Analysis on NYC 311 Complaints Data

Lili Xu
NYU, School of Engineering
Brooklyn, NY
lx474@nyu.edu

Tingjun Zhang
NYU, School of Engineering
Brooklyn, NY
tz976@nyu.edu

Suping Shi
NYU, School of Engineering
Brooklyn, NY
ss10438@nyu.edu

Abstract — In this project, we focus on applying different methods on big data analytics including data cleaning to check data quality. Then apply techniques on Hadoop Stack for data analyzing. The final results are expected to render the relationship between complaints type, complaints number along with the area, zip code, etc.

Keywords — Hadoop, Spark, Big Data Analysis, Data Cleaning

I. INTRODUCTION

The goals of our project are cleaning irrelevant or useless data, analyzing real big data and getting summary from data. Our team chooses to analyze the complaints data on NYC311 in 2009-2016. The project is divided into two parts. We do data cleaning in the first part and then data analysis in the second part. We clean data by means of seeking anomalies and eliminating them for every column. And then we plan to explore the data which has been cleaned in part one, in order to get conclusions.

In the data cleaning part, we do value check in various columns in order to find out what we need to clean. Then according to the result of value check, we discard useless column, coerce values and fill NULL value which makes it easy for us to manipulate data in the following part.

In the data analyzing part, in order to easy to understand for readers, we plot a lot of figures in many different cases. It aims to explore the relationship between the number of complaints and various factors. And we conclude the relationship in the end of every analysis, so it is easy for readers to find out the relationship with different factors.

II. DATA CLEANING

The purpose of this part is to check the quality of the dataset, get a summary of the data quality issues and apply multiple methods to clean the data and reduce the negative impact from the abnormal data.

2.1 Preliminary value check

We first start by have a preliminary view of the different columns. By taking a simple look, we found many possibilities of abnormal values. e.g. empty values, N/A string, Not Specified for the string and illegal values appearance. We apply check on those data in Spark and get corresponded output for each columns.

We firstly read that csv file in HDFS into spark dataframe and then apply multiple basic check functions including count(), describe() to see the overall data quality and we check the invalid data in each column including NULL, Unspecified, N/A and 0 Unspecified after then.

df = sqlContext.read.load('20.csv', format='com.databricks.spark.csv', header='true', inferSchema='true')*

df.count()

df.describe().show()

We found there are many null/empty values in the location related columns, almost all school related, park related columns are with the value of unspecified, almost all facility type columns have N/A and many other columns have N/A. 0 Unspecified is about half of all Community Board columns.

df.select([count(when(col(c).isNull(), c)).alias(c) for c in df.columns]).take(1)

df.select([count(when(col(c) == 'N/A', c)).alias(c) for c in df.columns]).take(1)

df.select([count(when(col(c)=='Unspecified', c)).alias(c) for c in df.columns]).take(1)

df.select([count(when(col(c)=='0 Unspecified', c)).alias(c) for c in df.columns]).take(1)

df.groupBy('Created Date').count().describe().show()

2.2 Column-specific value check

2.2.1 Classification attributes columns

For those classification attributes, such as ‘Agency’, ‘Agency Name’, ‘Complaint Type’, ‘Address Type’, ‘City’, ‘Facility Type’, ‘Borough’ and ‘Park Borough’, we apply a groupby function on each values and check if the minimum occurrence is legal. The result is shown below.

‘Agency’:

```
>>> df.groupBy('Agency').count().sort('count').show()
+----------+-----+
|    Agency|count|
+----------+-----+
|       LPC|    1|
|       TFA|    1|
|       OMB|    1|
|       DVS|    1|
|        DV|    3|
| DESIGNCOM|    3|
|       WF1|    3|
|NYCSERVICE|    4|
|       CWI|    4|
|       VAC|    4|
|      LOFT|    4|
|       TAT|    5|
|      EMTF|    5|
|       OAE|    5|
|    NYCERS|    5|
|       CEO|    6|
|       SBS|    7|
|       OEC|    7|
|       OPA|    7|
|     OCHIA|    8|
+----------+-----+
```

Overall, the data seems good.

'Agency Name':

```
>>> df.groupBy('Agency Name').count().sort('count').show()
+--------------------+-----+
|         Agency Name|count|
+--------------------+-----+
|School - ALC Park...|    1|
|School - The Scho...|    1|
|School - Communit...|    1|
|School - Carl C I...|    1|
|School - Forsyth ...|    1|
|School - Global T...|    1|
|School - Urban As...|    1|
|School - MS M245 ...|    1|
|     School - PS X017|    1|
|School - PS 23 at...|    1|
|School - PS X037 ...|    1|
| CFC - Staten Island|    1|
|School - World Ac...|    1|
|School - Emolior ...|    1|
|School - Carl C I...|    1|
|School - Herbert ...|    1|
|New York Police P...|    1|
|CFC - Brooklyn South|    1|
|School - PS 287 B...|    1|
|School - Mount Ed...|    1|
+--------------------+-----+
```

Overall, the data seems good.

'Complaint Type':

```
>>> df.groupBy('Complaint Type').count().sort('count').show()
+--------------------+-----+
|      Complaint Type|count|
+--------------------+-----+
|     Advocate - Lien|    1|
|Asbestos/Garbage ...|    1|
|               SG-99|    1|
|         Unspecified|    1|
|                 SNW|    1|
|   Sewer Maintenance|    1|
|        Micro Switch|    1|
|Sidewalk Cafe Heater|    1|
|                MOLD|    1|
|                 CST|    1|
|Meals Home Delive...|    1|
|      Unlicensed Dog|    1|
|                LEAD|    1|
|     Trapping Pigeon|    2|
|          Laboratory|    3|
|Advocate-Business...|    3|
|             Comment|    3|
|     Advocate - RPIE|    4|
|  Hazardous Material|    4|
|        Advocate-UBT|    5|
+--------------------+-----+
```

One 'Unspecified' row detected.

'Address Type':

```
>>> df.groupBy('Address Type').count().sort('count').show()
+------------+--------+
|Address Type|   count|
+------------+--------+
|   PLACENAME|    8717|
|     LATLONG|  110624|
|   BLOCKFACE|  456203|
|            |  793237|
|INTERSECTION| 2667045|
|     ADDRESS|12992974|
+------------+--------+
```

793237 null value detected.

