Оглавление

Практическая часть (1)

Набор данных: https://www.kaggle.com/paultimothymooney/denver-crime-data

Необходимо провести анализ датасета и сделать обработку данных по предложенному алгоритму. Ответить на следующие вопросы:

1. Сколько в датасете объектов и признаков? Дать описание каждому признаку, если оно есть.
2. Сколько категориальных признаков, какие?
3. Столбец с максимальным количеством уникальных значений категориального признака?
4. Есть ли бинарные признаки?
5. Какие числовые признаки?
6. Есть ли пропуски?
7. Сколько объектов с пропусками?
8. Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?

* Доп. задание: построить двухмерное представление данных с помощью алгоритма t-SNE. На сколько кластеров визуально на ваш взгляд разделяется выборка?

Решение

1. Сколько в датасете объектов и признаков? Дать описание каждому признаку, если оно есть.

Размеры таблицы: 487997 rows × 19 columns => в датасете 487997 объектов и 19 признаков.

Описание признаков:

1) *INCIDENT_ID* - id инцидента (уникальное значение)
2) *OFFENSE_ID* - id правонарушения (уникальное значение)
3) *OFFENSE_CODE* - код правонарушения (категориальное значение)
4) *OFFENSE_CODE_EXTENSION* - код обобщенного правонарушения (категориальное значение)
5) *OFFENSE_TYPE_ID* – правонарушение (категориальное значение)
6) *OFFENSE_CATEGORY_ID* - категория правонарушения (категориальное значение)
7) *FIRST_OCCURRENCE_DATE* - дата первого происшествия (нельзя назвать числовым или категориальным)
8) *LAST_OCCURRENCE_DATE* - дата последнего происшествия (нельзя назвать числовым или категориальным)
9) *REPORTED_DATE* - дата заявления (нельзя назвать числовым или категориальным)
10) *INCIDENT_ADDRESS* - адрес инцидента (нельзя назвать числовым или категориальным)
11) *GEO_X* - (числовое значение)
12) *GEO_Y* - (числовое значение)
13) *GEO_LON* – долгота (числовое значение)
14) *GEO_LAT* – широта (числовое значение)
15) *DISTRICT_ID* - id района (категориальное значение)
16) *PRECINCT_ID* - id избирательного участка (категориальное значение)

17) *NEIGHBORHOOD_ID* – соседство (категориальное значение)
18) *IS_CRIME* - является ли преступлением (бинарное значение)
19) *IS_TRAFFIC* - наличие траффика (бинарное значение)

2. Сколько категориальных признаков, какие?

Категориальных признаков 7:

1) *OFFENSE_CODE*
2) *OFFENSE_CODE_EXTENSION*
3) *OFFENSE_TYPE_ID*
4) *OFFENSE_CATEGORY_ID*
5) *DISTRICT_ID*
6) *PRECINCT_ID*
7) *NEIGHBORHOOD_ID*

3. Столбец с максимальным количеством уникальных значений категориального признака?

Выполним код:

```
categorical = ['OFFENSE_CODE', 'OFFENSE_CODE_EXTENSION', 'OFFENSE_TYPE_ID', 'OFFENSE_CATEGORY_ID', 'DISTRICT_ID', 'PRECINCT_ID', 'NEIGHBORHOOD_ID']
for column in df.columns:
    if column in categorical:
        print(column, df[column].nunique())
```

```
OFFENSE_CODE 155
OFFENSE_CODE_EXTENSION 6
OFFENSE_TYPE_ID 200
OFFENSE_CATEGORY_ID 15
DISTRICT_ID 7
PRECINCT_ID 36
NEIGHBORHOOD_ID 79
```

Рисунок 7. Результат работы команды print(column, df[column].nunique())

Из рисунка 7 видно, что максимальное количество уникальных значений имеет столбец *OFFENSE_TYPE_ID*. Он содержит 200 уникальных значений.

4. Есть ли бинарные признаки?

Выполним код:

```
i = 1
for column in df.columns:
    if df[column].nunique() == 2:
        print(i, column)
        i += 1
```

Из описания признаков, приведённого в пункте 1, а также по результатам работы функции видно, что бинарных признаков 2: *IS_CRIME* и *IS_TRAFFIC*.

