работа №5

Считываем датасет:

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

```
B [2]: df = pd.read_csv('DfAli2.csv')
       df
```

Out[2]:

	Source	Target	Trade_Id	Bitcoins	Money	Money_Rate	Date	label
0	586159	100349	1373958491820869	1.250628	124.06234	99.200000	2013-07-16 07:08:11	0
1	199328	115248	1358039479966822	0.054551	0.77872	14.275064	2013-01-13 01:11:20	0
2	105211	96376	1329371016524489	13.543310	56.09165	4.141650	2012-02-16 05:43:36	0
3	69334	320942	1365522594463088	1.074300	239.99089	223.392805	2013-04-09 15:49:55	0
4	89169	28388	1322881924264838	8.443058	25.93792	3.072100	2011-12-03 03:12:04	0
...	...	...	...	...	...	...	...	...
67746	67321	31219	1322472062155919	4.840776	12.16971	2.514000	2011-11-28 09:21:02	0
67747	231	499498	1370529976369716	1.267475	153.61795	121.199951	2013-06-06 14:46:16	0
67748	THK	527401	1378329329034856	13.944896	188256.09400	13500.000031	2013-09-04 21:15:29	1
67749	90128	36865	1345470700972834	0.021053	0.19147	9.094549	2012-08-20 13:51:40	0
67750	231	235134	1354578128349347	0.010000	0.12660	12.660000	2012-12-03 23:42:08	0

67751 rows × 8 columns

Преобразование utc в datastamp:

```
B [3]: import datetime as DT
       for i, strdate in enumerate(df['Date']):
           dt = DT.datetime.strptime(strdate, '%Y-%m-%d %H:%M:%S')
           dt2 = dt.timestamp()
           df['Date'][i] = dt2
           print(i, dt2)
```

```
20 1382100444.0
21 1332320318.0
22 1346463956.0
23 1369732853.0
24 1309732937.0
25 1373144386.0
26 1340380915.0
27 1355830420.0
28 1369032888.0
29 1343927635.0
30 1372910027.0
31 1307354131.0
32 1365022940.0
33 1348115512.0
34 1355917563.0
35 1375798381.0
36 1347382887.0
37 1349091705.0
38 1374527651.0
```

В колонке Source имеются не числовые значения, преобразовываем их в числовые:

```
B [4]: for i, znach in enumerate(df['Source']):
           if(znach == 'THK' ):
               znach2 = 111
               df['Source'][i] = znach2
           elif(znach == 'TIBANNE_LIMITED_HK'):
               znach3 = 333
               df['Source'][i] = znach3
           else:
               df['Source'][i] = int(znach)
           print(i, type(df['Source'][i]))
```

```
246 <class 'int'>
247 <class 'int'>
248 <class 'int'>
249 <class 'int'>
250 <class 'int'>
251 <class 'int'>
252 <class 'int'>
253 <class 'int'>
254 <class 'int'>
255 <class 'int'>
256 <class 'int'>
257 <class 'int'>
258 <class 'int'>
259 <class 'int'>
260 <class 'int'>
261 <class 'int'>
262 <class 'int'>
263 <class 'int'>
264 <class 'int'>
265 <class 'int'>
```

Удаляем строки, содержащие NaN:

```
B [7]: df = df.dropna()
       df.shape
```

```
Out[7]: (67751, 8)
```

Разбиваем данные на тренировочные и тестовые (75%-25%):

```
B [8]: from sklearn.model_selection import train_test_split
```

```
B [9]: points_train, points_test, labels_train, labels_test = train_test_split(df.iloc[:, :-1], df['label'], test_size=0
```

Нормируем данные:

```
B [10]: from sklearn.preprocessing import StandardScaler
        ss = StandardScaler()
        ss.fit(points_train)
        points_train.iloc[:, :] = ss.transform(points_train)
        points_test.iloc[:, :] = ss.transform(points_test)
```

Строим классификатор «наивный метод Байеса»:

```
B [13]: print(points_train.shape, points_test.shape)
```

```
(50813, 7) (16938, 7)
```

```
B [14]: from sklearn.naive_bayes import GaussianNB
        gnb = GaussianNB()
        gnb.fit(points_train, labels_train)
```

```
Out[14]: GaussianNB()
```

```
B [15]: prediction = gnb.predict(points_test)
        print(points_test.assign(predict=prediction))
```

```
         Source    Target  Trade_Id  Bitcoins     Money  Money_Rate      Date  \
26938 -0.869112 -0.209965  0.193209  0.251463 -0.022716   -0.066498  0.339523
26030  1.224167 -0.974354  0.293816 -0.174730 -0.038919   -0.010965  1.327846
14684  2.046214 -0.733441  0.232184 -0.163508 -0.034269   -0.001966  0.722397
8193   1.053076  2.281185  0.293572 -0.151917 -0.030732   -0.011321  1.325448
15776 -0.021237  3.667124  0.279197 -0.174602 -0.021030    8.539579  1.184238
...         ...       ...       ...       ...       ...         ...       ...
52961 -0.850914  0.231069  0.222904 -0.174730 -0.038941   -0.027255  0.631231
57465 -0.776444 -0.974354  0.265505 -0.173556 -0.038641   -0.028800  1.049727
46780 -0.756565 -0.804173 -6.036450 -0.137861 -0.037249   -0.064940 -1.937066
2162  -0.823536 -0.941829 -0.002740  0.055473 -0.035735   -0.070629 -1.585401
3103  -0.881209  0.553082  0.175613 -0.145709  0.042157    0.404320  0.166668

       predict
26938        0
26030        0
14684        0
8193         0
15776        1
...        ...
52961        0
57465        0
46780        0
2162         0
3103         1

[16938 rows x 8 columns]
```

```
B [16]: print(format(gnb.score(points_test, labels_test)))
```

```
0.916814263785571
```

Даем классификатору новые данные:

```
B [11]: points_new = pd.DataFrame({'Source':[0, 111],
                                   'Target':[0, 527401],
                                   'Trade_Id':[0, 1378329329034856],
                                   'Bitcoins':[0, 13.944896],
                                   'Money':[0, 1882.09400],
                                   'Money_Rate':[0, 1350.000031],
                                   'Date':[0, 1378318529.0] })
```

Выполняем проверку значений признаков на доверительные интервалы:

```
B [32]: predict_s_proverkoi = []
def proverka_na_dov_intervals():
    anomalyclass = 2
    for i in points_new.index:
        elem = prediction_new[i]
        if (points_new.iloc[i][0] < otric_dovint_Source or points_new.iloc[i][0] > poloj_dovint_Source) or (points_new.iloc[i][1]
            predict_s_proverkoi.append(anomalyclass)
        else:
            predict_s_proverkoi.append(int(elem))
    print(predict_s_proverkoi)
    print(len(predict_s_proverkoi))
```

Получаем результат классификации на новых данных с проверкой на доверительные интервалы:

```
B [33]: prediction_new = gnb.predict(points_new)
```

```
B [34]: print(proverka_na_dov_intervals())
```

```
[2, 1]
2
None
```

```
B [35]: points_new.assign(label=predict_s_proverkoi)
```

Out[35]:

	Source	Target	Trade_Id	Bitcoins	Money	Money_Rate	Date	label
0	-0.881971	-0.975890	-6.036450	-0.174961	-0.039002	-0.073069	-60.857728	2
1	-0.881209	2.529156	0.284819	0.146376	0.075151	0.538324	1.239463	1

```
B [ ]:
```