Задание 3 [до 10 баллов]

1. По набору данных из задания 1 на основе метода k ближайших соседей построить классификатор.

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].

Дата сет «СЛИТЫЕ БИТКОИН ТРАНЗАКЦИИ»

Ссылка: https://www.kaggle.com/xblock/mtgox-leaked-transaction

```
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

Удаляем строки, содержащие NaN:

```
B [7]: df.shape
```

Out[7]: (67751, 8)

```
B [8]: df = df.dropna()
       df.shape
```

Out[8]: (67751, 8)

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

```
B [9]: from sklearn.model_selection import train_test_split
```

```
B [10]: points_train, points_test, labels_train, labels_test = train_test_split(df.iloc[:, :-1], df['label'], test_size=0.25, random_stat
```

Нормируем данные:

```
B [11]: from sklearn.preprocessing import StandardScaler
```

```
B [12]: ss = StandardScaler()
        ss.fit(points_train)
        points_train.iloc[:, :] = ss.transform(points_train)
        points_test.iloc[:, :] = ss.transform(points_test)
```

Строим классификатор:

```
B [16]: from sklearn.neighbors import KNeighborsClassifier
```

```
B [17]: knn = KNeighborsClassifier(n_neighbors=3, metric='euclidean')
        knn.fit(points_train, labels_train)
```

```
Out[17]: KNeighborsClassifier(metric='euclidean', n_neighbors=3)
```

```
B [18]: prediction = knn.predict(points_test)
        print(prediction)
        print(points_test.assign(predict=prediction))
```

```
[0 0 0 ... 0 0 1]
         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]
```

Результат:

```
B [19]: print(format(knn.score(points_test, labels_test)))
```

```
0.9845908607863975
```

Высчитываем мат.ожидание, среднеквадратическое отклонение, отрицательные и положительные доверительные интервалы для каждого признака:

In [21]:
```
# Souce mean std dovint
```

In [22]:
```
mean_Source = np.mean(df['Source'])
std_Source = np.std(df['Source'])
poloj_dovint_Source = mean_Source + 3 * std_Source
otric_dovint_Source = mean_Source - 3 * std_Source
print(mean_Source, "Мат. ожидание Source")
print(std_Source, "Среднеквадратическое отклонение Source")
print(otric_dovint_Source, "Отрицательный доверительный интервал для Source")
print(poloj_dovint_Source, "Положительный доверительный интервал для Source")
```

```
1.2742386443720708e-17 Мат. ожидание Source
0.9999999999999999 Среднеквадратическое отклонение Source
-2.9999999999999996 Отрицательный доверительный интервал для Source
2.9999999999999996 Положительный доверительный интервал для Source
```

In [23]:
```
# Target mean std dovint
```

In [24]:
```
mean_Target = np.mean(df['Target'])
std_Target = np.std(df['Target'])
poloj_dovint_Target = mean_Target + 3 * std_Target
otric_dovint_Target = mean_Target - 3 * std_Target
print(mean_Target, "Мат. ожидание Target")
print(std_Target, "Среднеквадратическое отклонение Target")
print(otric_dovint_Target, "Отрицательный доверительный интервал для Target")
print(poloj_dovint_Target, "Положительный доверительный интервал для Target")
```

```
2.2967758281274362e-17 Мат. ожидание Target
1.0 Среднеквадратическое отклонение Target
-3.0 Отрицательный доверительный интервал для Target
3.0 Положительный доверительный интервал для Target
```

In [25]:
```
# Trade_Id mean std dovint
```

In [26]:
```
mean_Trade_Id = np.mean(df['Trade_Id'])
std_Trade_Id = np.std(df['Trade_Id'])
poloj_dovint_Trade_Id = mean_Trade_Id + 3 * std_Trade_Id
otric_dovint_Trade_Id = mean_Trade_Id - 3 * std_Trade_Id
print(mean_Trade_Id, "Мат. ожидание Trade_id")
print(std_Trade_Id, "Среднеквадратическое отклонение Trade_Id")
print(otric_dovint_Trade_Id, "Отрицательный доверительный интервал для Trade_Id")
print(poloj_dovint_Trade_Id, "Положительный доверительный интервал для Trade_Id")
```

