Шмаков Данила 181-351

Задание 7

H	I	J	K	L	M	N	O	P	Q	R	S	T
service	duration	orig_bytes	resp_bytes	conn_state	local_orig	local_resp	missed_bytes	history	orig_pkts	orig_ip_bytes	resp_pkts	resp_ip_bytes
-	2.998333	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.997182	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.996286	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.995263	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.999262	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.998315	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.999395	0	0	S0	-	-	0	S	3	180	0	0
-	0.012703	48	48	SF	-	-	0	Dd	1	76	1	76
-	35.017664	288	288	SF	-	-	0	Dd	6	456	6	456
-	38.017650	288	288	SF	-	-	0	Dd	6	456	6	456
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.998379	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.997387	0	0	S0	-	-	0	S	3	180	0	0
-	-	-	-	S0	-	-	0	S	1	60	0	0
-	2.996386	0	0	S0	-	-	0	S	3	180	0	0

Прочерки содержат только столбцы: duration, orig_bytes, resp_bytes

local_orig/resp полностью состоят из пропусков

Для этого задания используется значительно большее количество признаков

```
data = pd.read_csv('IoT.csv', delimiter=',')
quantity = ['ts', 'duration', 'orig_bytes', 'resp_bytes', 'local_orig', 'local_resp',
            'missed_bytes', 'orig_pkts', 'resp_pkts', 'orig_ip_bytes', 'resp_ip_bytes']
quality = ['uid', 'id.orig_h',  'id.orig_p', 'id.resp_h', 'id.resp_p',
           'proto', 'service', 'conn_state', 'history', 'tunnel_parents']
# print(data)
```

ПредПредобработка данных такая же как и в заданиях 3-5 (заменяем '-' на среднее значение). Также в отличие от 3-5 заменяем nan на 0, так как используются полностью пустые признаки.

```
# только эти 3 признака содержат '-'
imp = SimpleImputer(missing_values=np.nan, strategy='mean')
data[['duration', 'orig_bytes', 'resp_bytes']] = imp.fit_transform(data[['duration', 'orig_bytes', 'resp_bytes']])

# так как есть полностью пустые признаки меняем их на нули
imp = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value=0)
data[data.columns] = imp.fit_transform(data[data.columns])
```

Кодируем качественные в количественные признаки

```
le = LabelEncoder()

for item in quality:
    data[item]= le.fit_transform(data[item])
    dtype = float
```

Нормируем. (-2, так как теперь еще используется detailed-label)

```
ss = StandardScaler()
data.iloc[:, :-2] = ss.fit_transform(data.iloc[:, :-2])
```

Создаем объект метода главных компонент

```
pca = PCA(svd_solver='full')
```

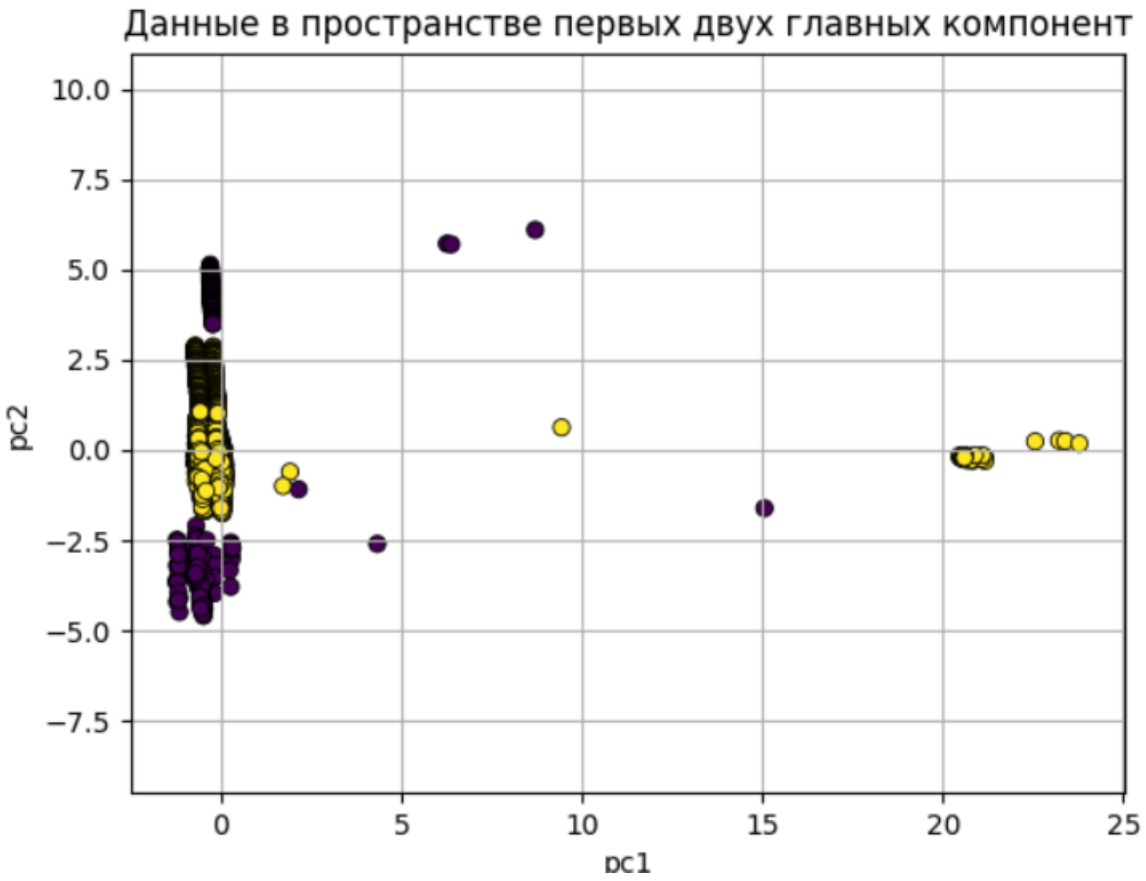
Используем метод главных компонент

```
data2 = data.copy()

column = []
for i in range(len(data.columns) -2):
    column.append(f'pc{i+1}')
column.append('label')
column.append('detailed-label')
data2.columns = column
data2.iloc[:, :-2] = pca.fit_transform(data2.iloc[:, :-2])
```

