

# Report of Data Science for DC Crime

## 0. 小组成员

Name	Student ID	Task 2	Task 3	Task 4
赵钊	12110120	✓	✓	
杨皓翔	12112523	✓		✓

## 1. 背景

在现代社会，犯罪问题是一个普遍存在的社会现象，它不仅关系到经济、文化、政治和技术的发展，更直接影响到人民的幸福感和社会的稳定。华盛顿特区（DC）作为美国首都，其犯罪数据的分析对于理解犯罪模式、预防犯罪行为以及提升公共安全具有重要意义。本项目旨在通过深入分析2008年至2017年DC地区的犯罪数据，运用数据挖掘技术揭示其中的潜在信息，并探索犯罪与住房价格之间的关系，以期为政策制定者和社会规划者提供数据支持和决策参考。

随着大数据技术的发展，我们现在有能力处理和分析大规模的犯罪数据集，从而更准确地理解犯罪分布的地理和时间特征。本项目将使用 `DC_Crime.csv` 数据集，该数据集包含了犯罪类型、地理位置、时间等多个维度的信息。此外，为了深入分析犯罪与经济行为之间的关系，我们还将结合DC地区的住房数据集 `DC_Properties.csv`，通过地理信息和时间信息的关联，探索犯罪率与房价之间的潜在联系。

本项目的研究内容包括但不限于以下几个方面：

- 数据预处理：对原始数据进行清洗，处理缺失值和异常值，为后续分析打下坚实基础。
- 数据探索与可视化：通过统计和可视化手段，探索数据中的模式和趋势，如犯罪类型、时间分布、地理分布等。
- 属性相关性分析：分析不同犯罪特征之间的相关性，如犯罪时间与犯罪类型之间的关系。
- 地理区域与犯罪数量的关联：通过地理空间分析，研究犯罪事件在不同地区的分布情况。
- 房价预测：结合犯罪数据和住房数据，构建房价预测模型，分析犯罪率对房价的潜在影响。

通过本项目，我们期望能够为理解DC地区的犯罪现象提供新的视角，并为相关领域的研究者和决策者提供有价值的信息和见解。

## 2. 数据预处理

### 2.1 属性分类

`DC_Crime.csv` 中共有 29 个属性，每个属性分别带有一些方面的信息，现在，我们把这 29 个属性分成 3 类，分别为地理相关、时间相关、犯罪相关的属性，分别用 G、T、C 标记。分类的结果如下表：

Attribute	Category	Attribute	Category	Attribute	Category
NEIGHBORHOOD_CLUSTER	G	YEAR	T	START_DATE	T
CENSUS_TRACT	G	offensekey	C	CCN	C

Attribute	Category	Attribute	Category	Attribute	Category
offensegroup	C	BID	G	OFFENSE	C
LONGITUDE	G	sector	G	OCTO_RECORD_ID	C
END_DATE	T	PSA	G	ANC	G
offense-text	C	ucr-rank	C	REPORT_DATE	T
SHIFT	T	BLOCK_GROUP	G	METHOD	C
YBLOCK	G	VOTING_PRECINCT	G	location	G
DISTRICT	G	XBLOCK	G	LATITUDE	G
WARD	G	BLOCK	G		

由于原数据表 `DC_Crime.csv` 中的变量排布较为混杂，因此考虑先按照属性的分类对变量进行分组。使用如下代码将变量划分为 3 组并输出到 3 个文件中，再进行下一步处理。

```
df = pd.read_csv('DC_Crime.csv')

new_order = ['NEIGHBORHOOD_CLUSTER', 'CENSUS_TRACT', 'LONGITUDE', 'YBLOCK',
'DISTRICT', 'WARD', 'BID', 'sector', 'PSA', 'BLOCK_GROUP', 'VOTING_PRECINCT',
'XBLOCK', 'BLOCK', 'ANC', 'location', 'LATITUDE']
df1 = df.reindex(columns=new_order)
df1.to_csv('DC_Crime_G.csv', index=True)

new_order = ['END_DATE', 'SHIFT', 'YEAR', 'START_DATE', 'REPORT_DATE']
df2 = df.reindex(columns=new_order)
df2.to_csv('DC_Crime_T.csv', index=True)

new_order = ['offensegroup', 'offense-text', 'offensekey', 'ucr-rank', 'CCN',
'OFFENSE', 'OCTO_RECORD_ID', 'METHOD']
df3 = df.reindex(columns=new_order)
df3.to_csv('DC_Crime_C.csv', index=True)
```

## 2.2 数据筛选

注意到数据的维数很大，因此需要对数据进行筛选。考虑先对数据间的相关性进行分析，相关性强的数据包含的重复信息较多，对该部分进行筛选。

### 地理信息筛选

首先，删去以下 3 个变量：

- `BID`：该变量的空缺率高达 **83.3%**，并且提供的信息不重要，因此舍弃该变量
- `BLOCK`：该变量是一个字符串，根据理解，大致是街区的具体位置。但由于各个街区的表示方式不同不能形成统一，且字符串过长，难以数字化解析，因此舍弃该变量
- `location`：该变量是一个包含经度和纬度的二维元组，所包含的信息完全与变量 `LONGITUDE` 和 `LATITUDE` 重合，属于冗余变量，因此舍弃该变量

对剩下的变量两两之间计算皮尔逊相关系数。但考虑到部分变量为字符串类型，现对其进行转化，将字符串类型成整数：

- `NEIGHBORHOOD_CLUSTER`：该变量表示 what neighborhood cluster the case belongs to，格式为 cluster + number，例如 'cluster 21'，对该变量的处理是直接提取出 cluster 的编号

- `VOTING_PRECINCT`: 该变量的格式与 `NEIGHBORHOOD_CLUSTER` 类似, 例如 'precinct 75', 处理方式相同, 提取出 precinct 对应的编号即可
- `ANC`: 该变量是一种地理信息, 格式为一位数字加一个大写字母, 如 '5E'。由于 ASCII 码最大值为 256, 因此采用将字符串看作 256 进制, 然后根据对应的 ASCII 码转通过计算换成数字。函数的代码如下:

```
def ANC_convert(s):
    try:
        ascii_val = [ord(char) for char in s]
        return 256 * ascii_val[0] + ascii_val[1]
    except ValueError:
        return 0
```

- `sector`: 该变量的处理方式和 `ANC` 同理, 函数代码如下:

```
def sector_convert(s):
    try:
        ascii_val = [ord(char) for char in s]
        return 256 * 256 * ascii_val[0] + 256 * ascii_val[1] + ascii_val[2]
    except ValueError:
        return 0
```

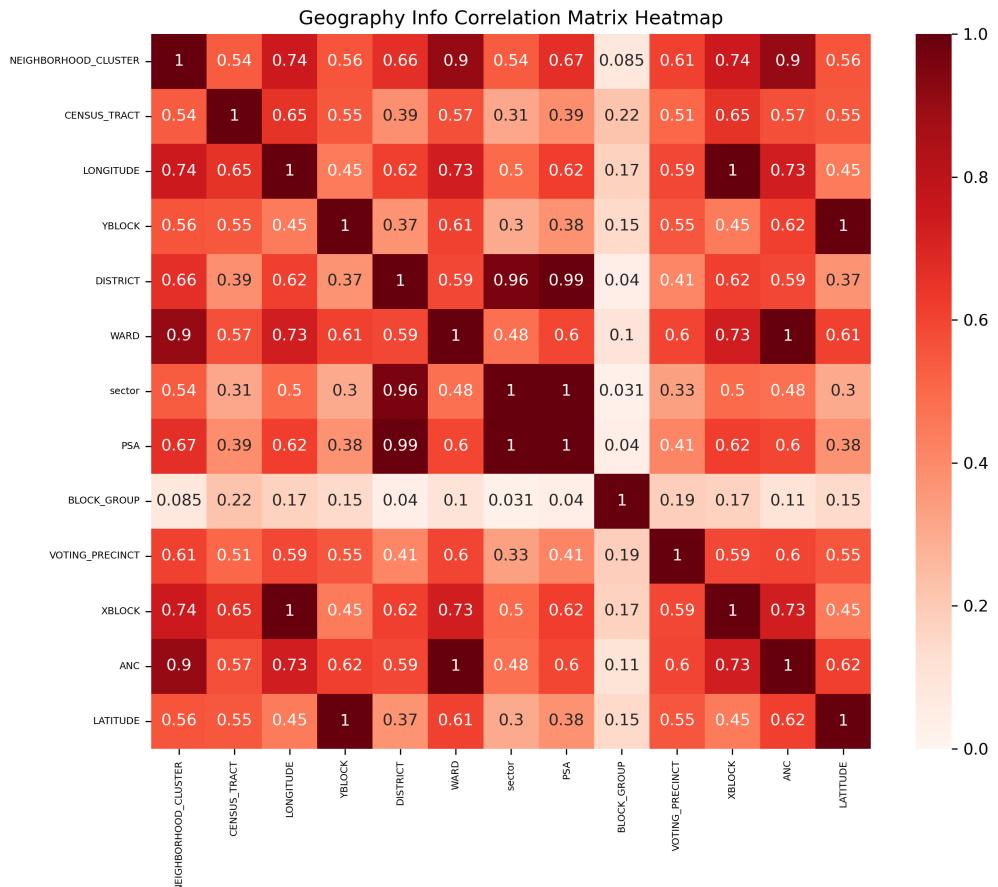
- `BLOCK_GROUP`: 该变量的形式为 'XXXXXX XX', 每个 X 代表一位数字, 其中前 6 位数字与 `CENSUS_TRACT` 属性完全一致, 因此只取出最后两位转化为数字, 即完成了字符串到整数类型的转化

注意, 以上转化时, 会将缺失值直接转化成 0, 但由于缺失值均较少 (最多不超过 2%) , 因此不会对下面计算相关性产生太大的影响。

对 13 个变量进行两两之间的相关性分析, 计算两两之间的皮尔逊相关系数, 核心代码如下:

```
df = pd.read_csv(path)
correlation_matrix = df.corr(method='pearson')
```

得到的结果如下图, 颜色越深代表相关性越强:



我们认为相关系数大于 0.95 为相关性极强，为冗余变量，因此有以下 4 组冗余变量

- `LONGITUDE` 和 `XBLOCK`
- `LATITUDE` 和 `YBLOCK`
- `ANC` 和 `WARD`
- `PSA`, `sector` 和 `DISTRICT`

综合变量类型，变量含义，包含信息量与分析的难易度，分别选择 `LONGITUDE`, `LATITUDE`, `ANC`, `PSA` 作为代表变量，其余作为冗余变量舍弃。

## 时间信息筛选

观察到时间信息有以下特点：

- `END_DATE`, `START_DATE`, `YEAR`, `REPORT_DATE` 存在较大的信息重复
- 每个犯罪记录的 `END_DATE` 和 `START_DATE` 相差很小，也与 `REPORT_DATE` 几乎重合
- `END_DATE` 存在数据缺失，而 `START_DATE` 和 `REPORT_DATE` 没有数据缺失

过于具体的时间没有太大的意义，因此选择 `START_DATE` 和 `SHIFT` 代表时间信息，其余变量冗余舍弃。

## 犯罪信息筛选

从 8 个有关犯罪信息的变量中，可以发现：

- `offense-text` 与 `OFFENSE` 的内容是完全一致的
- `offensekey` 仅为 `offensegroup` 和 `OFFENSE` 的直接拼接，为冗余变量

- `OCTO_RECORD_ID` 为 `CCN` 后面拼接字符串 '-01'，因此为冗余变量

根据以上，选择 `offensegroup`, `OFFENSE`, `ucr-rank`, `CCN`, `METHOD` 作为描述犯罪信息的变量

## 2.3 预处理

### 缺失值处理

统计筛选过后选出的变量对应的缺失率（如下表），发现缺失率均较低，且缺失数据由于数据类型及数据含义等原因不易于填补，因此采用将含有缺失列的数据删去的方式。

Attribute	Missing Rate	Attribute	Missing Rate	Attribute	Missing Rate
NEIGHBORHOOD_CLUSTER	1.22%	VOTING_PRECINCT	0.02%	OFFENSE	0
CENSUS_TRACT	0.27%	ANC	0	METHOD	0
LONGITUDE	0	START_DATE	≈ 0	ucr-rank	0
LATITUDE	0	SHIFT	0	CCN	0
PSA	0.05%	offensegroup	0		

### 原子性拆分

删去缺失值的同时，为了使数据耦合度更低，进行以下处理：

- 考虑到 `ANC` 包含一个数字和一个字母，可能是某种编号，但为了使数据更符合原子性，我们将这一变量解耦合，拆分成 `ANC1` 和 `ANC2`，`ANC1` 为第一位数字，`ANC2` 为第二位字母映射到 0 到 25 的值。
- `START_DATE` 精确到秒，然而只需要年月日信息，将变量解耦合拆分成 `YEAR`, `MONTH`, `DAY` 并转化为整数。

### 哑变量设置

由于 `SHIFT`, `OFFENSE`, `OFFENSE_GROUP`, `METHOD` 均为类别变量，将它们强制编码成数字并没有实际意义，因此采用哑变量的方式来处理这 4 个变量。

进行数据预处理的核心代码如下：

```
df = df.dropna()

df['NEIGHBORHOOD_CLUSTER'] = df['NEIGHBORHOOD_CLUSTER'].astype(str)
df['NEIGHBORHOOD_CLUSTER'] =
df['NEIGHBORHOOD_CLUSTER'].apply(NEIGHBORHOOD_CLUSTER_convert)
df['VOTING_PRECINCT'] = df['VOTING_PRECINCT'].astype(str)
df['VOTING_PRECINCT'] = df['VOTING_PRECINCT'].apply(VOTING_PRECINCT_convert)

df['ANC'] = df['ANC'].astype(str)
df['ANC1'] = df['ANC'].apply(anc1)
df['ANC2'] = df['ANC'].apply(anc2)
df['START_DATE'] = df['START_DATE'].astype(str)
df['YEAR'] = df['START_DATE'].apply(year_cal)
df['MONTH'] = df['START_DATE'].apply(month_cal)
df['DAY'] = df['START_DATE'].apply(day_cal)
```

```
df = pd.get_dummies(df, columns=['SHIFT', 'OFFENSE_GROUP', 'OFFENSE', 'METHOD'])
```

### 3. 属性相关性分析

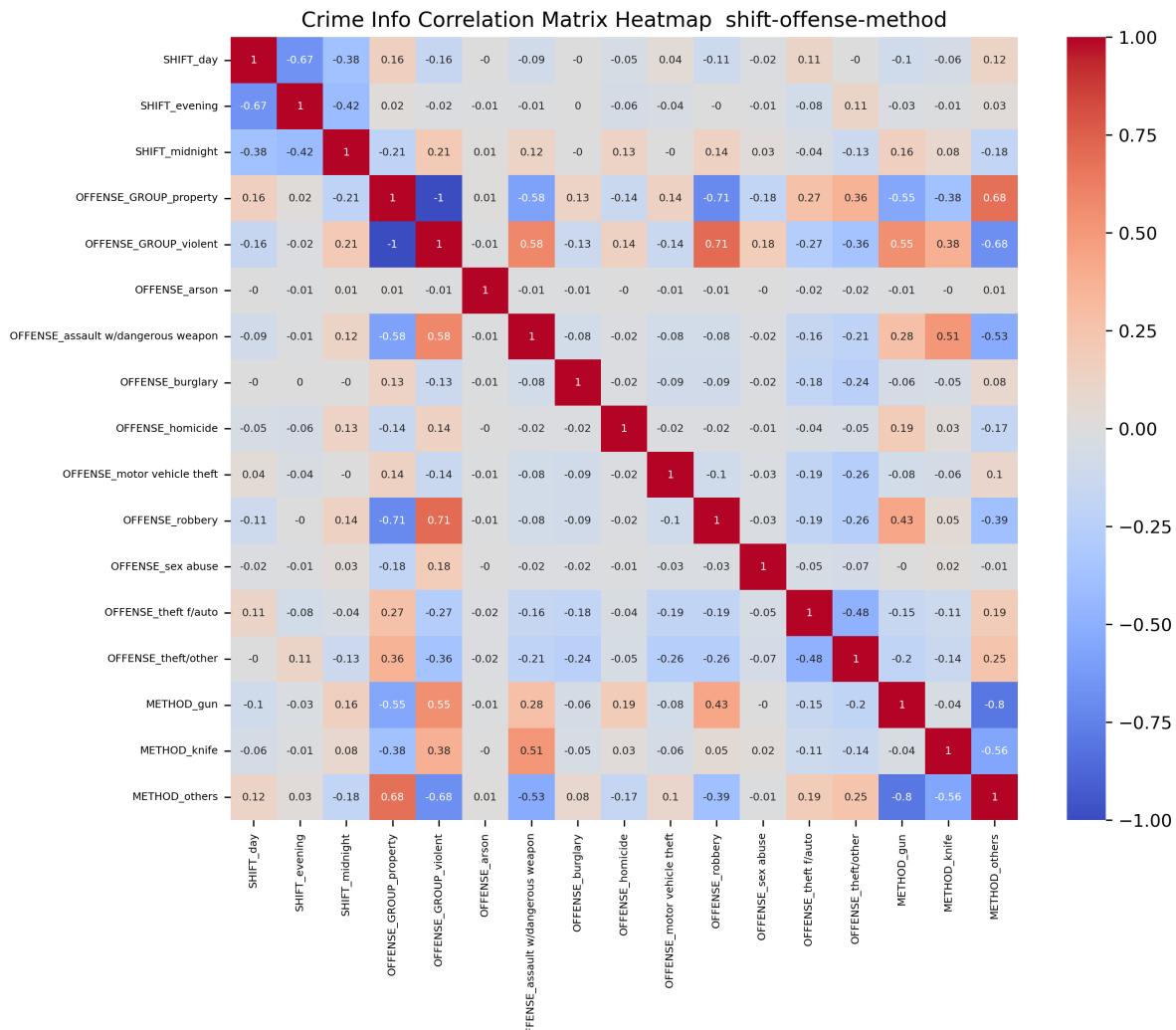
在对 SHIFT, OFFENSE, OFFENSE\_GROUP, METHOD 这四个类别进行热编码处理之后，我们即可对这四个属性进行相关性分析。首先我们直接计算这些属性的相关性矩阵，并绘制热图。

计算与绘制的核心代码如下：

```
index = df.columns.get_loc('SHIFT_day')
CA_df = df.iloc[:, index :]

correlation_matrix = CA_df.corr()
correlation_matrix=correlation_matrix.abs()
correlation_matrix=correlation_matrix.round(2)

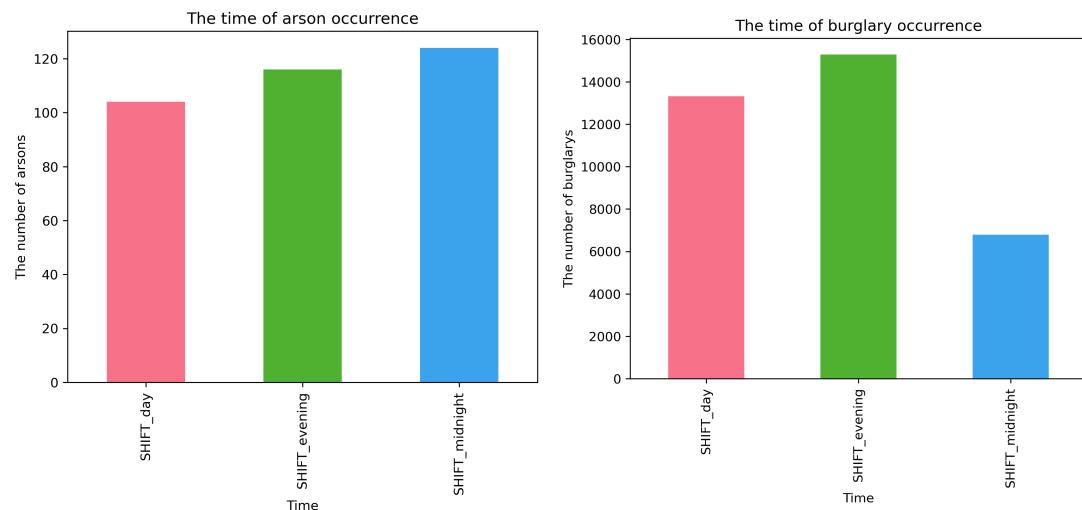
Heatmap=sns.heatmap(correlation_matrix, annot=True, cmap='Reds', vmin=0,
vmax=1,annot_kws={'fontsize': 6})
plt.title('Crime Info Correlation Matrix Heatmap shift-offense-method')
plt.xticks(fontsize=6)
plt.yticks(fontsize=6)
plt.show()
```