5. Какие числовые признаки?

Выполним код:

```
i = 1

for column in df.columns:
    if df[column].nunique() > 200 and column != 'INCIDENT_ADDRESS' and column != 'FIRST_OCCURRENCE_DATE' and column != 'LAST_OCCURRENCE_DATE' and column != 'REPORTED_DATE':
        print(i, column, df[column].nunique())
        i += 1
```

```
1 INCIDENT_ID 455362
2 OFFENSE_ID 487997
3 GEO_X 47407
4 GEO_Y 44975
5 GEO_LON 96426
6 GEO_LAT 95893
```

Рисунок 8. Результат работы функции print(i, column, df[column].nunique())

Из описания признаков, приведённого в пункте 1 и из результатов работы функции видно, что чисто числовыми являются *INCIDENT_ID, OFFENSE_ID, GEO_X, GEO_Y, GEO_LON, GEO_LAT*.

6. Есть ли пропуски?

```
INCIDENT_ID                  0
OFFENSE_ID                   0
OFFENSE_CODE                 0
OFFENSE_CODE_EXTENSION       0
OFFENSE_TYPE_ID              0
OFFENSE_CATEGORY_ID          0
FIRST_OCCURRENCE_DATE        0
LAST_OCCURRENCE_DATE    319926
REPORTED_DATE                0
INCIDENT_ADDRESS         42113
GEO_X                     4184
GEO_Y                     4184
GEO_LON                   4185
GEO_LAT                   4185
DISTRICT_ID                  1
PRECINCT_ID                  1
NEIGHBORHOOD_ID              1
IS_CRIME                     0
IS_TRAFFIC                   0
dtype: int64
```

Рисунок 9. Результат работы функции df.isnull().sum(axis = 0)

7. Сколько объектов с пропусками?

Как видно из результата работы функции isnull().sum(axis = 0), в датасете есть пропуски в *LAST_OCCURRENCE_DATE(319926)*, *INCIDENT_ADDRESS(42113)*, *GEO_X* и *GEO_Y*(4184), *GEO_LON* и *GEO_LAT*(4185), *DISTRICT_ID*, *PRECINCT_ID* и *NEIGHBORHOOD_ID*(1)

8. Сколько признаков достаточно для объяснения 90% дисперсии после применения метода PCA?

Стандартизируем данные:

```
from sklearn import preprocessing
stand_X = pd.DataFrame(preprocessing.scale(x_train), columns = x_train.columns)
```

Применим PCA так, чтобы он объяснил 90% дисперсии:

```
from sklearn.decomposition import PCA
for i in range(1, len(stand_X.columns)):
    pca = PCA(n_components=i)
    pca.fit(stand_X)
    print(i, sum(pca.explained_variance_ratio_))
```

```
1 0.21045094721963134
2 0.3853202730611467
3 0.4910738303035732
4 0.592028624894279
5 0.6525318402724691
6 0.7100709120953597
7 0.7634204888703511
8 0.8160730063309259
9 0.8678143956151481
10 0.9173963560079272
11 0.9595187543341178
12 0.9832948722246759
13 0.9992347739094721
14 0.9998188576054453
15 0.999902849760987
16 0.9999719035575223
17 0.9999999860174998
18 1.0
```

Рисунок 10. Результат работы команды print(i, sum(pca.explained_variance_ratio_))

Из рисунка 10 видим, что для объяснения 90% дисперсии достаточно 10 признаков.