'City':

```
>>> df.groupBy('City').count().sort('count').show()
+---------------+-----+
|           City|count|
+---------------+-----+
|         NAVADA|    1|
|  CAROLL STREAM|    1|
|       NEW HOPE|    1|
|        PHONIEX|    1|
|         SONAMA|    1|
|      HICKSVILE|    1|
|       WALDWICK|    1|
|         ORNAGE|    1|
|  NORTH  BERGEN|    1|
|    WESTBOROUGH|    1|
|          ELLEN|    1|
|WEST HARRINGTON|    1|
|       HIOBOKEN|    1|
|  INDIANAPOLLIS|    1|
|        TAMARAC|    1|
|     N. MERRICK|    1|
| FARMINGDALE NY|    1|
|   WADING RIVER|    1|
|       RED BANK|    1|
|        MENDHAM|    1|
+---------------+-----+
```

Overall, the data seems good.

'Facility Type':

```
>>> df.groupBy('Facility Type').count().sort('count').show()
+---------------+--------+
|  Facility Type|   count|
+---------------+--------+
|School District|    3588|
|         School|   13147|
|               |   22683|
|    DSNY Garage|  548736|
|       Precinct| 3640367|
|            N/A|12800279|
+---------------+--------+
```

Huge amount of Null or N/A value detected. This column might not be useful for further exploration.

'Borough':

```
>>> df.groupBy('Borough').count().sort('count').show()
+-------------+-------+
|      Borough|  count|
+-------------+-------+
|STATEN ISLAND| 807513|
|  Unspecified|1573765|
|        BRONX|2883670|
|    MANHATTAN|3253820|
|       QUEENS|3694933|
|     BROOKLYN|4815099|
+-------------+-------+
```

Many ‘Unspecified’ rows detected. This column might not be useful for further exploration.

‘Park Borough’:

```
>>> df.groupBy('Park Borough').count().sort('count').show()
+-------------+-------+
| Park Borough|  count|
+-------------+-------+
|STATEN ISLAND| 807513|
|  Unspecified|1573765|
|        BRONX|2883670|
|    MANHATTAN|3253820|
|       QUEENS|3694933|
|     BROOKLYN|4815099|
+-------------+-------+
```

Many ‘Unspecified’ rows detected. This column might not be useful for further exploration.

2.2.2 Date attributes check

```
>>> df.groupBy('Created Date').count().sort('count').show()
+--------------------+-----+
|        Created Date|count|
+--------------------+-----+
|11/22/2010 09:18:...|    1|
|11/22/2010 06:02:...|    1|
|11/17/2010 06:00:...|    1|
|11/19/2010 01:27:...|    1|
|11/17/2010 03:13:...|    1|
|11/19/2010 04:01:...|    1|
|11/18/2010 06:50:...|    1|
|11/20/2010 09:08:...|    1|
|11/18/2010 02:22:...|    1|
|11/20/2010 09:11:...|    1|
|11/19/2010 08:12:...|    1|
|11/20/2010 10:50:...|    1|
|11/19/2010 04:19:...|    1|
|11/21/2010 01:57:...|    1|
|11/19/2010 12:05:...|    1|
|11/21/2010 08:13:...|    1|
|11/19/2010 10:43:...|    1|
|11/21/2010 12:30:...|    1|
|11/17/2010 04:54:...|    1|
|11/21/2010 12:05:...|    1|
+--------------------+-----+
```

We also did similar checks for ‘Due Date’ and ‘Resolution Action Updated Date’, results are similar, no wired data detected.

2.2.3 Zip code check

There are some illegal zip codes in this dataset. We use regular expression to filter out those illegal zip codes. The basic idea to find out those zip codes that contain alphabetic characters or whose length is suspicious. The pyspark code is:

df.where(length(col('Incident Zip')) > 0).select(col('Incident Zip')).filter(col('Incident Zip').rlike('^(\d{5}(-)?(\d{4})?|[A-Z]\d[A-Z] ?\d[A-Z]\d)$')==False).groupBy('Incident Zip').count().show()

Part of the illegal zip codes are shown below as an example.

```
+------------+-----+
|Incident Zip|count|
+------------+-----+
|        1143|    1|
|        1305|    1|
|   11434-420|    1|
|      080111|    1|
|   1182-9060|    1|
|    0000-000|    1|
|    NY 10604|    1|
|  48195/0954|    1|
|         N.A|    1|
|     UNKNOWN|   33|
|         NTY|    1|
|        7823|    1|
|          NA|  241|
|       00000|    1|
|        0031|    1|
|        1373|    1|
|  30348/5689|    1|
|        7666|    1|
|  NY 10010-3|    1|
|   60076-102|    1|
+------------+-----+
only showing top 20 rows
```

2.2.4 Surprising or suspicious data

We also apply some other functions to do some additional checking on data types integrity, value range and surprisingly occurrence.

Count the number of the cases grouped by creation date and check if any number is surprisingly low or high.

```
>>> df.groupBy('Created Date').count().describe().show()
+-------+-----------------+
|summary|            count|
+-------+-----------------+
|  count|         10382112|
|   mean|1.639252109782672|
| stddev|32.01452978693912|
|    min|                1|
|    max|             9397|
+-------+-----------------+
```

Count and group by closed date.

```
>>> df.groupBy('Closed Date').count().describe().show()
+-------+-----------------+
|summary|            count|
+-------+-----------------+
|  count|          6931064|
|   mean|2.4554525827491998|
| stddev| 225.5681315170334|
|    min|                1|
|    max|           582797|
+-------+-----------------+
```

We found many case are not yet closed and with empty row, which is normal.

```
>>> df.groupBy('Closed Date').count().orderBy('count',ascending=False).take(5)
[Row(Closed Date=u'', count=582797), Row(Closed Date=u'01/21/2009 12:00:00 AM', count=7759), Row(Clos
ed Date=u'02/19/2009 12:00:00 AM', count=7551), Row(Closed Date=u'11/07/2012 12:00:00 AM', count=7462
), Row(Closed Date=u'02/27/2009 12:00:00 AM', count=7363)]
```

2.3 Data Clean

2.3.1 Discard Columns

So far, as we found that many columns are with majority of the invalid values. We choose to discard those columns as they do not contribute to the final analysis.

drop_list = ['Facility Type', 'School Name', 'School Number', 'School Region', 'School Code', 'School Phone Number', 'School Address', 'School City', 'School State', 'School Zip']

df = df.select([column for column in df.columns if column not in drop_list])

2.3.1 Coerce Values

As we found some columns are with some invalid values, while those values are of different patterns. We coerce those patterns into a fixed value for later easy manipulation. e.g. For values in ‘Incident Zip’, we change all those invalid values into a fixed ‘N/A’ string.