得到的结果如上图，由于我们是通过热编码对四类数据进行预处理，四类数据中的每一种可能的取值都会形成新的一列，故我们在观察热图时只针对从不同属性中独立编码得到的数据之间的关系。由于热编码的特点，同一属性中热编码得到的不同分类之间一定是呈负相关的，这与我们热图表现的结果一致。观察热图得到的部分信息如下：

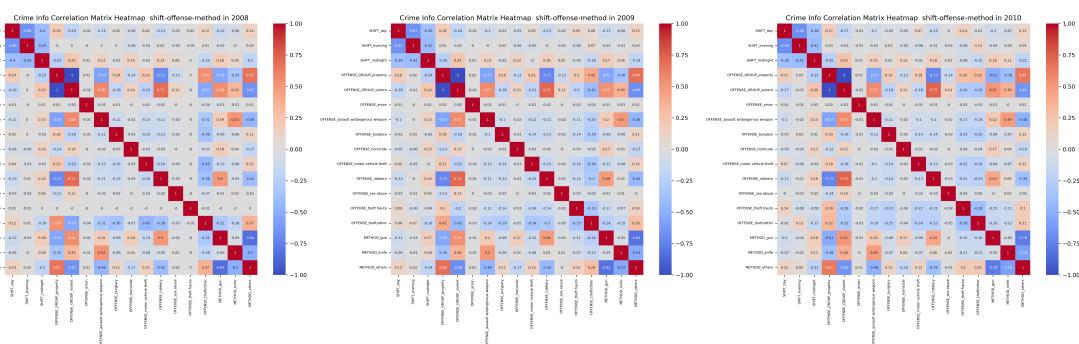
- 暴力犯罪团体的活动集中在夜晚，而财产犯罪团体的活动集中在白天
  - 杀人案件在夜晚更为频繁，其中枪杀的占比要高于刀，而这两者最为常见
  - 暴力犯罪往往采用枪械，刀等攻击性较强的武器，而财产类犯罪则更偏好其他武器
  - 抢劫相较于其他我们通常认识的财产类犯罪拥有更强的暴力属性，甚至超过了谋杀、纵火等我们通常认为的暴力犯罪
  - 盗窃案件与时间的关系并不大

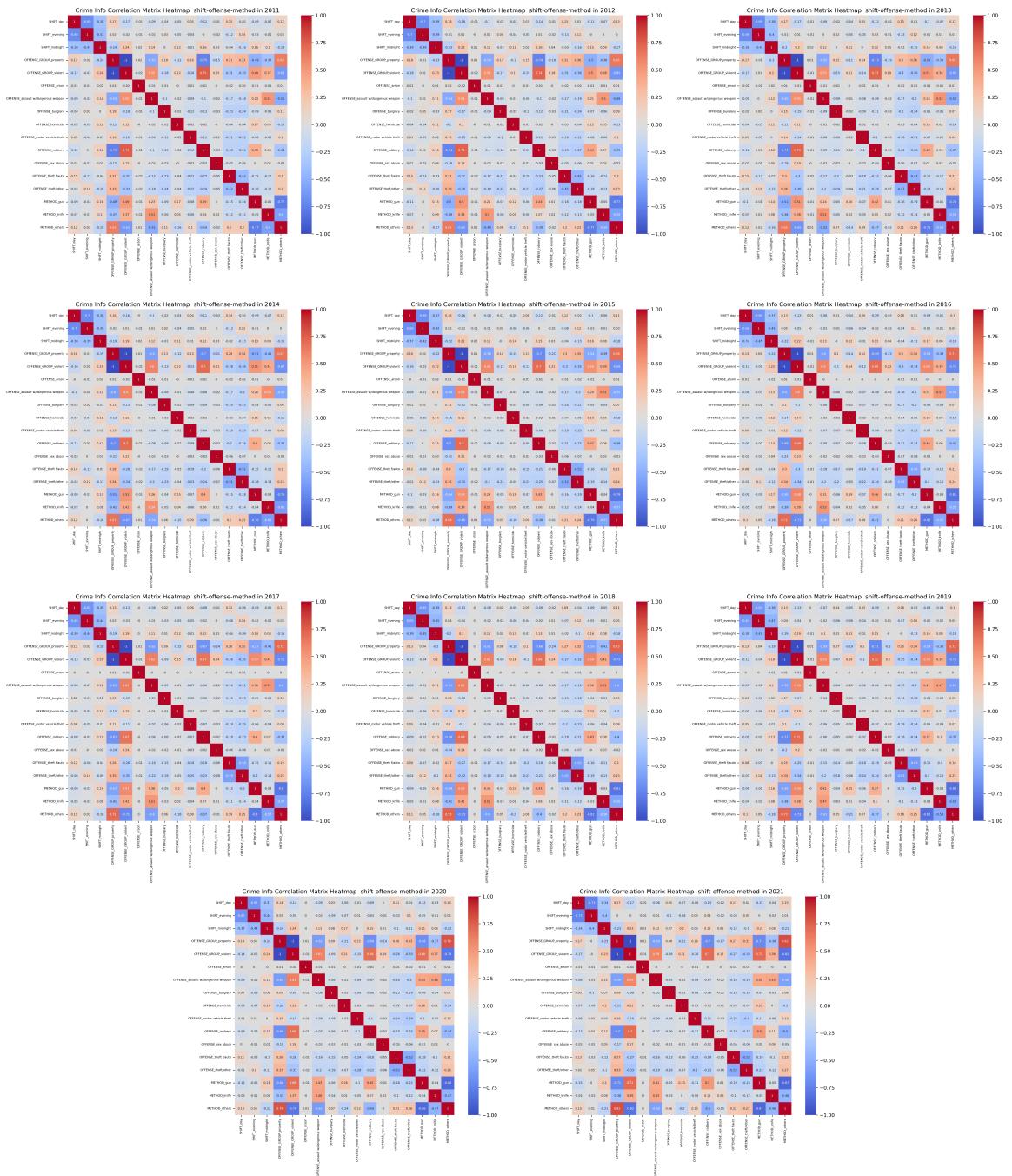
注意到纵火与盗窃相较于时间的相关性系数都很小，我们针对这两种犯罪进行更细致的分析。



上图为纵火案件与盗窃案件发生时段的柱状图，可以看到在 day 和 evening 盗窃事件的数量差不多，而在 midnight 发生的盗窃事件明显少于 day 和 evening。然而在 midnight，总犯罪事件的发生也少于 day 和 evening，故我们得到的相关系数是具有合理性的。而对于纵火案件，我们注意到纵火案件在三种犯罪时间中发生的次数几乎一致，但结合总发生次数，我们可以认为在 midnight 发生纵火案件的占比相较于 day 和 evening 更大。热图中反映的情况可能是因为这类犯罪整体占比比较低，导致对应的所有系数均偏小。

接下来我们分析不同属性之间的相关系数是否会随着年份不同发生变化，我们挑选出不同年份的犯罪信息并计算相关系数矩阵、绘制热图。以下为 2008 到 2021 年共 14 年，每年的相关系数矩阵对应的图像





2008-2021年，大部分属性相关性并无明显变化，但仍然可以观察到

- 总体而言暴力犯罪集团的活动始终在夜晚更为常见，但在2016-2020年其活动的与夜晚的相关性有所下降
- 2008-2020年，抢劫事件与夜晚的相关性几乎逐年增长，可以认为犯罪团体变得更加热衷于在夜晚实施抢劫活动
- 2008-2021年，暴力犯罪团体与枪械的相关性在升高，同时与刀等锐器的相关性呈波动态势，可以认为暴力犯罪团体的武装力量有所上升。

## 4. 犯罪事件与时间、空间关联性分析

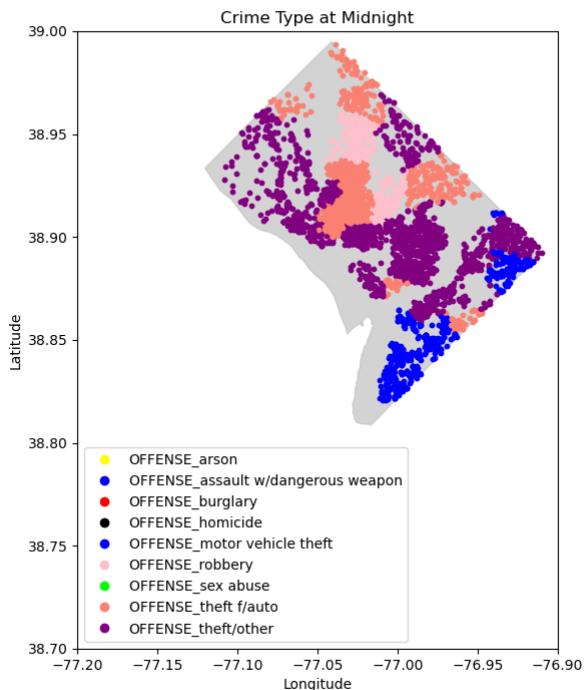
犯罪的时间有3种，分别为 day, evening, midnight；犯罪的事件种类共9种，分别为 arson, assault w/dangerous weapon, burglary, homicide, motor vehicle theft, robbery, sex abuse, theft f/auto, theft/other

下表为 3 个时间的犯罪记录数量，可以看到在 day 和 evening 犯罪事件的数量相差不多，而在 midnight 发生的犯罪事件明显少于 day 和 evening