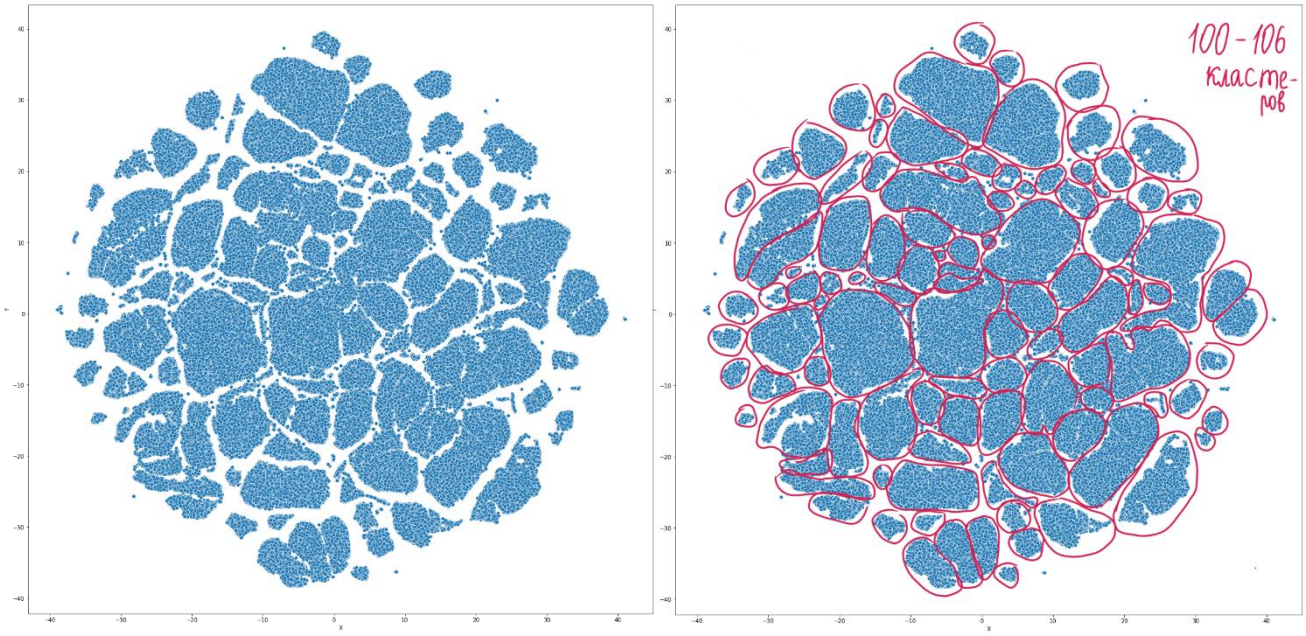
* Доп. задание: построить двухмерное представление данных с помощью алгоритма t-SNE. На сколько кластеров визуально на ваш взгляд разделяется выборка?

Выполним код:

```
less_dimensional_X = pca.transform(stand_X)
from sklearn.manifold import TSNE
tsne = TSNE(n_components = 2, random_state = 0)
tsne_results = tsne.fit_transform(less_dimensional_X)
```

Визуализируем кластеры:

```
tsne_df = pd.DataFrame({'X':tsne_results[:,0], 'Y':tsne_results[:,1], 'real_ans':y_train})
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 20))
sns.scatterplot(x="X", y="Y", data=tsne_df);
```



Рисунки 11-12. Результат работы команды sns.scatterplot(x="X", y="Y", data=tsne_df)

Визуально можно выделить от 100 до 106 кластеров.

Полный код решения задачи приведён в Приложении 6.

Выводы

Проанализировав наш набор данных, мы можем заключить, что в датасете 7 категориальных признаков, 2 бинарных и 6 числовых. В наборе есть пропуски в *LAST_OCCURRENCE_DATE* (319926), *INCIDENT_ADDRESS* (42113), *GEO_X* и *GEO_Y* (по 4184), *GEO_LON* и *GEO_LAT* (по 4185), *DISTRICT_ID*, *PRECINCT_ID* и NEIGHBORHOOD_ID (по 1). Для описания 90% дисперсии хватает 10 признаков. По результатам кластеризации можно явно выделить от 100 до 106 кластеров.

Практическая часть (2)

Набор данных: https://www.kaggle.com/paultimothymooney/denver-crime-data

1. Выделить районы с наибольшим или наименьшим количеством преступлений
2. Выделить наиболее популярные преступления в каждом районе

Решение

1. Выделить районы с наибольшим или наименьшим количеством преступлений.

