Модуль 2

Загружаем датасет из предыдущего модуля и немножко его преобрабатываем

In [1]:

```
#Импортируем библиотеки
import pandas as pd #библиотека для работы с датасетом
import warnings #игнорирование ошибок
warnings.filterwarnings("ignore")
```

In [2]:

```
#Загружаем обработанный датасет в предыдущей сессии
df = pd.read_csv('C3_M1.csv')
df = df.drop('Unnamed: 0', axis=1)
```

In [3]:

```
#Удаляем мешающие символы в признаке "покрытие дороги"
df['road_conditions'] = df['road_conditions'].str.replace('[','')
df['road_conditions'] = df['road_conditions'].str.replace(']','')
df['road_conditions'] = df['road_conditions'].str.replace("'",'')
```

In [4]:

```
# Кодируем объекты в целочисленные цифры
from sklearn.preprocessing import LabelEncoder

#Цикл кодировки
LE = LabelEncoder()
for i in df:
    if df[i].dtype == 'object':
        LE.fit(df[i])
        df[i] = LE.transform(df[i])
```

In [5]:

```
df
```

Out[5]:

	light	region	address	category	severity	injured_count	parent_region	rc
0	4	150	110982	16	0	4	0	
1	4	1429	75284	16	2	3	0	
2	4	1830	75232	12	2	1	0	
3	2	815	26278	9	1	0	0	
4	0	150	112217	9	2	1	0	

...	...	...	...	...	...	...	...
735091	0	252	390265	9	2	1	84
735092	4	252	91218	11	1	0	84
735093	2	252	12981	2	1	0	84
735094	4	252	91221	16	0	1	84
735095	4	252	91219	17	2	1	84

735096 rows × 9 columns

2.1 Формирование дополнительных атрибутов

Вычисление индекса происшествий Индекс происшествий рассчитывается по формуле - X = (qfp)/r. Где X - Искомый индекс; q - кол-во происшествий; f - частота происхождения за год; p - средняя тяжесть (1 - легкое, 2 - среднее, 3 - тяжёлое); r - доля происшествий в регионе\городе;

Пример счёта индекса из города Братск, по адресу проезд Стройиндустрии, 0, где за последние 6 лет произошло 55 случаев ДТП

`x = (55x7,8x2)/1242`
`x = 0.620` - индекс опасности участка и рекомендации к изменениям

2.3 Разбиение набора данных

Разбиение набора данных на обучающую и тестирующую выборки будем осуществлять через скейлирование из библиотеки sklearn, потому что это наиболее быстрый и оптимальный способ. Выборку будем делать предварительно и в будущем соотношение может поменяться. Пока оставляем 66% обучающей и 33% тестирующей с рандомным шагом 42.

In [6]:

```
# Масштабирование таблицы
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df.drop('severity', axis=1))
scaled_features = scaler.transform(df.drop('severity', axis=1))
scaled_data = pd.DataFrame(scaled_features, columns = df.drop('severity',
axis=1).columns)
```

In [7]:

```
# Загружаем библиотеку для разделения на выборки
from sklearn.model_selection import train_test_split

# Разделение в соотношении 66/33 с шагом 42
X_train, X_test, y_train, y_test = train_test_split(scaled_data, df.severi
ty, test_size=0.33, random_state=42)
```

2.2 Кластеризация набора данных

In [8]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(X_train, y_train)
p = kmeans.predict(X_test)
p
```

Out[8]:

```
array([1, 2, 2, ..., 2, 2, 2])
```

Метод показал точность 80%

In [9]:

```
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor(n_estimators=50)
rf.fit(X_train, y_train)
m = rf.predict(X_test)
m
```

Out[9]:

```
array([1.  , 0.16, 0.48, ..., 0.64, 0.8 , 0.  ])
```

Метод показал точность 68%

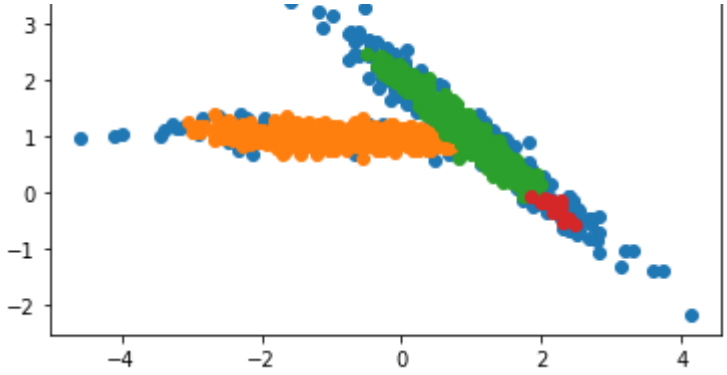
In [12]:

```
#Импортируем библиотеки
import numpy as np
from sklearn.datasets import make_classification
from sklearn.cluster import DBSCAN
from matplotlib import pyplot

# Определяем датасет
X, _ = make_classification(n_samples=1000, n_features=3, n_informative=3,
n_redundant=0, n_clusters_per_class=1, random_state=4)
# Определяем модель
model = DBSCAN(eps=0.30, min_samples=9)
#Создаём предсказание
yhat = model.fit_predict(X)
#Создаём кластеры
clusters = np.unique(yhat)

#Создаём для каждого кластера матрицу
for cluster in clusters:
    row_ix = np.where(yhat == cluster)
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])

#Показываем график
pyplot.show()
```



Модель показала 3 кластера и данные, выделяющиеся от остальных данных (шумы)

In [13]:

```
#Импортируем библиотеки
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

#Исключаем информацию, сохраняем для дальнейшего использования
varieties = list(df.pop('severity'))

# Извлекаем измерения как массив NumPy
samples = df.values

#Реализация иерархической кластеризации при помощи функции linkage
mergings = linkage(samples, method='complete')

#Строим дендрограмму, указав параметры удобные для отображения
dendrogram(mergings,
           labels=varieties,
           leaf_rotation=90,
           leaf_font_size=6,
           )

plt.show()
```

```
---------------------------------------------------------------
--------------
MemoryError                               Traceback (most rec
ent call last)
<ipython-input-13-5b90b86e9a73> in <module>
     10
     11 #Реализация иерархической кластеризации при помощи фу
нкции linkage
---> 12 mergings = linkage(samples, method='complete')
     13
     14 #Строим дендрограмму, указав параметры удобные для от
ображения

~\AppData\Roaming\Python\Python37\site-packages\scipy\cluster
\hierarchy.py in linkage(y, method, metric, optimal_ordering)
   1058                          'matrix looks suspiciously l
ike an uncondensed '
   1059                          'distance matrix')
-> 1060         y = distance.pdist(y, metric)
   1061     else:
   1062         raise ValueError("`y` must be 1 or 2 dimensio
nal ")
```

```
~\AppData\Roaming\Python\Python37\site-packages\scipy\spatial
\distance.py in pdist(X, metric, out, **kwargs)
   2248         if metric_info is not None:
   2249             pdist_fn = metric_info.pdist_func
-> 2250             return pdist_fn(X, out=out, **kwargs)
   2251         elif mstr.startswith("test_"):
   2252             metric_info = _TEST_METRICS.get(mstr, Non
e)

MemoryError: Unable to allocate 1.97 TiB for an array with sh
ape (270182697060,) and data type float64
```

Методу не достаточно кол-во ОЗУ на рабочей станции.

Среди всех методов кластеризации наилучшим оказался метот DBSCAN

2.4 Отчёт

In [14]:

```
#Сохранение в CSV
df.to_csv('C3_M2.csv')
#Сохранения разбиение данных
X_train.to_csv('X_train.csv')
X_test.to_csv('X_test.csv')
y_train.to_csv('y_train.csv')
y_test.to_csv('y_test.csv')
```

In [15]:

```
#Сохранение HTML
!jupyter nbconvert C3_M2.ipynb --to html
```

```
[NbConvertApp] Converting notebook C3_M2.ipynb to html
[NbConvertApp] Writing 426250 bytes to C3_M2.html
```

In [ ]:

```

```