df = df.withColumn('Incident Zip', when(col('Incident Zip').rlike('^(\d{5}(-)?(\d{4})?|[A-Z]\d[A-Z] ?\d[A-Z]\d)$')== False, 'N/A').otherwise(df['Incident Zip']))

2.3.1 Fill Null Values

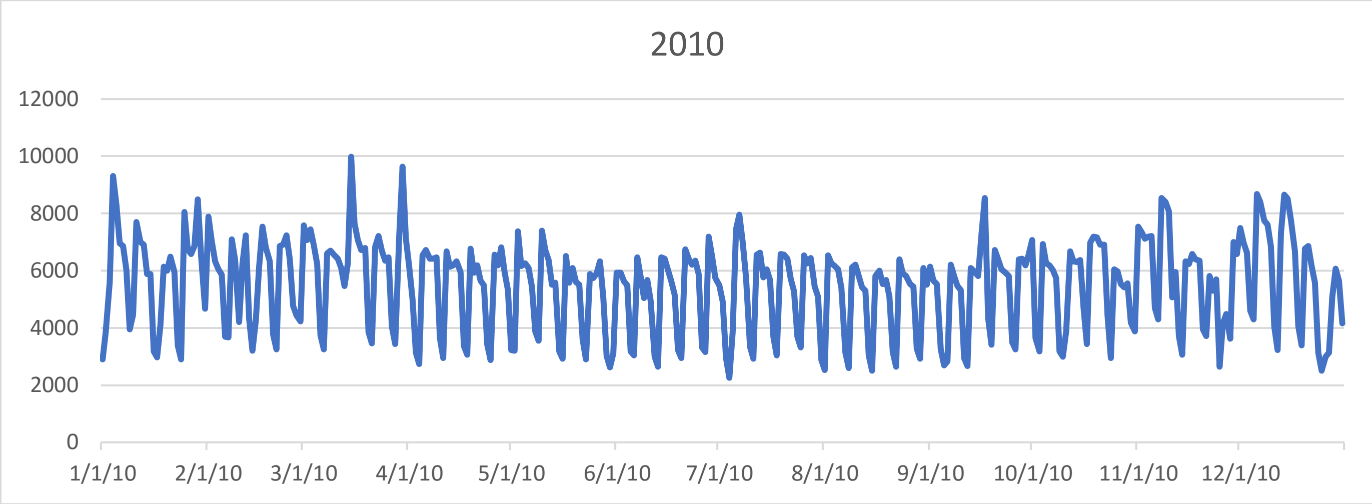
To eliminate the null values, we choose to transform it into ‘N/A’ string for all columns.

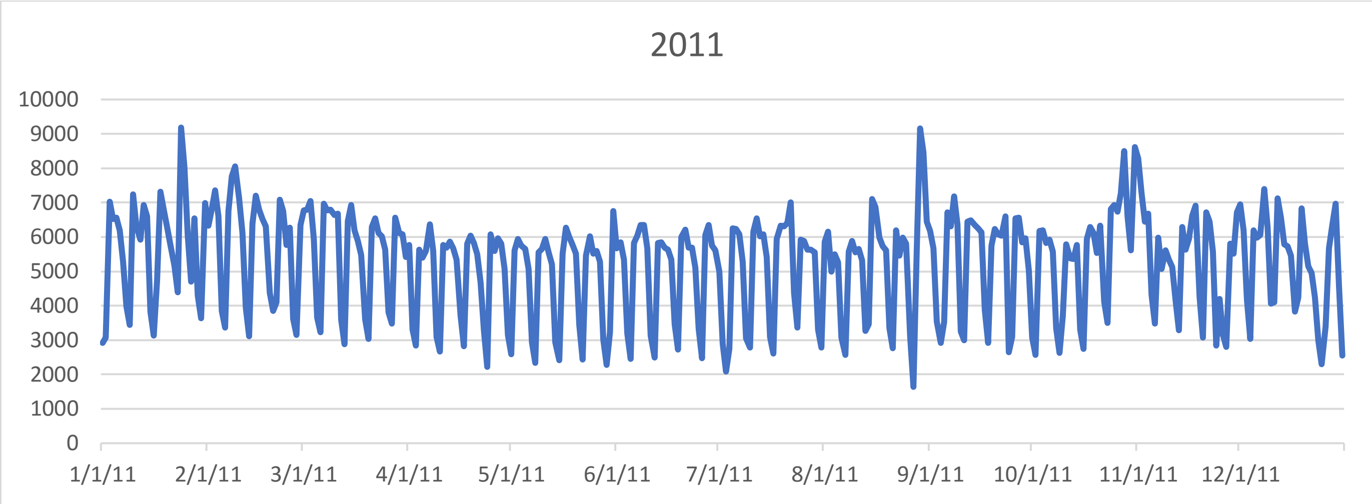
df = df.fillna(‘N/A’)