Считываем данные из файлов, как и в работе 5:
```
df = pd.read_csv('C:\\RLib\\crime.csv')
```

Снова исключим столбцы с большим количеством пропусков, а также строки с пропусками:
```
df = df.drop(['INCIDENT_ADDRESS', 'LAST_OCCURRENCE_DATE', 'FIRST_OCCURRENCE_DATE'], axis=1)
df = df.dropna()
```

Затем преобразуем столбец *REPORTED_DATE* и разобьем его на стобцы *year, month, day, hour* и *minute,* а затем уберем столбец *REPORTED_DATE* из таблицы:

```
df['REPORTED_DATE']=df.REPORTED_DATE.apply(lambda x:datetime.datetime.strptime(x,'%m/%d/%Y %I:%M:%S %p'))
df['year']=df.REPORTED_DATE.apply(lambda x:x.strftime('%Y')).astype(int)
df['month']=df.REPORTED_DATE.apply(lambda x:x.strftime('%m')).astype(int)
df['day']=df.REPORTED_DATE.apply(lambda x:x.strftime('%d')).astype(int)
df['hour']=df.REPORTED_DATE.apply(lambda x:x.strftime('%H')).astype(int)
df['minute']=df.REPORTED_DATE.apply(lambda x:x.strftime('%M')).astype(int)
df = df.drop(['REPORTED_DATE'], axis=1)
```

Нормируем числовые признаки:
```
numeric = ['INCIDENT_ID', 'OFFENSE_ID', 'GEO_X', 'GEO_Y', 'GEO_LON', 'GEO_LAT']
for column in numeric:
    df[column] = (df[column] - df[column].mean()) / (math.sqrt(df[column].var()))
```

Найдем количество различных районов и видов преступлений:
```
print('NEIGHBORHOOD_ID', df['NEIGHBORHOOD_ID'].nunique())
print('OFFENSE_TYPE_ID', df['OFFENSE_TYPE_ID'].nunique())
```

По результатам выполнения кода видим, что различных районов 79, а видов преступлений – 192.

Преобразуем текстовые значения в числовые, сохраняя при этом отдельно названия районов и видов преступлений в массивы neighborhood_id и offense_type_id соответственно:

```
cur_set = set(df['OFFENSE_CATEGORY_ID'])
i = 0
for item in cur_set:
    df['OFFENSE_CATEGORY_ID'] = df['OFFENSE_CATEGORY_ID'].replace(item, i)
    i = i + 1

cur_set = set(df['OFFENSE_TYPE_ID'])
```

```
offense_type_id = []
i = 0
for item in cur_set:
    offense_type_id.append(item)
    df['OFFENSE_TYPE_ID'] = df['OFFENSE_TYPE_ID'].replace(item, i)
    i += 1

cur_set = set(df['NEIGHBORHOOD_ID'])
neighborhood_id = []
i = 0
for item in cur_set:
    neighborhood_id.append(item)
    df['NEIGHBORHOOD_ID'] = df['NEIGHBORHOOD_ID'].replace(item, i)
    i += 1
```

Всего различных районов 79, посчитаем количество преступлений для каждого из них:

```
count_crimes = [0] * 79
for index, row in df.iterrows():
    if df['IS_CRIME'][index]:
        count_crimes[df['NEIGHBORHOOD_ID'][index]] += 1
```

По результатам подсчетов найдем район с минимальным и район с максимальным количеством преступлений:

```
min_ind = 0
max_ind = 0
for i in range(79):
    if count_crimes[i] < count_crimes[min_ind]:
        min_ind = i
    if count_crimes[i] > count_crimes[max_ind]:
        max_ind = i

print("Район с минимальным количеством преступлений: ", f"{neighborhood_id[min_ind]} - {count_crimes[min_ind]}")
print("Район с максимальным количеством преступлений: ", f"{neighborhood_id[max_ind]} - {count_crimes[max_ind]}")
```

```
Район с минимальным количеством преступлений:  wellshire - 395
Район с максимальным количеством преступлений:  five-points - 22108
```

Рисунок 13. Районы с минимальным и максимальным количеством преступлений (и количество преступлений)

2. Выделить наиболее популярные преступления в каждом районе.

