Махмадзиеев Али Олимович 181-331

Задание 4 [до 10 баллов] По набору данных из задания 1 на основе метода решающих деревьев 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].

Лабораторная работа №4

Считаем датасет:

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

```
In [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:

```
In [7]: df = df.dropna()
        df.shape
Out[7]: (67751, 8)
```

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

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

```
In [9]: points_train, points_test, labels_train, labels_test = train_test_split(df.iloc[:, :-1], df['label'], test_size=0.25, random_stat
```

```
In [ ]:
```

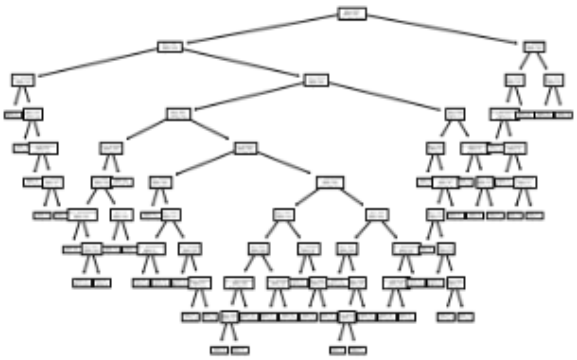
```
In [10]: print(points_train.shape, points_test.shape)
         (50813, 7) (16938, 7)
```

Строим классификатор «метод решающих деревьев», указываем максимальную глубину дерева 10 и в criterion энтропию:

```
В [11]: from sklearn.tree import DecisionTreeClassifier
        dt = DecisionTreeClassifier(criterion='entropy', max_depth=10)
        dt.fit(points_train, labels_train)

Out[11]: DecisionTreeClassifier(criterion='entropy', max_depth=10)
```

```
В [12]: import matplotlib.pyplot as plt
        from sklearn import tree
        plt.figure()
        tree.plot_tree(dt)
        plt.show()
```



Результат классификации:

```
В [14]: print(format(dt.score(points_test, labels_test)))

        0.9995276892195064
```

Даём новые данные классификатору:

```
В [30]: 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] })
```

Проверяем значения признаков на доверительные интервалы:

```
B [29]: 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))
```

Результат классификации:

```
B [31]: prediction_new = dt.predict(points_new)
```

```
B [32]: print(proverka_na_dov_intervals())
```

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

```
B [33]: points_new.assign(label=predict_s_proverkoi)
```

Out[33]:

	Source	Target	Trade_Id	Bitcoins	Money	Money_Rate	Date	label
0	0	0	0	0.000000	0.000	0.000000	0.000000e+00	2
1	111	527401	1378329329034856	13.944896	1882.094	1350.000031	1.378319e+09	1

```
B [ ]:
```