Shift	day	evening	midnight
Number of Crime	167077	189015	86101

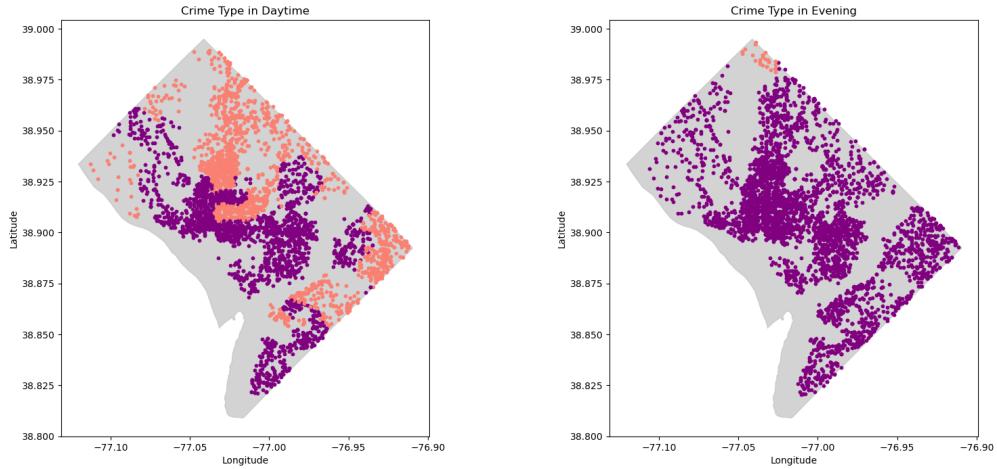
对于 midnight 时段，统计每个 cluster 发生最多的犯罪类型，计算部分的核心代码如下

```
df = pd.read_csv('DC_Crime_Preprocessed.csv')
crime_counts = df.groupby('NEIGHBORHOOD_CLUSTER')[crime_columns].sum()
most_common_crime = crime_counts.apply(lambda x: x.idxmax(), axis=1)
```



得到的结果如上图，观察可得

- 偷窃事件发生的最频繁
- 在华盛顿的中北部抢劫事件也发生较多
- 在华盛顿的东部和南部危险武器袭击事件较多



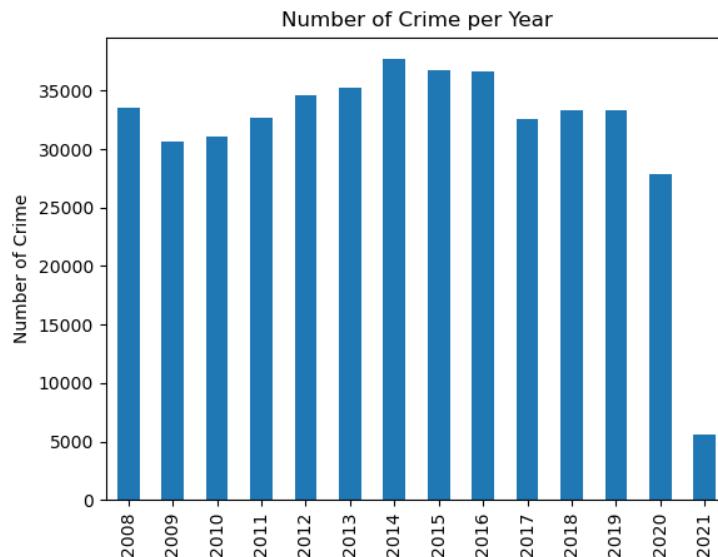
观察 `day` 和 `evening` 时段，容易发现，整个华盛顿 DC 每个 cluster 最常见的犯罪类型都是偷窃，但在晚上偷窃载具较为普遍。

对比 `midnight` 时段，总结并解释现象

- 结合犯罪记录数量，白天的犯罪并没有夜晚多，而白天和夜晚均为偷窃事件最常见
- 白天的载具很多都投入了使用，而夜晚处于闲置，因此夜晚更易发生载具偷窃
- 在午夜，人员活动稀少，因此抢劫和袭击更易发生，与统计数据相吻合

## 5. 犯罪数量与地理区域关联性分析

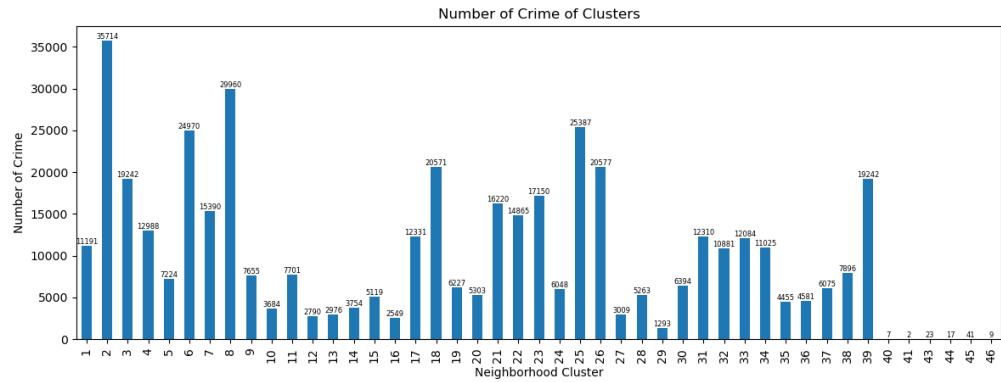
### 5.1 总犯罪数量



上图为华盛顿特区每年所记录的犯罪数量随年份变化的柱状图，有如下几点规律

- 2021 年数量远少于其他年份，原因是数据集只包含 2021 年的前几个月
- 2008 到 2019 年犯罪记录数量有波动，但整体变化不大，基本维持在平均值上下
- 2020 年的犯罪记录数量有较为明显的下降，可能由于治安等方面的提升导致