Посчитаем, сколько преступлений каждого вида в каждом из районов, а затем, на основе вычислений найдем наиболее популярные в каждом районе преступления:

Рисунок 14. Популярные преступления в каждом районе

Полный код решения задачи приведён в Приложении 7.

Выводы

Проанализировав наш набор данных, мы можем заключить, что в районе wellshire насчитывается минимальное количество преступлений во всем Денвере (их 395), а в районе five-points – максимальное количество преступлений (22108). В каждом районе было выделено самое популярное преступление.

Заключение

При решении задачи мы описали набор данных *Denver Crime Data*. После подготовки набора данных к работе, нами было установлено, что в датасете 7 категориальных признаков, 2 бинарных и 6 числовых. По результатам кластеризации можно легко визуально выделить кластеры.

В результате анализа обработанных данных были выявлены районы с максимальным и минимальным количеством преступлений, а также в каждом районе было найдено наиболее часто совершаемое преступление.

Список литературы

1. **У. Маккинли** Python и анализ данных / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2015 – 482 с.: ил.
2. **Доугерти К.** Введение в эконометрику: Учебник. 3-е изд. / Пер. с англ./Доугерти К. – М.: ИНФРА-М, 2009. – 465 с.
3. **Я.Р. Магнус** Эконометрика. Начальный курс: Учеб. – 6-е изд., перераб. и доп./ Я.Р.Магнус, П.К. Катышев, Пересецкий А.А. – М: Дело, 2004. – 576 с.
4. **С. Рашка** Python и машинное обучение: машинное и глубокое обучение с использованием Python, sci-kit learn и TensorFlow 2, 3-е изд.: Пер. с англ./ С. Рашка, В. Мирджалили – СПб.: ООО «Диалектика», 2020. – 848 с.: ил. – Парал. тит. англ.
5. **Stock, James H.** Introduction to econometrics / James H. Stock, Harvard University, Mark W. Watson, Princeton University. ISBN 978-0-13- 348687-2—ISBN 0-13-348687-7
6. **Вербик М.** Путеводитель по современной эконометрике. / Вербик М. – М.: Научная книга, 2006.

Приложение 1

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('C:\\RLib\\crime.csv')
df

categorical = ['OFFENSE_CODE', 'OFFENSE_CODE_EXTENSION', 'OFFENSE_TYPE_ID',
'OFFENSE_CATEGORY_ID',
'DISTRICT_ID', 'PRECINCT_ID', 'NEIGHBORHOOD_ID']

for column in df.columns:
    if column in categorical:
        print(column, df[column].nunique())

i = 1
for column in df.columns:
    if df[column].nunique() == 2:
        print(i, column)
        i += 1

i = 1
for column in df.columns:
    if df[column].nunique() > 200 and column != 'INCIDENT_ADDRESS' and column !=
'FIRST_OCCURRENCE_DATE' and column != 'LAST_OCCURRENCE_DATE' and column !=
'REPORTED_DATE':
        print(i, column, df[column].nunique())
        i += 1

df.isnull().sum(axis = 0)

df = df.drop(['INCIDENT_ADDRESS', 'LAST_OCCURRENCE_DATE',
'FIRST_OCCURRENCE_DATE'], axis=1)
df = df.dropna()

for column in df.columns:
    print(df[column].value_counts())

import datetime
df['REPORTED_DATE']=df.REPORTED_DATE.apply(lambda
x:datetime.datetime.strptime(x,'%m/%d/%Y %I:%M:%S %p'))
df['year']=df.REPORTED_DATE.apply(lambda x:x.strftime('%Y')).astype(int)
df['month']=df.REPORTED_DATE.apply(lambda x:x.strftime('%m')).astype(int)
df['day']=df.REPORTED_DATE.apply(lambda x:x.strftime('%d')).astype(int)
df['hour']=df.REPORTED_DATE.apply(lambda x:x.strftime('%H')).astype(int)
df['minute']=df.REPORTED_DATE.apply(lambda x:x.strftime('%M')).astype(int)
df.head()
```