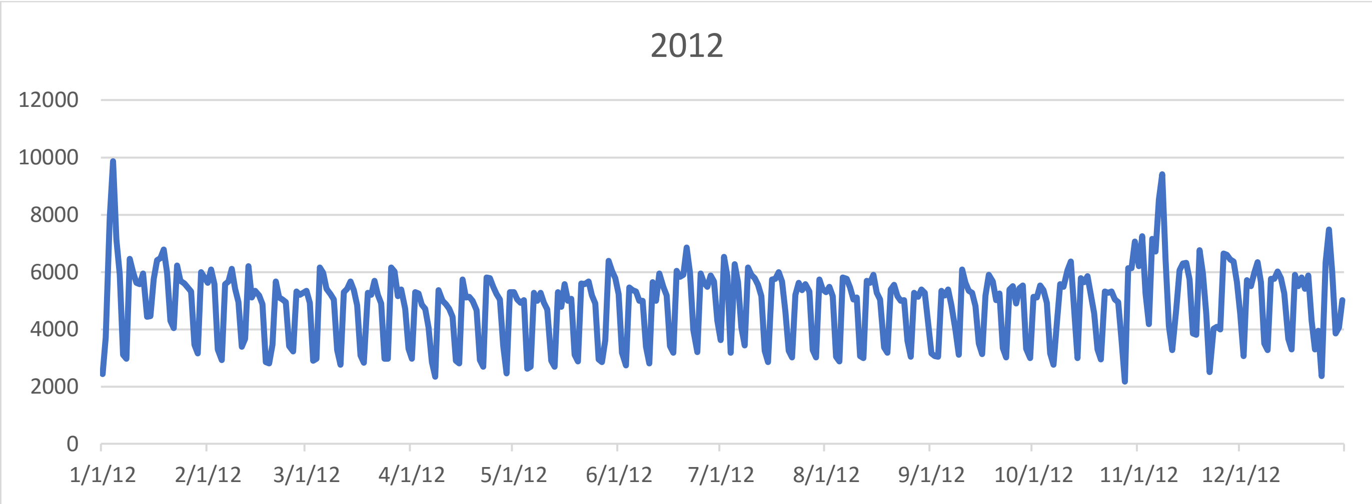
III. DATA ANALYSIS

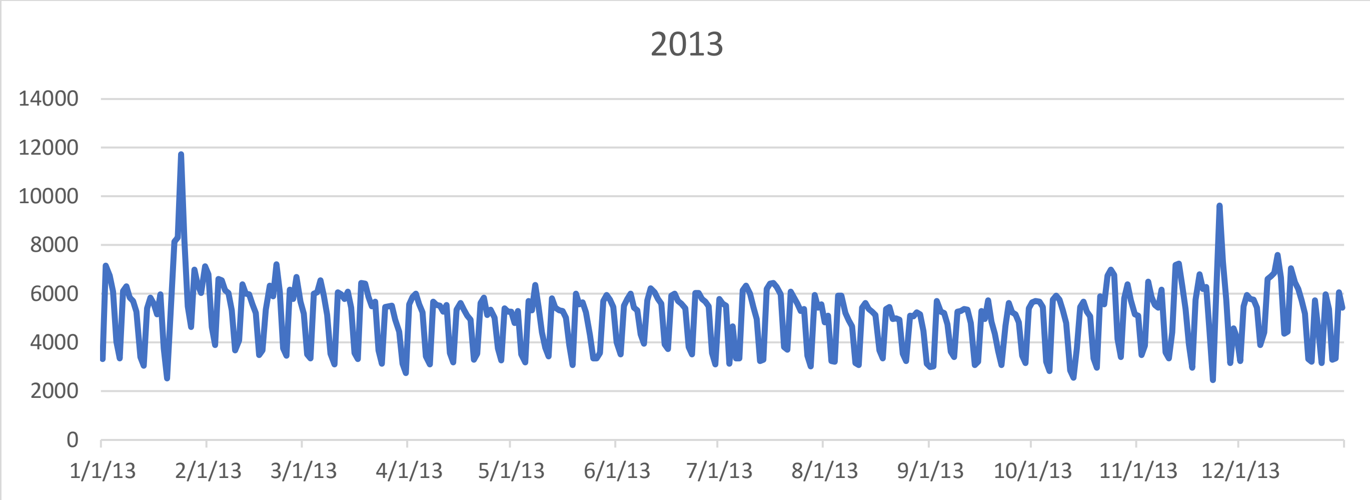
3.1 Case Creation Date

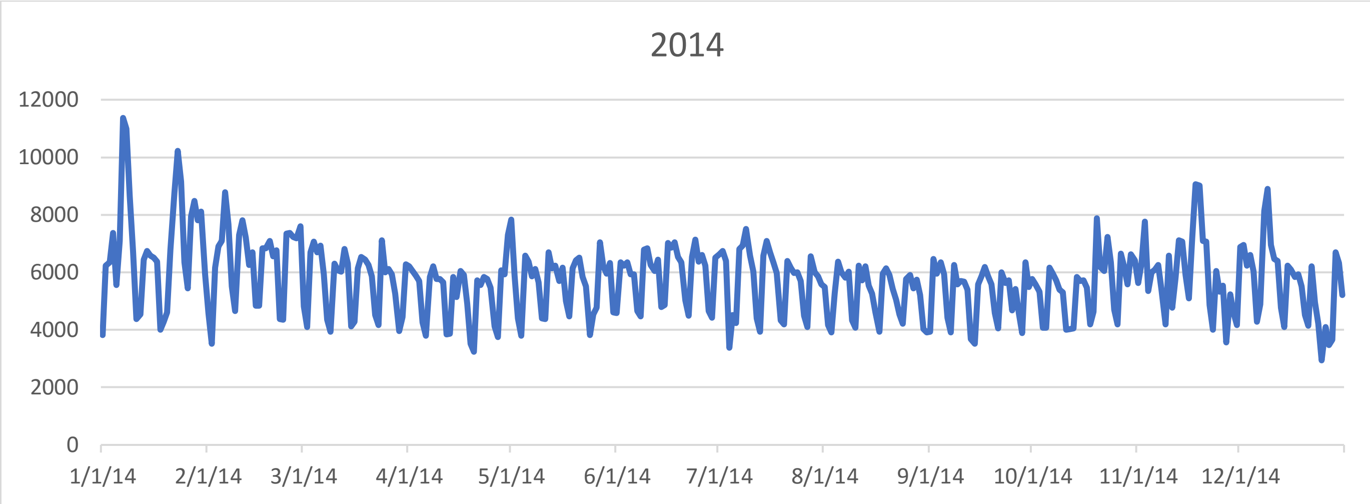
Create Days column is useful to explore when were most complaints created and the trend of the whole year. By grouping it into individual days, it helps us better explore the relationship between the day and the number of complaints.