```
2.2967758281274362e-17 Мат. ожидание Trade_id
1.0 Среднеквадратическое отклонение Trade_Id
-3.0 Отрицательный доверительный интервал для Trade_Id
3.0 Положительный доверительный интервал для Trade_Id
```

В [27]:
```
# Bitcoins mean std dovint
```

В [28]:
```
mean_Bitcoins = np.mean(df['Bitcoins'])
std_Bitcoins = np.std(df['Bitcoins'])
poloj_dovint_Bitcoins = mean_Bitcoins + 3 * std_Bitcoins
otric_dovint_Bitcoins = mean_Bitcoins - 3 * std_Bitcoins
print(mean_Bitcoins, "Мат. ожидание Bitcoins")
print(std_Bitcoins, "Среднеквадратическое отклонение Bitcoins")
print(otric_dovint_Bitcoins, "Отрицательный доверительный интервал для Bitcoins")
print(poloj_dovint_Bitcoins, "Положительный доверительный интервал для Bitcoins")
```

```
2.2967758281274362e-17 Мат. ожидание Bitcoins
1.0 Среднеквадратическое отклонение Bitcoins
-3.0 Отрицательный доверительный интервал для Bitcoins
3.0 Положительный доверительный интервал для Bitcoins
```

В [29]:
```
# Money mean std dovint
```

В [30]:
```
mean_Money = np.mean(df['Money'])
std_Money = np.std(df['Money'])
poloj_dovint_Money = mean_Money + 3 * std_Money
otric_dovint_Money = mean_Money - 3 * std_Money
print(mean_Money, "Мат. ожидание Money")
print(std_Money, "Среднеквадратическое отклонение Money")
print(otric_dovint_Money, "Отрицательный доверительный интервал для Money")
print(poloj_dovint_Money, "Положительный доверительный интервал для Money")
```

```
2.2967758281274362e-17 Мат. ожидание Money
1.0 Среднеквадратическое отклонение Money
-3.0 Отрицательный доверительный интервал для Money
3.0 Положительный доверительный интервал для Money
```

В [31]:
```
# Money_Rate mean std dovint
```

В [32]:
```
mean_Money_Rate = np.mean(df['Money_Rate'])
std_Money_Rate = np.std(df['Money_Rate'])
poloj_dovint_Money_Rate = mean_Money_Rate + 3 * std_Money_Rate
otric_dovint_Money_Rate = mean_Money_Rate - 3 * std_Money_Rate
print(mean_Money_Rate, "Мат. ожидание Money_Rate")
print(std_Money_Rate, "Среднеквадратическое отклонение Money_Rate")
print(otric_dovint_Money_Rate, "Отрицательный доверительный интервал для Money_Rate")
print(poloj_dovint_Money_Rate, "Положительный доверительный интервал для Money_Rate")
```

```
2.2967758281274362e-17 Мат. ожидание Money_Rate
1.0 Среднеквадратическое отклонение Money_Rate
-3.0 Отрицательный доверительный интервал для Money_Rate
3.0 Положительный доверительный интервал для Money_Rate
```

В [33]:
```
# Date mean std dovint
```

В [34]:
```
mean_Date = np.mean(df['Date'])
std_Date = np.std(df['Date'])
poloj_dovint_Date = mean_Date + 3 * std_Date
otric_dovint_Date = mean_Date - 3 * std_Date
print(mean_Date, "Мат. ожидание Date")
print(std_Date, "Среднеквадратическое отклонение Date")
print(otric_dovint_Date, "Отрицательный доверительный интервал для Date")
print(poloj_dovint_Date, "Положительный доверительный интервал для Date")
```

```
2.2967758281274362e-17 Мат. ожидание Date
1.0 Среднеквадратическое отклонение Date
-3.0 Отрицательный доверительный интервал для Date
3.0 Положительный доверительный интервал для Date
```

Даем классификатору новые данные:

```
В [13]: 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] })

В [14]: points_new.iloc[:, :] = ss.transform(points_new)
```

Выполняем проверку значений признаков на доверительные интервалы:

```
В [35]: 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:

```
В [36]: prediction_new = knn.predict(points_new)

В [37]: print(proverka_na_dov_intervals())

        [2, 1]
        2
        None

В [ ]:

В [38]: points_new.assign(label=predict_s_proverkoi)
```

Out[38]:

	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

```
В [ ]:
```