```
import math
numeric = ['INCIDENT_ID', 'OFFENSE_ID', 'GEO_X', 'GEO_Y', 'GEO_LON', 'GEO_LAT']
for column in numeric:
    df[column] = (df[column] - df[column].mean()) / (math.sqrt(df[column].var()))

text = ['OFFENSE_TYPE_ID', 'OFFENSE_CATEGORY_ID', 'NEIGHBORHOOD_ID']
for column in text:
    cur_set = set(df[column])
    i = 0
    for item in cur_set:
        df[column] = df[column].replace(item, i)
        i = i + 1

max_average = -math.inf
max_name = ''
for column in numeric:
    if df[column].mean() > max_average:
        max_name = column
        max_average = df[column].mean()
print(f"{max_average} - {max_name}")

df_train = df.drop(['NEIGHBORHOOD_ID', 'REPORTED_DATE'], axis=1)
df_test = df['NEIGHBORHOOD_ID']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df_train, df_test, test_size=.3, random_state=42)
x_train

from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
vif_data["column_name"] = df_train.columns
vif_data["VIF"] = [variance_inflation_factor(df_train.values, i) for i in
range(len(df_train.columns))]

vif_data.loc[vif_data['VIF'] >= 10.0]

from sklearn import preprocessing
stand_X = pd.DataFrame(preprocessing.scale(x_train), columns = x_train.columns)

from sklearn.decomposition import PCA
for i in range(1, len(stand_X.columns)):
    pca = PCA(n_components=i)
    pca.fit(stand_X)
    print(i, sum(pca.explained_variance_ratio_))

pca = PCA(n_components=10)
x = pca.fit(stand_X)

first_component = pca.components_[0]
greatest_contribution_i = 0
greatest_contribution = first_component[greatest_contribution_i]
```

```
for i in range(0, len(first_component)):
    if  abs(first_component[i]) >  abs(greatest_contribution):
        greatest_contribution = first_component[i]
        greatest_contribution_i = i
print(greatest_contribution_i, greatest_contribution)

stand_X.iloc[:, 6]

less_dimensional_X = pca.transform(stand_X)

from sklearn.manifold import TSNE
tsne = TSNE(n_components = 2, random_state = 0)
tsne_results = tsne.fit_transform(less_dimensional_X)