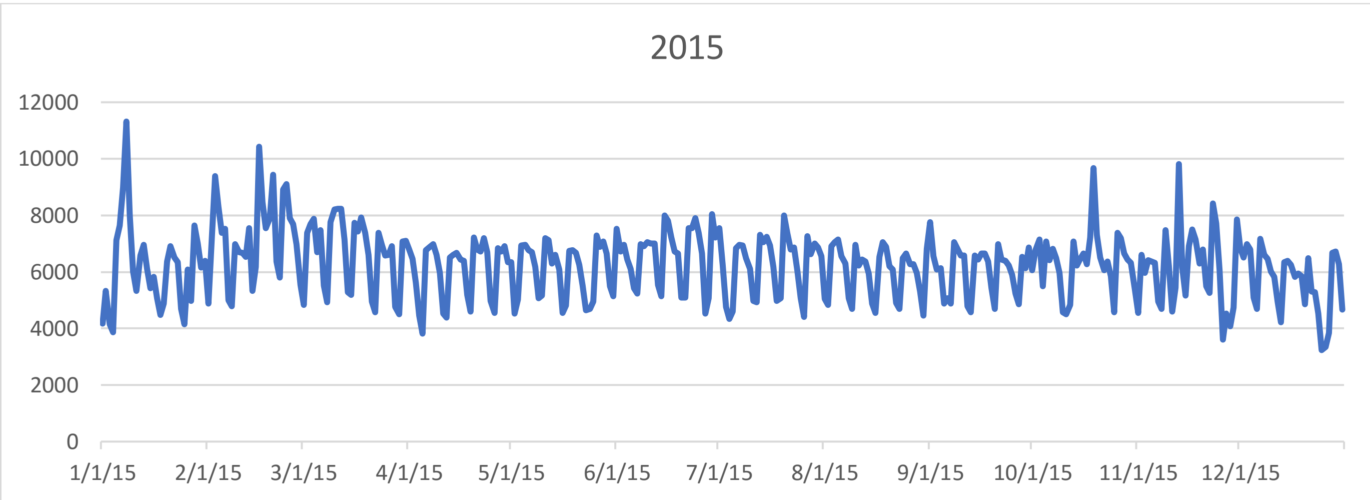


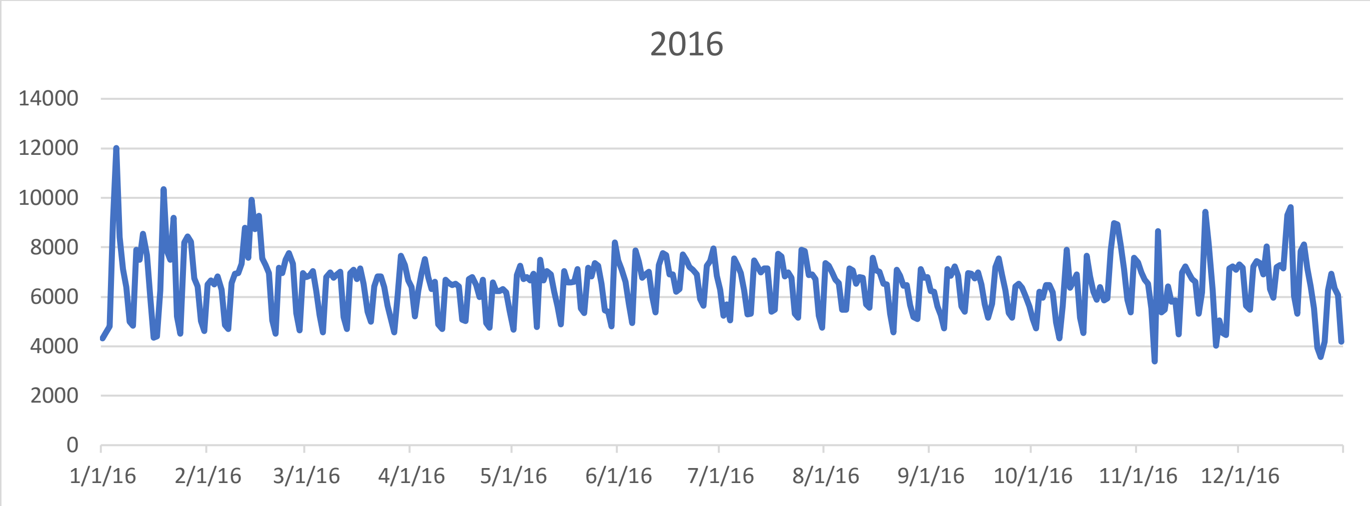





2013
14000
12000
10000
8000
6000
4000
2000
0
1/1/13
2/1/13
3/1/13
4/1/13
5/1/13
6/1/13
7/1/13
8/1/13
9/1/13
10/1/13
11/1/13
12/1/13

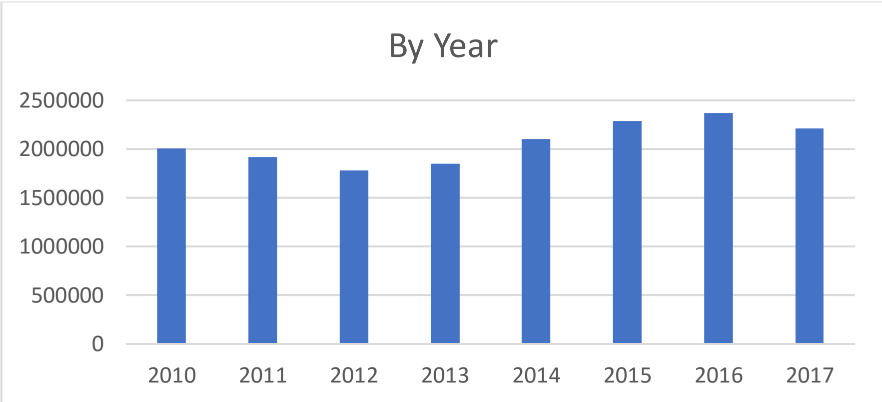

2014
12000
10000
8000
6000
4000
2000
0
1/1/14
2/1/14
3/1/14
4/1/14
5/1/14
6/1/14
7/1/14
8/1/14
9/1/14
10/1/14
11/1/14
12/1/14

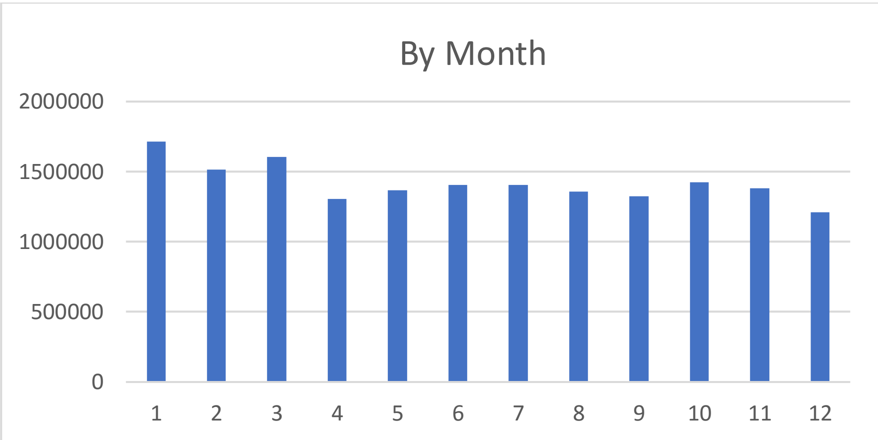

2015
12000
10000
8000
6000
4000
2000
0
1/1/15
2/1/15
3/1/15
4/1/15
5/1/15
6/1/15
7/1/15
8/1/15
9/1/15
10/1/15
11/1/15
12/1/15



As we see, the complaints are usually more in winter compared with other normal days. The count are choppy with days.

We apply further exploration on these count by different years and months.





The cases in January are the most, it is clearer that cold winter has the positive relationship with the number of complaints. By year, we can see that the previous year complaints might have some relationship with the following year.

Following part will show that five highest and lowest the number of complaints in each. Because the data of year 2009 and 2017 is incomplete, it will show 2010-2016.

