Данила Борисович Шмаков 181-351

Задание 3

Предобработка данных.

```
def normalize(data, quantity):
    for i in quantity:
        k = 0
        for item in data[i]:
            if item == '-':
                data[i][k] = np.nan
            k += 1

    imp = SimpleImputer(missing_values=np.nan, strategy='mean')
    imp.fit(data[quantity])
    data[quantity] = imp.transform(data[quantity])
    print('Количество пропусков в:', data.isnull().sum())
    return data
```

Так как в количественных данных встречаются '-', то их нужно убрать. Я заменяю их на Nan и удаляю наны. Такая предобработка будет использоваться и в дальнейшем, поэтому я вынес функцию в отдельный файл и не буду упоминать об этой предобработке в лабораторных 4-5.

Далее нужно перевести качественные в количественные равноправные признаки.

```
data = pd.get_dummies(data, columns=['proto', 'conn_state'])
dtype = float;
```

Далее высчитываем доверительные интервалы для каждого количественного признака и удаляем аномалии (значения вне интервалов).

```
for index in quantity:
     trust_old[index] = [np.mean(data[index]) - (3 * np.std(data[index])),
                 np.mean(data[index]) + (3 * np.std(data[index]))]
     data.loc[(data[index] > trust_old[index][1]) | (data[index] <
trust_old[index][0]), index] = np.nan

data = data.dropna()
```

Преобразуем метку в количественных признак

```
le = LabelEncoder()
le.fit(labels.label)
labels['label'] = le.transform(labels.label)
data = data.assign(label=labels['label'])
```

Нормализуем данные и находим дов. Интервалы для нормализованных данных.

```
ss = StandardScaler()
data.iloc[:, :-1] = ss.fit_transform(data.iloc[:, :-1])
trust = dict()
for index in quantity:
    trust[index] = [np.mean(data[index]) - (3 * np.std(data[index])),
                    np.mean(data[index]) + (3 * np.std(data[index]))]
```

Создаем и обучаем классификатор

```
points train, points test, labels train, labels test =
train_test_split(data.iloc[:, :-1], data['label'],

test_size=0.25,random_state=0)

knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')
knn.fit(points_train, labels_train)
prediction = knn.predict(points_test)
```

Создаем новую пробную точку и нормализуем ее

```
test point = {'duration': [2.71], 'orig bytes': [3], 'orig pkts': [3],
'resp_bytes': [2.7], 'resp_pkts': [3],
          'proto_icmp': [0], 'proto_udp': [0], 'proto_tcp': [1],
'conn_state_S0': [0],
          'conn_state_SF': [0], 'conn_state_REJ': [1], 'conn_state_OTH': [0],
'conn_state_RSTR': [0]}

data_new = pd.DataFrame(test_point)

ss.fit(data.iloc[:, :-1])
data_new.iloc[:, :] = ss.transform(data_new.iloc[:, :])
```

Проверяем входит ли в доверительные интервалы точка и скармливаем классификатору

```
if funcs.check_data(data_new, trust, quantity):
    prediction = knn.predict(data_new)
    data_new = data_new.assign(label=prediction)
    data = data.append(data_new, ignore_index=True)
```

Выводим оценку

```
print('ОЦЕНКА: ', format(knn.score(points_test, labels_test)))
```

```
ОЦЕНКА:  1.0
```