tsne_df = pd.DataFrame({'X':tsne_results[:,0], 'Y':tsne_results[:,1], 'real_ans':y_train})
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 20))
sns.scatterplot(x="X", y="Y", data=tsne_df);
```

Приложение 2

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv('C:\\RLib\\crime.csv')

df = df.drop(['INCIDENT_ADDRESS', 'LAST_OCCURRENCE_DATE',
'FIRST_OCCURRENCE_DATE'], axis=1)
df = df.dropna()

import datetime
df['REPORTED_DATE']=df.REPORTED_DATE.apply(lambda
x:datetime.datetime.strptime(x,'%m/%d/%Y %I:%M:%S %p'))
df['year']=df.REPORTED_DATE.apply(lambda x:x.strftime('%Y')).astype(int)
df['month']=df.REPORTED_DATE.apply(lambda x:x.strftime('%m')).astype(int)
df['day']=df.REPORTED_DATE.apply(lambda x:x.strftime('%d')).astype(int)
df['hour']=df.REPORTED_DATE.apply(lambda x:x.strftime('%H')).astype(int)
df['minute']=df.REPORTED_DATE.apply(lambda x:x.strftime('%M')).astype(int)

df = df.drop(['REPORTED_DATE'], axis=1)

import math
numeric = ['INCIDENT_ID', 'OFFENSE_ID', 'GEO_X', 'GEO_Y', 'GEO_LON', 'GEO_LAT']
for column in numeric:
    df[column] = (df[column] - df[column].mean()) / (math.sqrt(df[column].var()))

print('NEIGHBORHOOD_ID', df['NEIGHBORHOOD_ID'].nunique())
print('OFFENSE_TYPE_ID', df['OFFENSE_TYPE_ID'].nunique())

cur_set = set(df['OFFENSE_CATEGORY_ID'])
i = 0
for item in cur_set:
    df['OFFENSE_CATEGORY_ID'] = df['OFFENSE_CATEGORY_ID'].replace(item, i)
    i = i + 1

cur_set = set(df['OFFENSE_TYPE_ID'])
offense_type_id = []
i = 0
for item in cur_set:
    offense_type_id.append(item)
    df['OFFENSE_TYPE_ID'] = df['OFFENSE_TYPE_ID'].replace(item, i)
    i += 1

cur_set = set(df['NEIGHBORHOOD_ID'])
neighborhood_id = []
i = 0
for item in cur_set:
    neighborhood_id.append(item)
    df['NEIGHBORHOOD_ID'] = df['NEIGHBORHOOD_ID'].replace(item, i)
```

```
    i += 1

count_crimes = [0] * 79
for index, row in df.iterrows():
    if df['IS_CRIME'][index]:
        count_crimes[df['NEIGHBORHOOD_ID'][index]] += 1

print(count_crimes)

min_ind = 0
max_ind = 0
for i in range(79):
    if count_crimes[i] < count_crimes[min_ind]:
        min_ind = i
    if count_crimes[i] > count_crimes[max_ind]:
        max_ind = i

print("Район с минимальным количеством преступлений: ", f"{neighborhood_id[min_ind]} -
{count_crimes[min_ind]}")
print("Район с максимальным количеством преступлений: ", f"{neighborhood_id[max_ind]} -
{count_crimes[max_ind]}")

crimes_in_neighborhood = [0] * 79
for i in range(79):
    crimes_in_neighborhood[i] = [0] * 192

popular_crimes = [-1] * 79

for index, row in df.iterrows():
    if df['IS_CRIME'][index]:
        curr_neighborhood = df['NEIGHBORHOOD_ID'][index]
        curr_crime_type = df['OFFENSE_TYPE_ID'][index]
        crimes_in_neighborhood[curr_neighborhood][curr_crime_type] += 1
        if popular_crimes[curr_neighborhood] == -1 or
crimes_in_neighborhood[curr_neighborhood][popular_crimes[curr_neighborhood]] <
crimes_in_neighborhood[curr_neighborhood][curr_crime_type]:
            popular_crimes[curr_neighborhood] = curr_crime_type

print("Наиболее популярные преступления по каждому из районов:")
for i in range(79):
    print(f"{neighborhood_id[i]} - {offense_type_id[popular_crimes[i]]}")

df_train = df[['OFFENSE_TYPE_ID']]
df_test = df['NEIGHBORHOOD_ID']

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df_train, df_test, test_size=.3, random_state=42)

from sklearn import preprocessing
stand_X = pd.DataFrame(preprocessing.scale(x_train), columns = x_train.columns)

from sklearn.decomposition import PCA
```

```
pca = PCA()
pca.fit(stand_X)
less_dimensional_X = pca.transform(stand_X)

from sklearn.manifold import TSNE
tsne = TSNE(n_components = 2, random_state = 0)
tsne_results = tsne.fit_transform(less_dimensional_X)

tsne_df = pd.DataFrame({'X':tsne_results[:,0], 'Y':tsne_results[:,1], 'real_ans':y_train})
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 10))
sns.scatterplot(x="X", y="Y", data=tsne_df);

x = df.loc[:, df.columns.isin(['NEIGHBORHOOD_ID', 'hour', 'minute'])]
y = df.loc[:, df.columns.isin(['OFFENSE_TYPE_ID'])]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=.33, random_state=10)

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
param_grid = { 'n_estimators': [50, 100, 150],'max_features': ['auto'],'max_depth' : list(range(1,
10)), 'criterion' :['gini']}
RFC = GridSearchCV(estimator=RandomForestClassifier(), param_grid=param_grid, cv=5,
refit=True)
RFC.fit(x_train, y_train)

from sklearn.model_selection import cross_val_score
print("f1:"+str(np.average(cross_val_score(RFC.best_estimator_, x_test, y_test,
scoring='f1_macro'))))
print("precision:"+str(np.average(cross_val_score(RFC.best_estimator_, x_test, y_test,
scoring='precision_macro'))))
print("recall:"+str(np.average(cross_val_score(RFC.best_estimator_, x_test, y_test,
scoring='recall_macro'))))

model = LinearRegression().fit(x, y)
print ("R^2 =", model.score(x, y))
```