2010

Highest		Lowest	
Date	Number	Date	Number
03/15	9983	07/04	2254
03/30	9643	08/15	2496
01/04	9315	12/25	2496
12/06	8676	08/01	2531
12/14	8654	08/08	2600

2011

Highest		Lowest	
Date	Number	Date	Number
01/24	9184	08/27	1639
08/29	9150	07/03	2082
10/31	8621	04/24	2215
01/03	8491	05/29	2284
12/27	8463	12/25	2289

2012

Highest		Lowest	
Date	Number	Date	Number
01/04	9863	10/28	2169
11/08	9409	04/08	2345
11/07	8525	12/25	2364
01/03	7874	01/01	2426
12/27	7472	04/29	2464

2013

Highest		Lowest	
Date	Number	Date	Number
01/24	11732	11/23	2458
11/25	9624	01/20	2532
01/23	8322	10/13	2550
01/25	8185	03/31	2734
01/22	8148	10/06	2835

2014

Highest		Lowest	
Date	Number	Date	Number
01/07	11367	12/25	2940
01/08	10998	04/20	3234
01/23	10219	07/04	3371
01/24	9162	12/27	3477
11/18	9068	09/14	3509

2015

Highest		Lowest	
Date	Number	Date	Number
01/08	11318	12/25	3235
02/16	10434	12/26	3339
11/13	9819	11/26	3614
10/19	9662	04/05	3815
02/20	9438	12/27	3854

2016

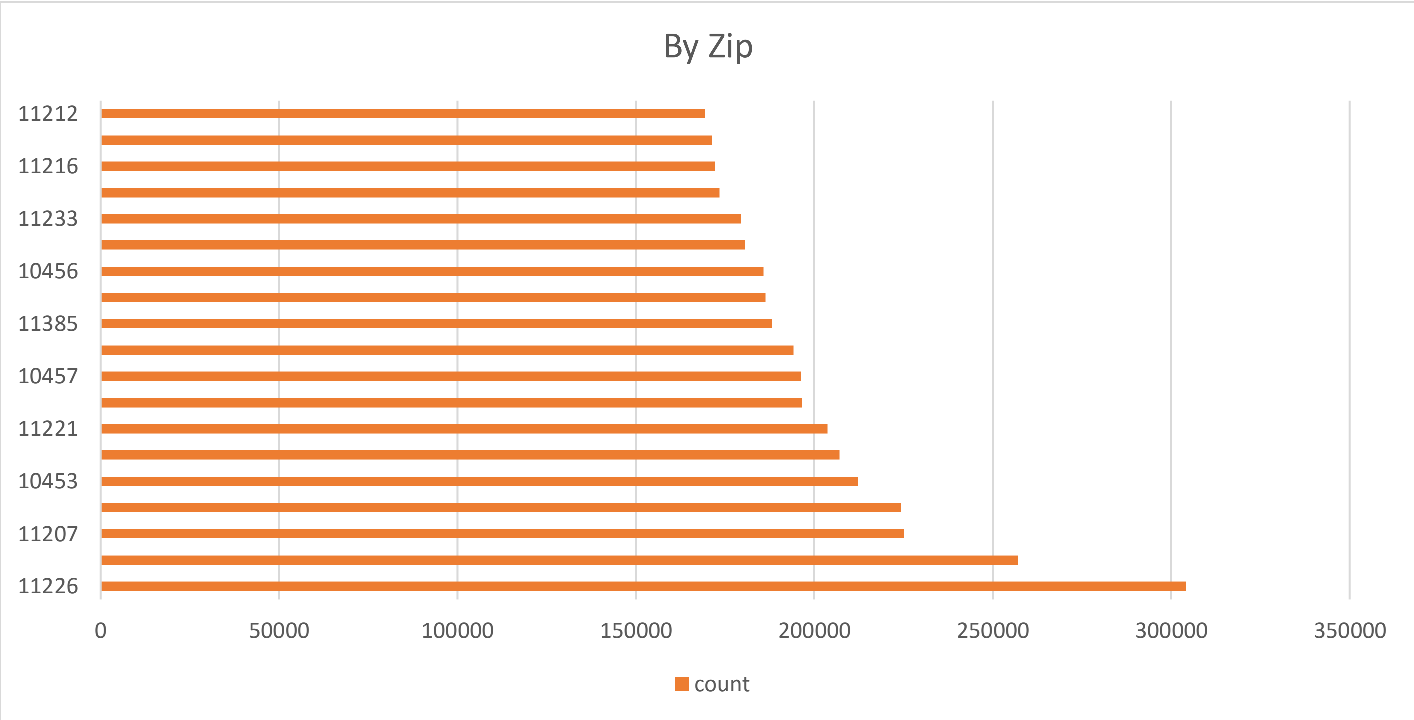
Highest		Lowest	
Date	**Number**	**Date**	**Number**
01/05	12012	11/06	3385
01/19	10338	12/25	3570
02/14	9907	12/24	3929
12/16	9614	11/24	4018
11/21	9430	12/26	4174

From above data, we can analyze the following three conclusions,

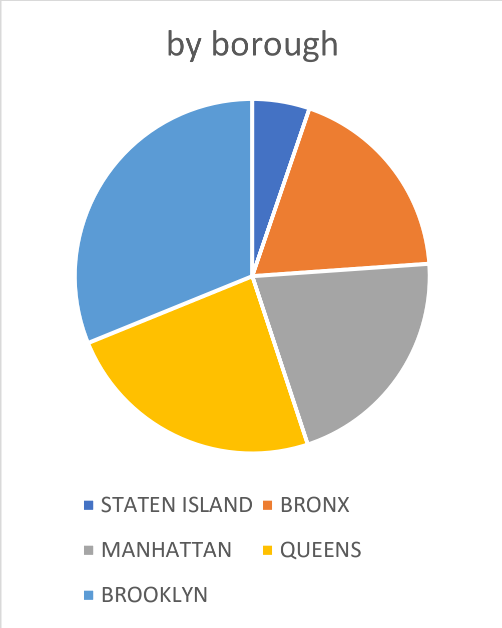
a. The number of complaints is the rising trend on the whole.
b. Holiday has an effluence on the number of complaints. For example, there are less complaints on Christmas Day every year.
c. Most complaints are in the winter. So, weather may affect the number of complaints to a certain extent. And we will discuss this relationship in the following part.