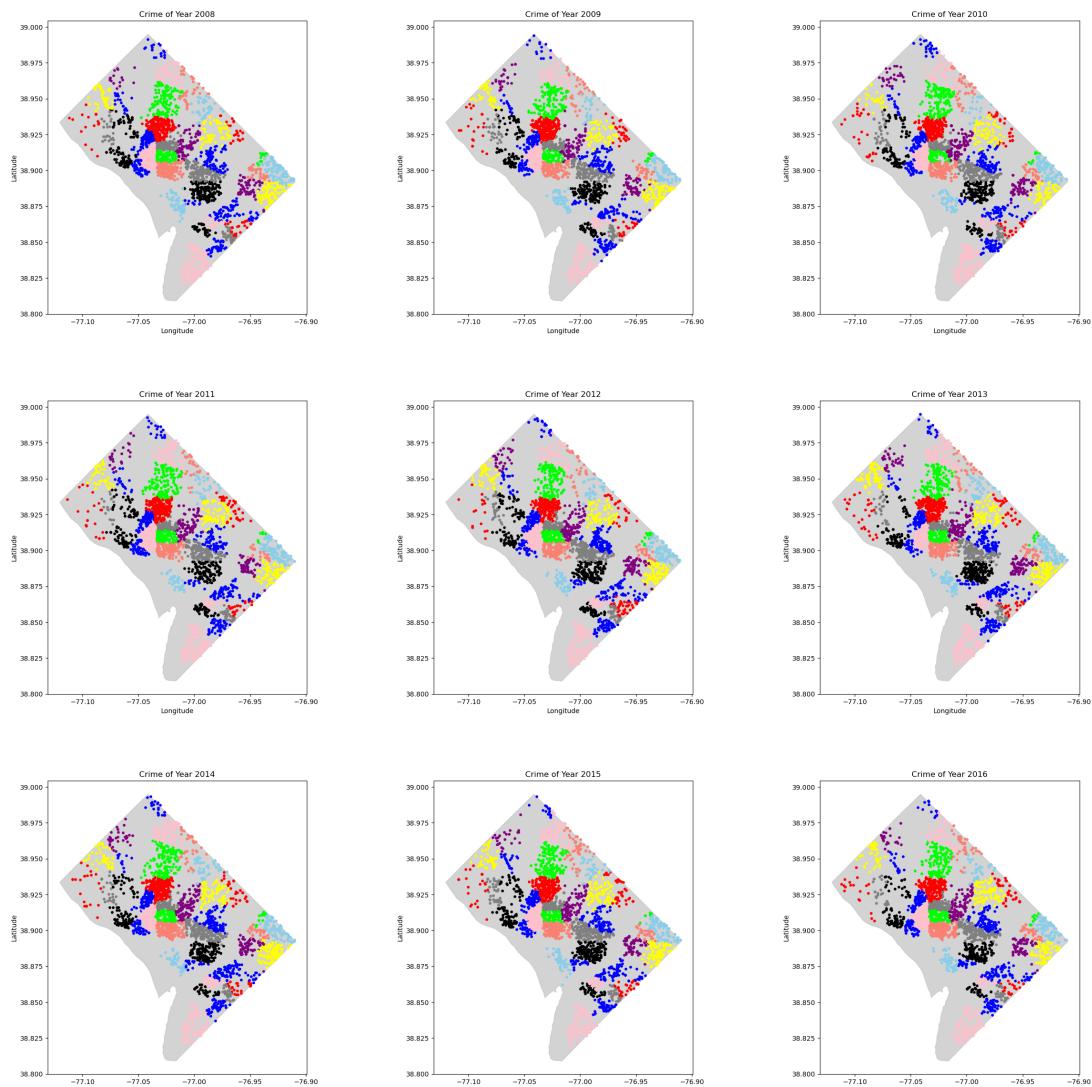
## 5.2 不同街区犯罪数量

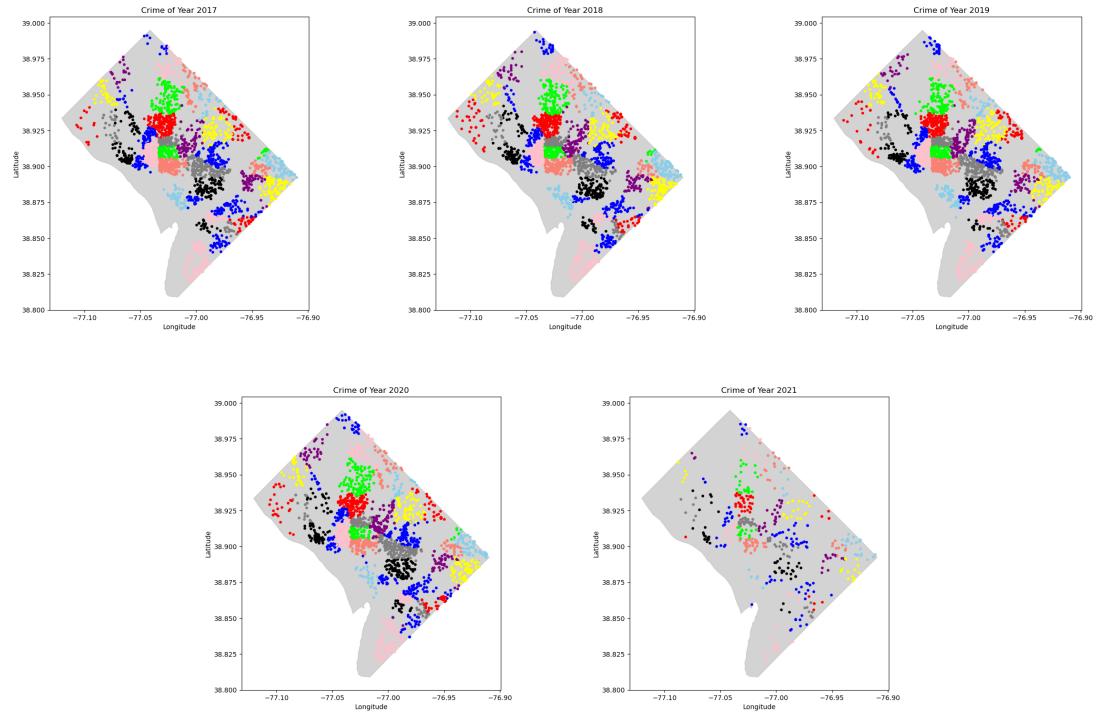


华盛顿特区划分为 46 个 cluster，编号分别为 1 到 46. 上面的柱状图统计了每个 cluster 在 2008 到 2021 年记录的犯罪数量。

- 总体来看，cluster 之间的记录数量方差很大
- cluster 40 到 46 的记录数量可以近似于没有

但由于犯罪数量和 cluster 面积大小也有一定关联，因此我们分析每个 cluster 的犯罪记录密度。以下为 2008 到 2021 年共 14 年，每年各个 cluster 的犯罪密度示意图

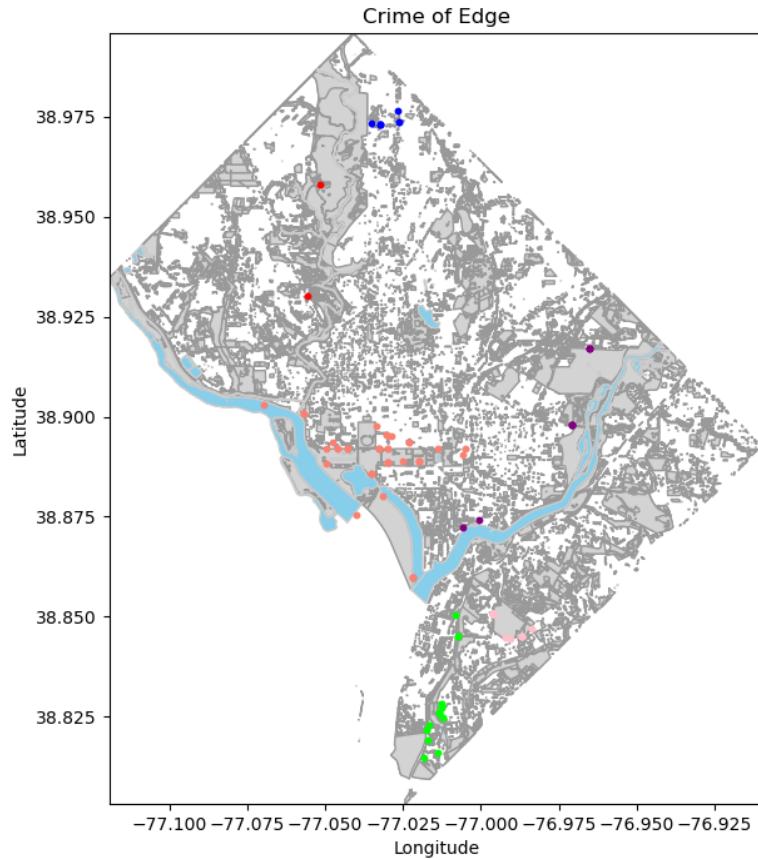




可以观察到

- 犯罪记录多集中在城市中心，犯罪记录密度明显大于四周
- 城市边缘几乎没有记录，而该区域对应的 cluster 编号为 40 到 46

针对编号为 40 到 46 的 cluster 进行深入研究，加入华盛顿 DC 的建筑分布及水域分布，并标记出记录的犯罪事件，得到的结果如下图



- 犯罪多分布在大型建筑或水域周围
- 大型建筑更可能是一些公共设施，如音乐厅，博物馆等，而非居民住宅区
- 发生在这类区域的犯罪数量少于居民区，结果较为符合常理

## 6. 地理区域分类

整个华盛顿共分为 46 个 cluster，而在本任务中，我们将根据犯罪信息对这 46 个 cluster 进行聚类。由于是以犯罪信息为基准进行的聚类，因此首先统计每个 cluster 的犯罪数量，平均危险等级，犯罪时间，犯罪类型和犯罪方式。

筛选处理过后的数据格式如下：

NEIGHBORHOOD_CLUSTER	crime_cnt	ucr_rank_avg	SHIFT_day_rate	SHIFT_evening_rate	SHIFT_midnight_rate	OFFENSE_GROUP_property_rate	OFFENSE_GROUP_violent_rate	METHOD_gun_rate	METHOD_knife_rate	METHOD_others_rate
1	11191	6.074256099	0.3897775	0.383343758	0.226878742	0.868197659	0.131802341	0.027879546	0.019122509	0.952997945
2	35714	5.379755838	0.367278938	0.443187546	0.189533516	0.830570645	0.169429355	0.04373635	0.032564261	0.92369939
3	19242	6.150452136	0.318418044	0.397983578	0.283598379	0.862540276	0.137459724	0.036378755	0.021096978	0.942521567
4	12988	6.073375423	0.342008007	0.556128734	0.101863258	0.948413921	0.051586079	0.008777333	0.006544503	0.984678164
5	7224	5.98961794	0.383582503	0.470791805	0.145625692	0.923172757	0.076827243	0.009689922	0.010797342	0.979512735
6	24970	6.096155386	0.371485783	0.422827393	0.205686824	0.911894273	0.088105727	0.013269555	0.011774129	0.974929916
7	15390	6.157959714	0.354191033	0.445289149	0.200519818	0.87842755	0.12157245	0.03625731	0.019363223	0.944379467
8	29960	6.018457944	0.332209613	0.474465955	0.193324433	0.89305741	0.10694259	0.01805741	0.018598113	0.962082777
9	7655	5.969693011	0.383278903	0.444076682	0.172044415	0.842325278	0.157974722	0.049771391	0.03109079	0.919137818
10	3684	6.380564604	0.496742671	0.426438654	0.076818675	0.957111835	0.042888165	0.013029316	0.004343105	0.982627579
11	7701	6.15084937	0.384573177	0.535385015	0.079041813	0.947798987	0.052201013	0.010518115	0.00771783	0.982710103

在进行聚类前，需对数据进行正则化。由于数据分布较为均匀且有界，因此选取 Z-Score 归一化来进行正则化。

```
df_normalized = (df - df.mean()) / df.std()
```

尝试 K-Means 等聚类算法并进行聚类效果评估。

### 6.1 K-Means

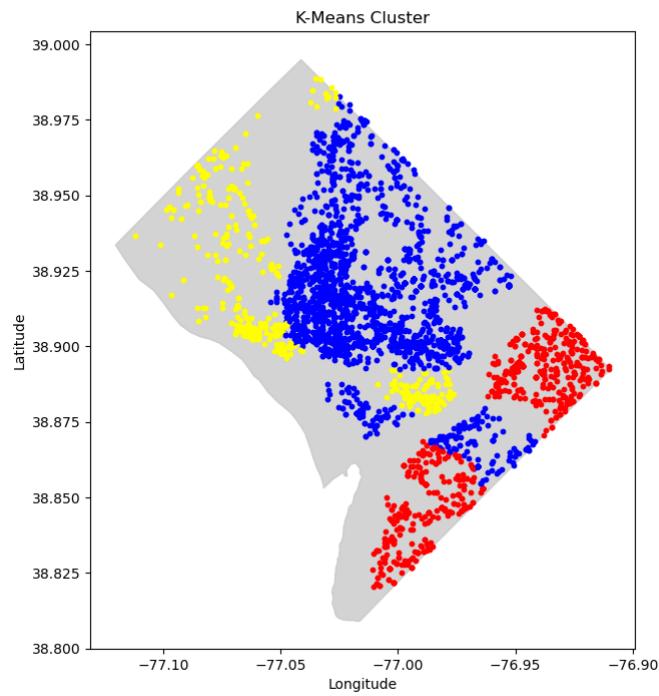
```
features = df.drop('NEIGHBORHOOD_CLUSTER', axis=1)
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features.iloc[:, 1:])

kmeans = KMeans(n_clusters=3)
kmeans.fit(scaled_features)

df['cluster'] = kmeans.labels_
```

观察到，当分类数量大于 3 时，cluster 41 总会被单独聚类为一类，而若仅分为两类，类别数量过少，因此分类参数选择为 3

对分类结果进行可视化得到的结果如下



## 6.2 Hierarchical Clustering

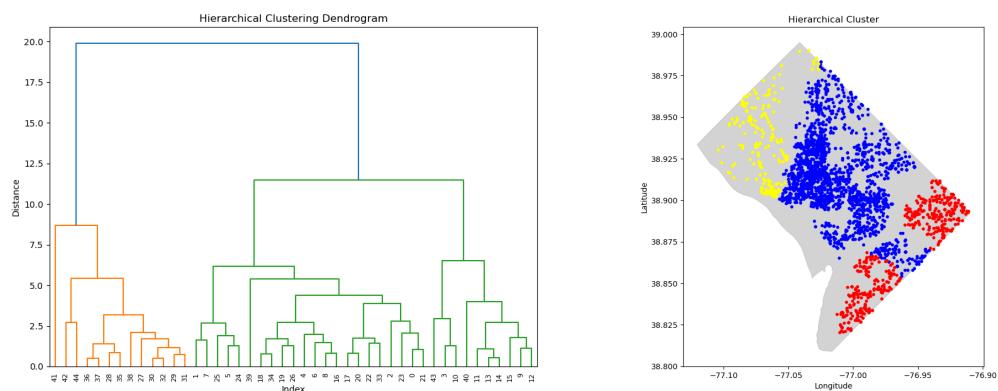
使用以下核心代码进行 Hierarchical Clustering

```
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df_without_first_column)

distance_matrix = pdist(df_scaled)
Z = linkage(distance_matrix, method='ward')

dendrogram(Z)
```