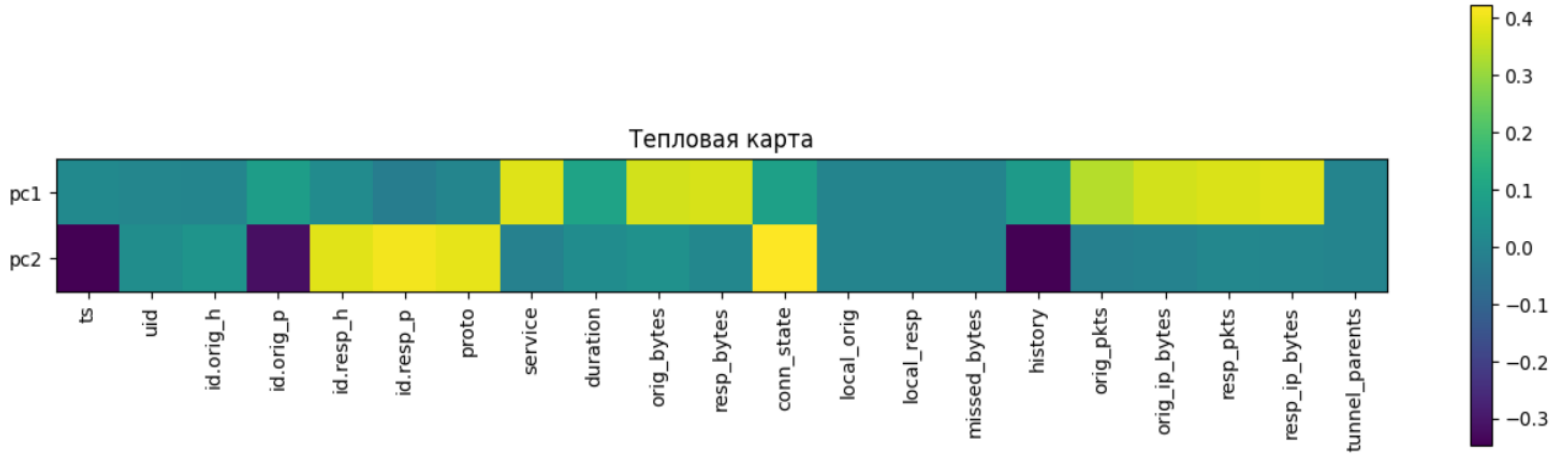
Рисуем данные в пространстве главных компонент

```
plt.figure()
plt.grid()
plt.scatter(data2['pc1'], data2['pc2'], c=le.fit_transform(data2['label']), lw=.6, edgecolors='black')
plt.axis('equal')
plt.title("Данные в пространстве первых двух главных компонент")
plt.xlabel("pc1")
plt.ylabel("pc2")
plt.show()
```



Рисуем тепловую карту

```
plt.matshow(pca.components_[:2])
plt.colorbar()
plt.gca().xaxis.tick_bottom()
plt.xticks(range(len(data.columns) - 2), data.iloc[:, :-2], rotation=90)
plt.yticks(range(2), data2[['pc1', 'pc2']])
plt.title("Тепловая карта")
plt.show()
```



Можно заметить, что признаки uid, id.orig_h, duration, tunnel_parents, (local_orig/resp, missed_bytes содержат одни нули) наименее информативны, так как значения близки к нулю. Признаки ts, id.orig_h, id.resp_h, proto, conn_state, history – наиболее информативны так как их значение наиболее отлично от нуля по первой компоненте.