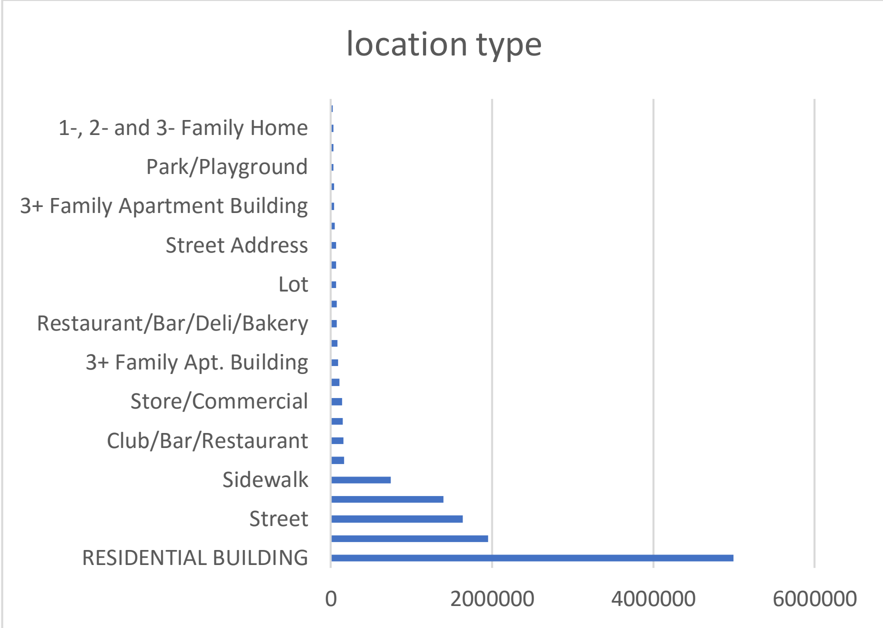
3.2 Location

We apply analysis on areas based on zip code (10 zip areas with the most complaint numbers) and borough.



Zip 11226 are with the most complaints. Staten island are of the least complaints, that might because of the population.

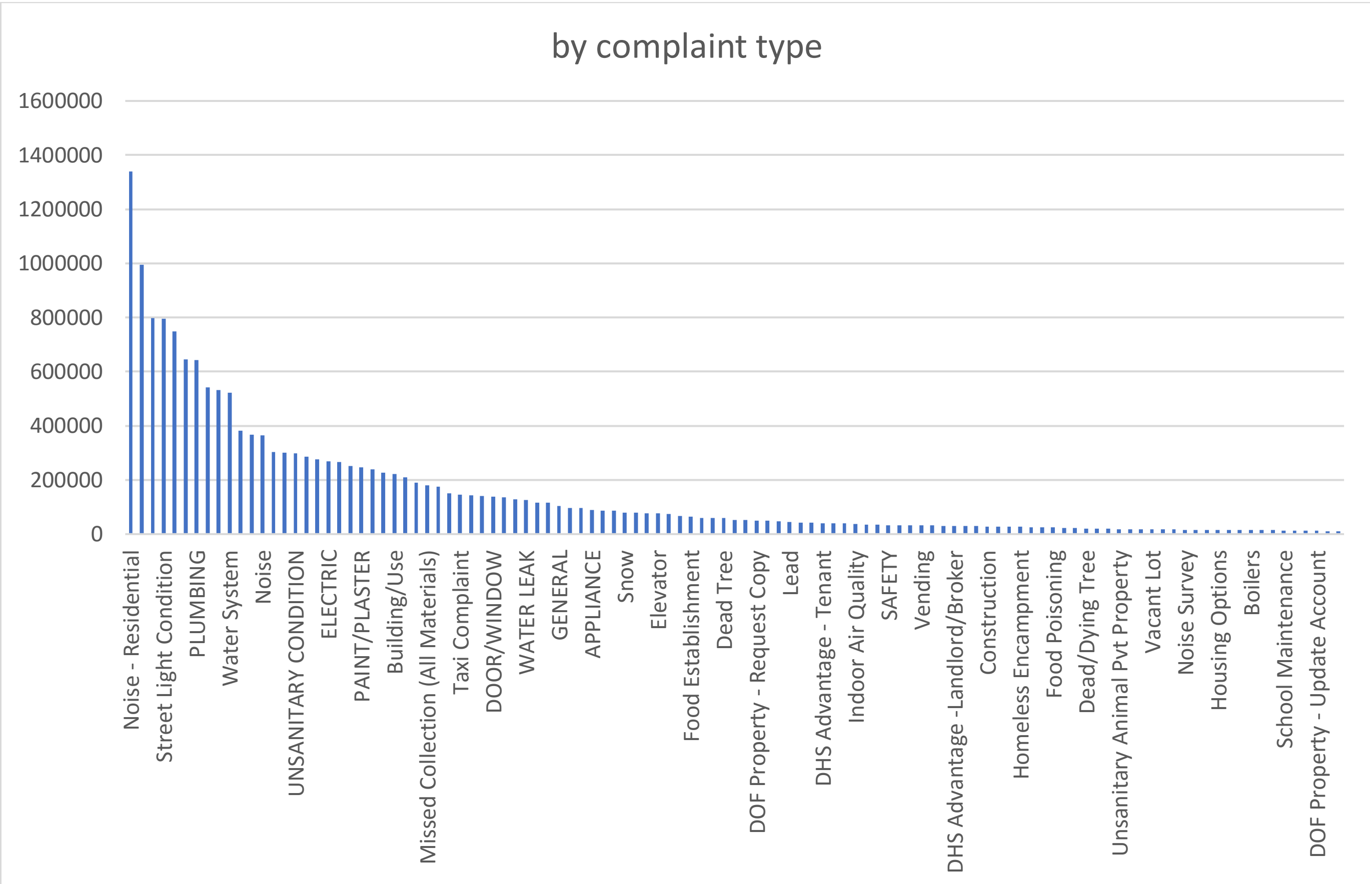




Most complaints are from residential buildings.

3.3 Complaint Type

Group by each different complaint type to have an overview of the number of different types.