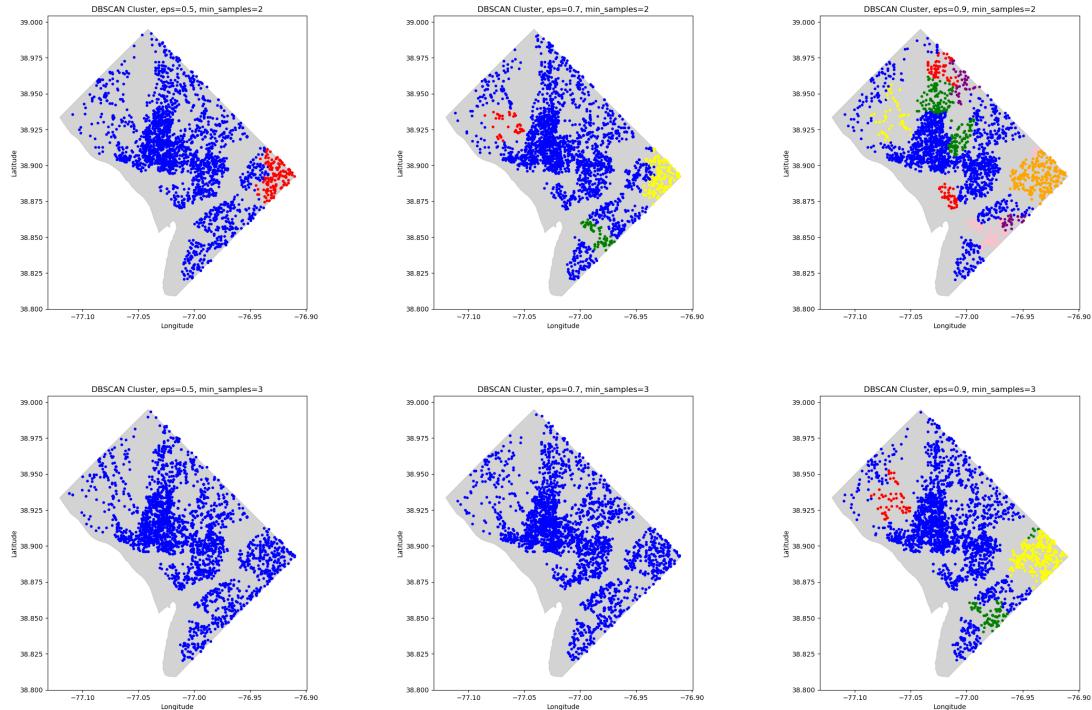
得到树状图如下左图所示。观察可以发现，整体被较为明显的分为 3 类，对结果进行整理并画出地图示意图，如右图



## 6.3 DBSCAN

```
dbscan = DBSCAN(eps=0.7, min_samples=2)  
dbscan.fit(df_scaled)
```

分别测试参数 eps 为 0.5, 0.7, 0.9 和 min\_sample 为 2, 3 时的效果，得到的结果如下

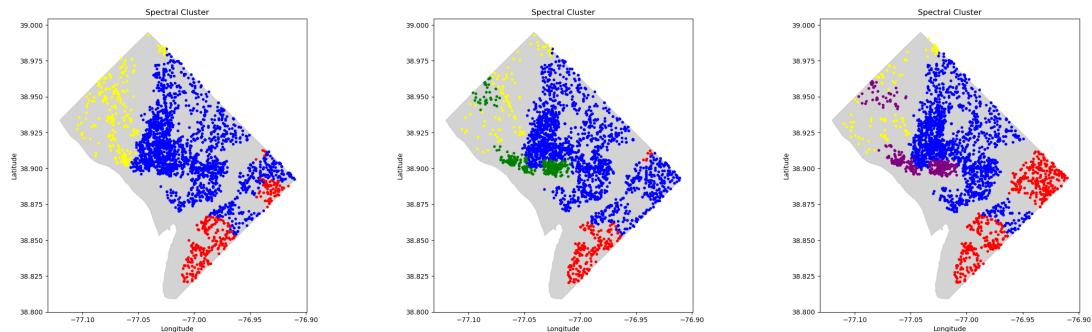


容易观察出，只有  $\text{eps} = 0.9$ ,  $\text{min\_sample} = 2$  时的聚类较为可信，DBSCAN 算法在这个问题上的表现相比于前两种聚类算法较差，容易将大量数据点分为同一类别，而聚类出的其他类别数据点十分稀少，可能原因是数据的密度分布比较不均匀而导致的聚类效果不佳。

## 6.4 Spectral Clustering

```
gamma = 0.1  
affinity_matrix = rbf_kernel(scaled_features, gamma=gamma)  
  
spectral = SpectralClustering(n_clusters=3, affinity='precomputed',  
random_state=42)  
spectral.fit(affinity_matrix)
```

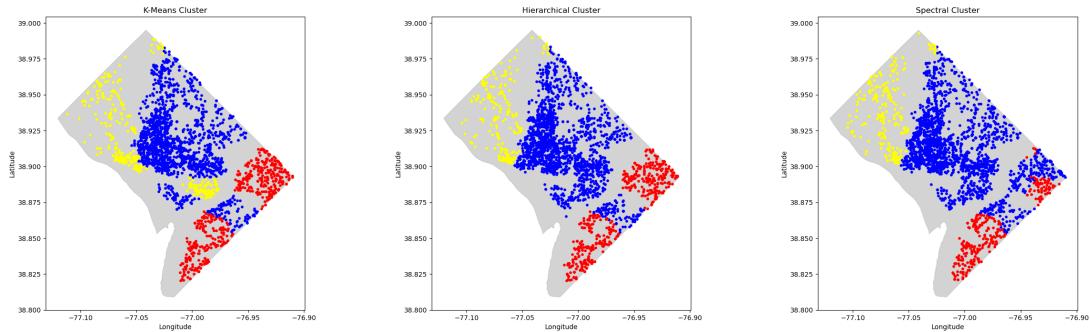
选择聚类类别数量为 3, 4, 5, 结果如下



发现类别数量为 4 和 5 时，得到的结果差异不大，原因是新分类出的类别很小，因此可视化后效果并不明显。

## 6.5 不同聚类方法的对比

对比聚类类别数量为 3 时的 K-Means, Hierarchical Clustering 和 Spectral Clustering 算法，其相似度较强。



聚类结果在地理上分为西北、中部、东南，总结特征如下

聚类类别	犯罪数量	ucr-rank	犯罪时间	犯罪种类	犯罪方式
西北	较少	较高	午夜较少	财产相关较多	使用刀枪较少
中部	较多	中等	午夜较多	财产相关较多	使用刀枪较少
东南	较少	较低	午夜较多	暴力事件较多	使用刀枪较多

- 西北部犯罪数量少但平均危险系数较高，犯罪事件多集中在白天和晚上，犯罪的性质多为财产相关
- 中部犯罪较多但危险系数适中，多发生在午夜，犯罪性质多为财产相关
- 东南部分的主要特征为犯罪数量少且犯罪事件的危险系数 ucr-rank 较低，基本不足 6.0，而更多发生在午夜，犯罪性质更接近使用刀枪较多的暴力事件。但值得一提的是，东南部的犯罪虽然看似更为暴力，发生在午夜且更多使用刀枪，但其危险系数却是最低的。猜测可能与犯罪数量很少，数据不全等问题有关。

## 7. 房价数据预处理

### 7.1 属性分类

`DC_Properties.csv` 中共有 48 个属性，每个属性分别带有一些方面的信息，现在，我们仿照在犯罪属性对数据进行的分类，将这 48 个属性分成 3 类，分别为地理位置相关、时间相关、房屋相关的属性，分别用 G、T、F 标记。分类的结果如下表：

Attribute	Category	Attribute	Category	Attribute	Category
BATHRM	F	BLDG_NUM	F	FULLADDRESS	G
HF_BATHRM	F	STYLE	F	CITY	G
HEAT	F	STRUCT	F	STATE	G
AC	F	GRADE	F	ZIPCODE	G
NUM_UNITS	F	CNDTN	F	NATIONALGRID	G

Attribute	Category	Attribute	Category	Attribute	Category
ROOMS	F	EXTWALL	F	LATITUDE	G
BEDRM	F	ROOF	F	LONGITUDE	G
AYB	T	INTWALL	F	ASSESSMENT_NBHD	G
YR_RMDL	T	KITCHENS	F	ASSESSMENT_SUBNBHD	G
EYB	T	FIREPLACES	F	CENSUS_TRACT	G
STORIES	F	USECODE	F	CENSUS_BLOCK	G
SALEDATE	T	LANDAREA	F	WARD	G
PRICE	F	GIS_LAST_MOD_DTTM	T	SQUARE	G
QUALIFIED	F	SOURCE	F	X	G
SALE_NUM	F	CMPLX_NUM	F	Y	G
GBA	F	LIVING_GBA	F	QUADRANT	G

特别的，由于需要根据房屋属性以及地区所存在的犯罪属性对房价进行预测，在数据处理时，直接删去所有缺失 `PRICE` 这一属性的行。

另外，注意到一个关键的属性：`SOURCE`。该属性表示房屋类型，有两个不同的取值：

- `Residential`：该取值表示这一房屋为住宅，不存在 `CMPLX_NUM` 与 `LIVING_GBA` 等属性。
- `Condominium`：该取值表示这一房屋为公寓，不存在 `STYLE` 等属性。

故我们根据 `SOURCE` 将房屋分两类分别进行属性分类，关键代码如下：

```
df = pd.read_csv('..\..\origin_material\DC_Properties.csv')
df = df.dropna(subset=['PRICE'])

df_residential = df[df['SOURCE'] == 'Residential']

df_condominium = df[df['SOURCE'] == 'Condominium']

new_order = ['FULLADDRESS', 'CITY', 'STATE', 'ZIPCODE', 'NATIONALGRID', 'LONGITUDE',
'ASSESSMENT_NBHD', 'ASSESSMENT_SUBNBHD', 'CENSUS_TRACT', 'CENSUS_BLOCK',
'WARD', 'SQUARE', 'X', 'Y', 'QUADRANT']
df11 = df_condominium.reindex(columns=new_order)
df11.to_csv('..\..\data\task4\DC_Properties_condominium_G.csv', index=True)
df12=df_residential.reindex(columns=new_order)
df12.to_csv('..\..\data\task4\DC_Properties_residential_G.csv', index=True)

new_order = ['AYB', 'YR_RMDL', 'EYB', 'SALEDATE', 'GIS_LAST_MOD_DTTM']
df21 = df_condominium.reindex(columns=new_order)
df21.to_csv('..\..\data\task4\DC_Properties_condominium_T.csv', index=True)
df22=df_residential.reindex(columns=new_order)
df22.to_csv('..\..\data\task4\DC_Properties_residential_T.csv', index=True)

new_order = ['BATHRM', 'HF_BATHRM', 'HEAT', 'AC', 'NUM_UNITS', 'ROOMS', 'BEDRM',
'STORIES', 'PRICE', 'QUALIFIED', 'SALE_NUM',

'GBA', 'BLDG_NUM', 'STYLE', 'STRUCT', 'GRADE', 'CNDTN', 'EXTWALL', 'ROOF', 'INTWALL', 'KITCHENS',
'FIREPLACES', 'USECODE', 'LANDAREA', 'SOURCE', 'CMPLX_NUM', 'LIVING_GBA']
df31 = df_condominium.reindex(columns=new_order)
df31.to_csv('..\..\data\task4\DC_Properties_condominium_F.csv', index=True)
```

```
df32=df_residential.reindex(columns=new_order)
df32.to_csv('..\..\data\task4\DC_Properties_residential_F.csv', index=True)
```

## 7.2 数据预处理

注意到数据的维数很大，且其中包含大量string类型的数据，因此需要对数据进行预处理。首先，为了减小数据的维度，考虑先对数据的含义以及数据间的相关性进行分析，对该部分进行筛选。

### 时间信息筛选

首先，删去以下1个变量：

- `GIS_LAST_MOD_DTTM`：该变量表示数据的最后修改日期，统一为'2018-07-22 18:01:43'，可以认为是无关变量。

对剩下的变量进行转化：

- `SALEDATE`：该变量表示对应房屋最近一次售卖发生的时间，变量的形式为'XXXX/XX/X 0:00'，为了简化问题，只保留其中的`YEAR`对应的属性。

### 空间信息筛选

从与空间有关的16个变量中，可以发现：

- `X` 与 `LONGITUDE`，`Y` 与 `LATITUDE` 虽然取值不同，但是现实意义是完全一样的，都表示房屋的经纬度信息。为减小模型的复杂程度，我们选择删去前两者。
- 对于住宅，所有数据的 `STATE` 与 `CITY` 对应的取值都是一致的，均为 `WASHINGTON`, `DC`，这是因为取样的区域限制决定的，故这部分属性对结果不具有影响。而公寓没这两项变量没有取值。综合考虑，选择直接删去这两个属性。
- 住宅的 `CENSUS_BLOCK` 的前半部分一定为 `CENSUS_TRACT`，因此后者为冗余变量。而公寓的 `CENSUS_BLOCK` 属性空缺，仅存在 `CENSUS_TRACT` 属性。

结合数据处理的难易程度与前述对犯罪地理位置的分析，最终选取 `LONGITUDE`，`LATITUDE`，`WARD` 与 `QUADRANT` 作为需要保留的数据，其中 `WARD` 的取值只需要直接保留数字即可，而 `QUADRANT` 直接选择独热编码取得八个独立的子属性。另外，希望寻找房价与犯罪率之间的关系，可以通过 `LONGITUDE`，`LATITUDE` 这两个属性对将给定房屋的位置通过k-NN算法映射到犯罪率问题中划分的cluster中，并且在给定年份之中，各个cluster犯罪表现差异不大，结合数据处理难度，可以直接忽略年份的影响，通过 cluster整体的的犯罪率来为房价预测增添一个新的属性：`crime background`。这个属性描绘了该房屋所在cluster犯罪的潜在发生率。通过这一属性对最终房价回归的影响，来描述犯罪率对于房价的影响。该映射过程的关键代码如下：

```
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X, y_encoded)

X_C = df_C[['LONGITUDE', 'LATITUDE']]
predicted_clusters = knn.predict(X_C)
X_G = df_G[['LONGITUDE', 'LATITUDE']]
predicted_clusters_G = knn.predict(X_G)

predicted_clusters_decoded = label_encoder.inverse_transform(predicted_clusters)
predicted_clusters_decoded_G =
label_encoder.inverse_transform(predicted_clusters_G)
```

```

df_C['NEIGHBORHOOD_CLUSTER'] = predicted_clusters_decoded
df_G['NEIGHBORHOOD_CLUSTER'] = predicted_clusters_decoded_G

df_C.to_csv('..\..\data\task4\Process_Properties_condominium.csv',
index=True)
df_G.to_csv('..\..\data\task4\Process_Properties_Residential.csv',
index=True)

```

## 房屋属性信息筛选

分别考虑住宅与公寓。首先统计缺失值大于50%的属性，由于这部分数据缺失值较多，难以对预测价格时提供有效的帮助，故选择舍去，最终在住宅中得到23个有效属性，公寓中得到14个有效属性：

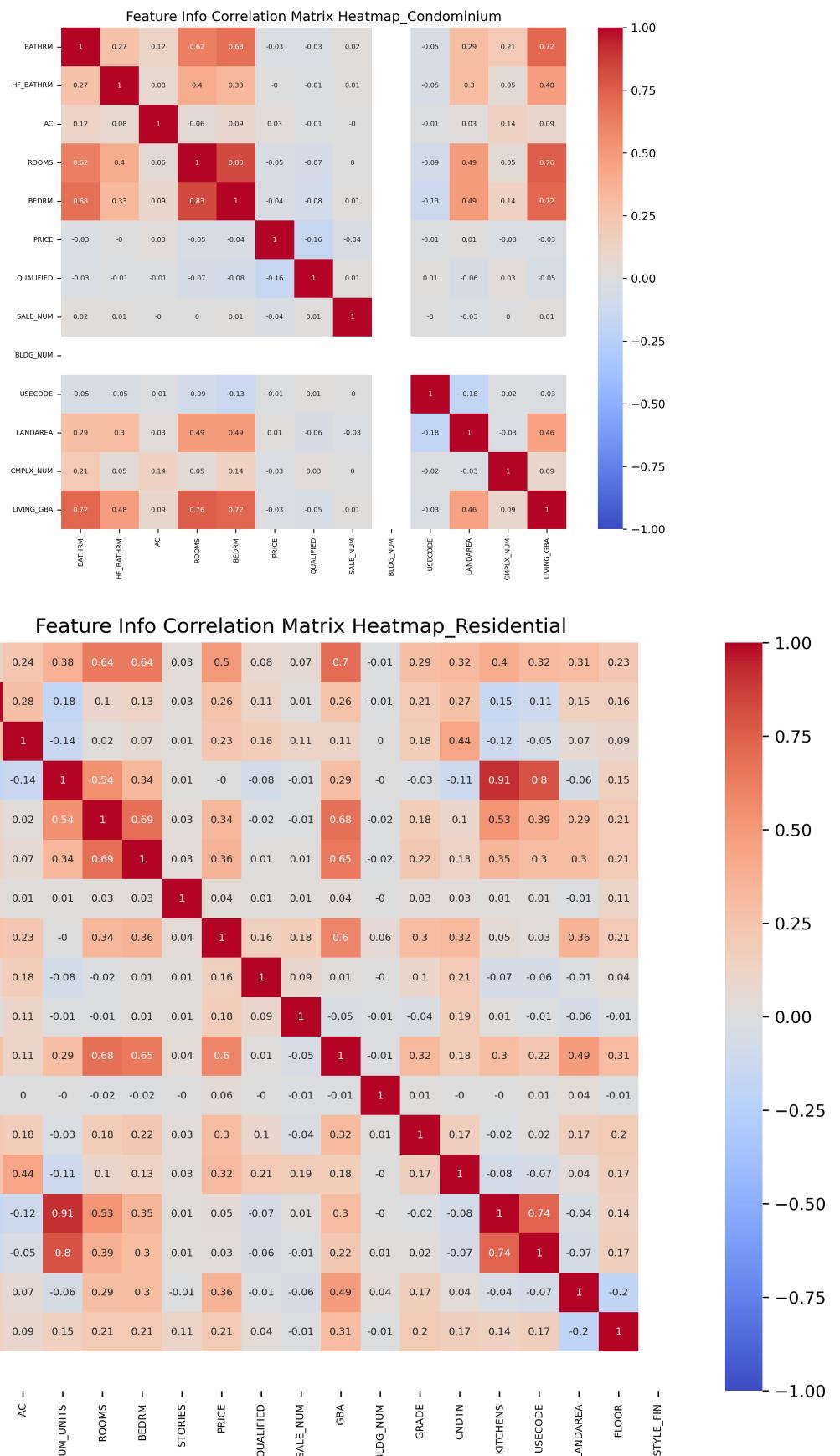
- AC 表示有无制冷，取值为 Y 或 N，可以直接映射为1或0。
- QUALIFIED 表示是否具备资格，取值为 Q 或 U，可以直接映射为1或0。
- STYLE 表示楼层，楼层数若为半层则可能还具有属性 Fin。我们选择提取楼层数据，并通过热编码统计 Fin 的信息。不符合该数据组成规律的数据数量小于千分之一，在此忽略不计。
- HEAT 表示有无供暖，EXTWALL，ROOF，INTWALL 分别表示外墙，天花板以及内墙材料，STRUCT 表示结构，这部分信息种类过于繁多，且难以直接映射处理。为了避免最终的数据维数过大，选择直接舍去。
- GRADE 与 CNDTN 均为房屋评价，分别有13种以及7种不同的评价，可以直接映射到数字，数字越大表示评价越好。对于住宅等级评价，在我们保留的数据中，有共计1.2%的评价为 Exceptional-D，Exceptional-C，Exceptional-A 以及 Exceptional-B，这部分评价含义理解困难，故将其均映射为0，避免其对结果拟合的影响。

这样，便得到了初步处理完毕的数据。再通过计算相关性矩阵制作热图大部分数据处理的方法与第二题中一致，对于变量 Fin 的处理关键代码如下：

```

df32 = pd.read_csv('..\..\data\task4\DC_Properties_residential_F.csv')
df32['FLOOR'] = df32['STYLE'].str.extract('(\d+)')
df32['STYLE_FIN'] = df32['STYLE'].str.replace('Fin', '1', regex=True)
df32['STYLE_FIN'] = df32['STYLE'].str.replace('Unfin', '0.5', regex=True)
df32['STYLE_FIN'] = df32['STYLE'].str.replace(r'^[^\d]', '0', regex=True)
df32 = df32.drop('STYLE', axis=1)

```



得到的结果如上图。该图像中并不存在相关系数大于95%的数据对，因此我们认为并无冗余变量。注意到在分析公寓的属性的相关性时变量 BLDG\_NUM 列的数据出现了空缺，这是因为该列的取值在公寓中均为常数1，与其他变量均无关。最后，将处理结束的几个变量拼接在一起并储存，用以接下来的回归。

## 8. 房屋价格预测

提供的房屋数据共有两种类型，而在本任务中，我们将根据房屋属性以及房屋所在cluster的犯罪信息对房屋价格进行预测。由于我们希望探索犯罪率对房屋价格到底有多大的影响，我们首先通过随机森林方法进行回归。通过不纯度减少的方式计算 `feature_importances_` 属性，并做出图像。关键代码如下：

```
x_C = df_C.drop('PRICE', axis=1)
y_C = df_C['PRICE']

x_Ctrain, x_Ctest, y_Ctrain, y_Ctest = train_test_split(x_C, y_C, test_size=0.2,
random_state=42)

rf = RandomForestRegressor(n_estimators=100, random_state=42)

rf.fit(x_Ctrain, y_Ctrain)

y_Cpred = rf.predict(x_Ctest)

mse_C = mean_squared_error(y_Ctest, y_Cpred)
print(f"Mean Squared Error: {mse_C}")

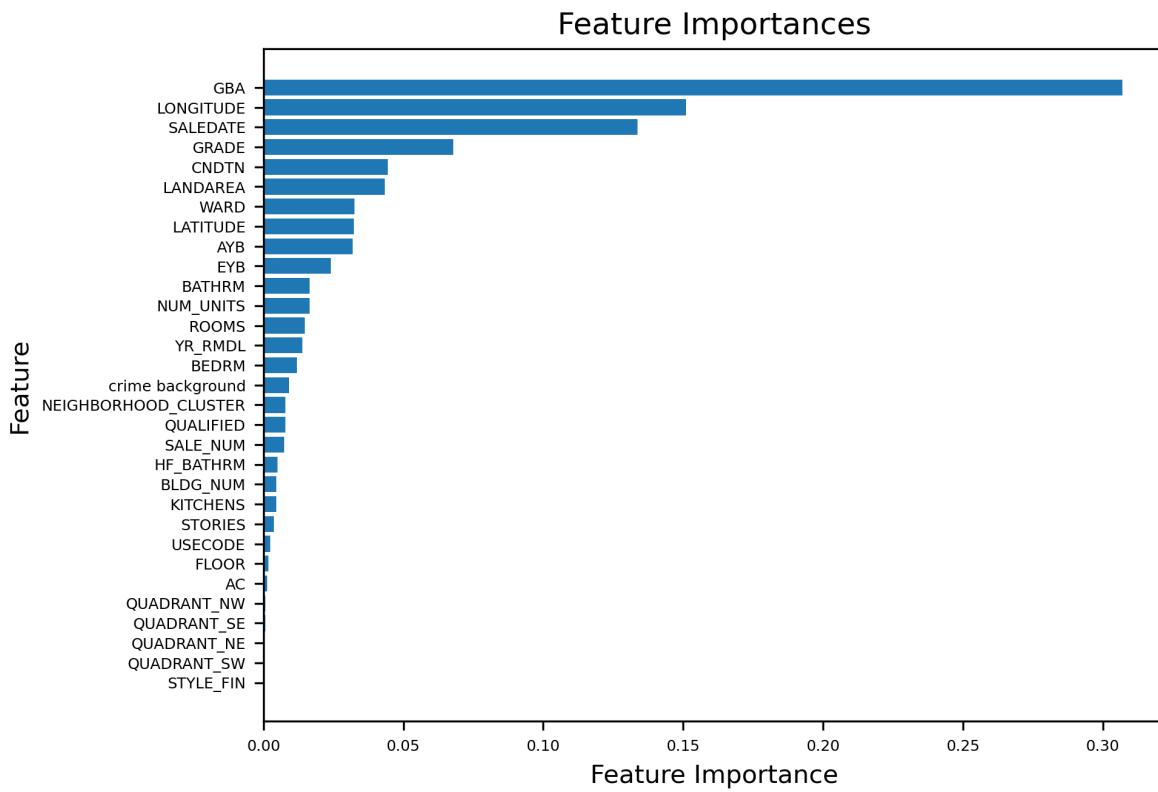
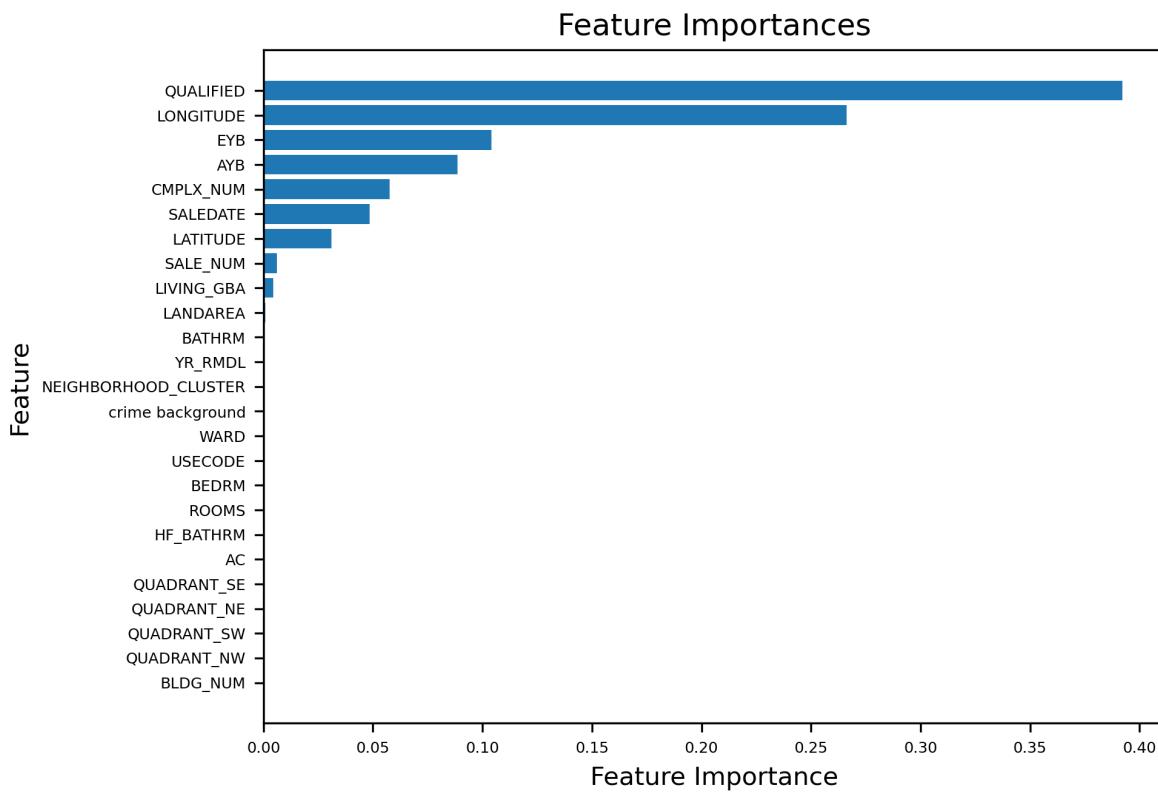
importances = rf.feature_importances_

feature_names = x_Ctrain.columns

sorted_indices = importances.argsort()

plt.barh(range(len(sorted_indices)), importances[sorted_indices], align='center')
plt.yticks(range(len(sorted_indices)), [feature_names[i] for i in sorted_indices])
plt.xlabel('Feature Importance')
plt.ylabel('Feature')
plt.title('Feature Importances')
plt.show()
```

注意这里我们导入的数据已经经过标准化处理。得到的特征重要性图像分别如下图所示：



发现对公寓价格影响最大的是 `QUALIFIED` 这一属性，其次是经度位置。而对住宅而言，`GBA`，即总建筑面积，对于房屋价格影响是最大的。而紧随其后的同样是经度信息。二者的均方误差分别为  $10^{-4}$  以及  $10^{-5}$  次方量级，可以认为回归效果本身是出色的。`QUALIFIED` 这一属性可能是指该公寓是否具有出租资格，非法出租与合法出租对于价格的影响是巨大的，而住宅的面积同样对于独栋建筑影响巨大，`GBA` 这一属性对于房屋价格影响最大也是意料之中。`LONGITUDE` 对于两种房屋的价格都具有巨大影响，我们可以推测华盛顿的东西部地区发展是不平衡的，从上图中的犯罪次数以及其他资料也反映出华盛顿的市中心位于给定数据分布的西北部以及西南部，总体而言西部更为发达。然而无论哪种房屋类型，我们发现犯罪对其价格的影响都是有限的。