3.4 Detailed Analysis on each year

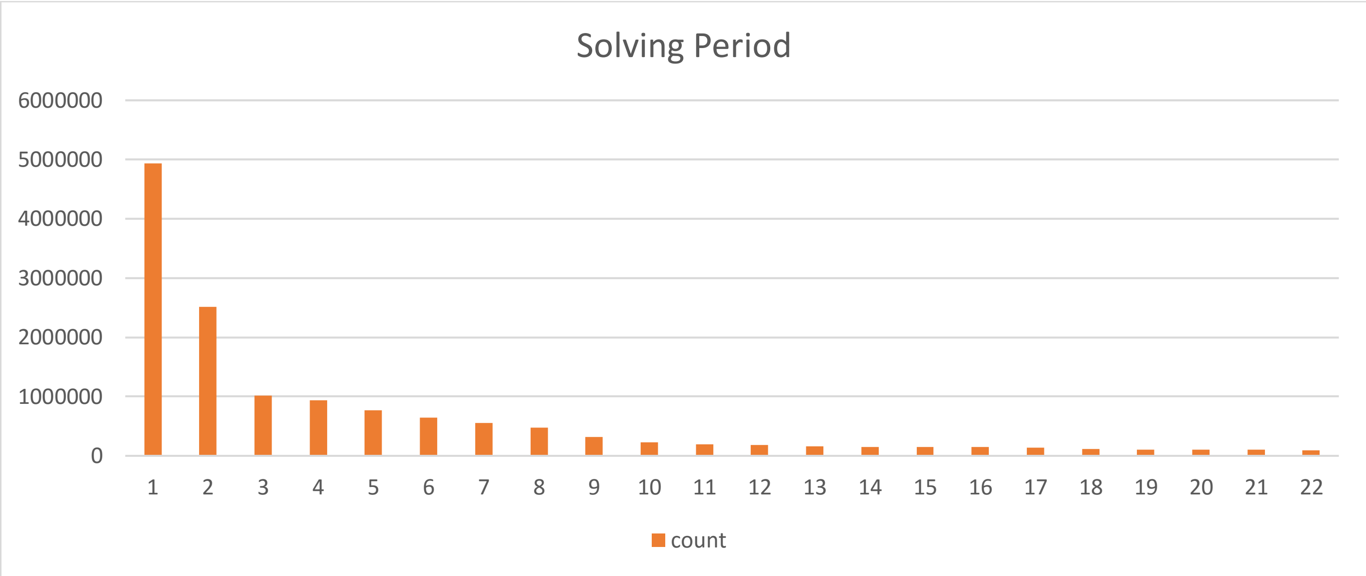
We apply a further close look on each year. To get analysis result based on different years.

Year	2010	2011	2012	2013	2014	2015	2016	2017
Total Case Created	2005760	1918896	1783212	1849019	2102226	2286951	2370339	2212889
Total Case closed	1856036	1792802	1730636	1799583	2057229	2243521	2303134	2169310
Daily average create	5495	5257	4872	5065	5759	6265	6476	6726
Daily average close	5085	4911	4728	4930	5636	6146	6292	6593
Most complaint typ	HEATING	HEATING	HEATING	HEATING	Noise - Residential	HEAT/HOT WATER	HEAT/HOT WATER	Noise Residential
Most/least zip	11226	11226	11226	11226	11226	11226	11226	11226
Most/least borough	BROOKLYN	BROOKLYN	BROOKLYN	BROOKLYN	BROOKLYN	BROOKLYN	BROOKLYN	BROOKLYN
Most Agency	HPD	HPD	HPD	HPD	HPD	HPD	NYPD	NYPD
Location Type	RESIDENTIAL BUILDING	RESIDENTIAL BUILDING	RESIDENTIAL BUILDING	RESIDENTIAL BUILDING	RESIDENTIAL BUILDING	RESIDENTIAL BUILDING	RESIDENTIAL BUILDING	RESIDENTIAL BUILDING

3.5 Case solving efficiency

An important aspect is the case solving efficiency, which is represented by the duration of the case opening, that is (close date – create date).

Almost all cases are solved within one week, most of them are within one week, which is efficient.

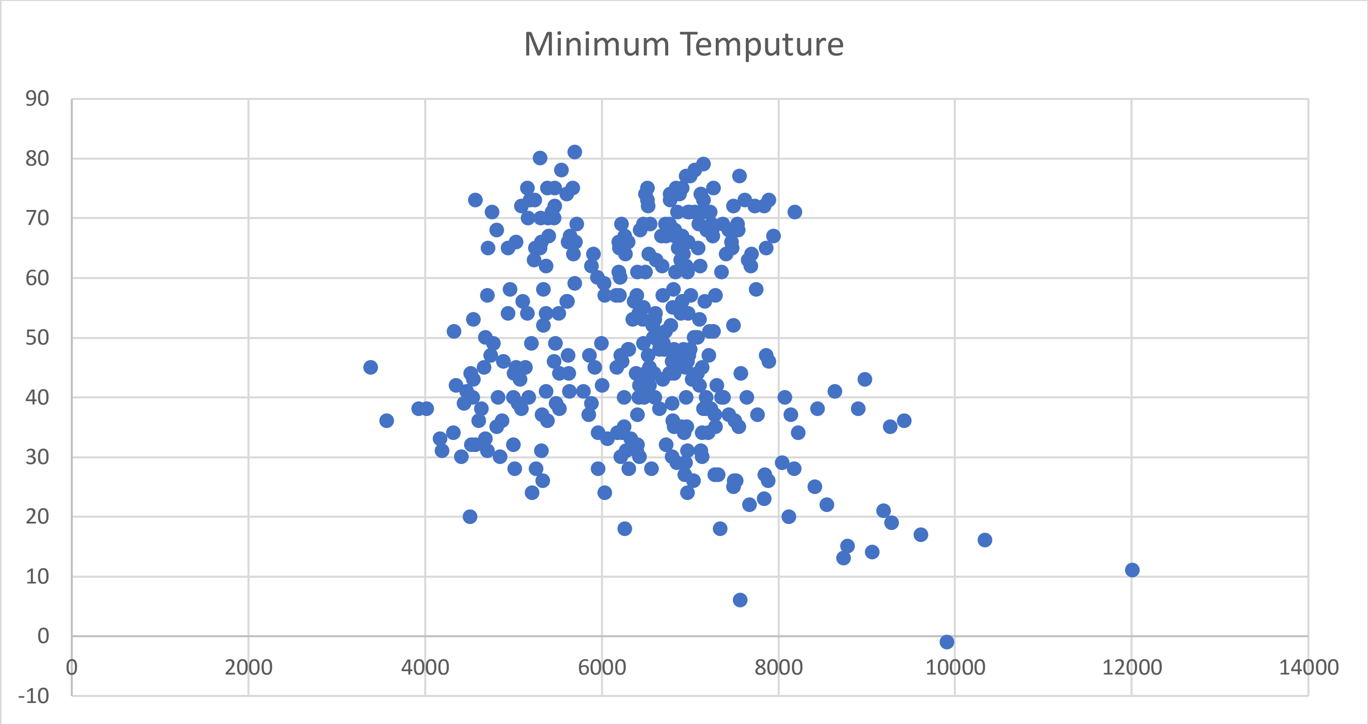


IV. FURTHER INVESTATION (BONUS PART)

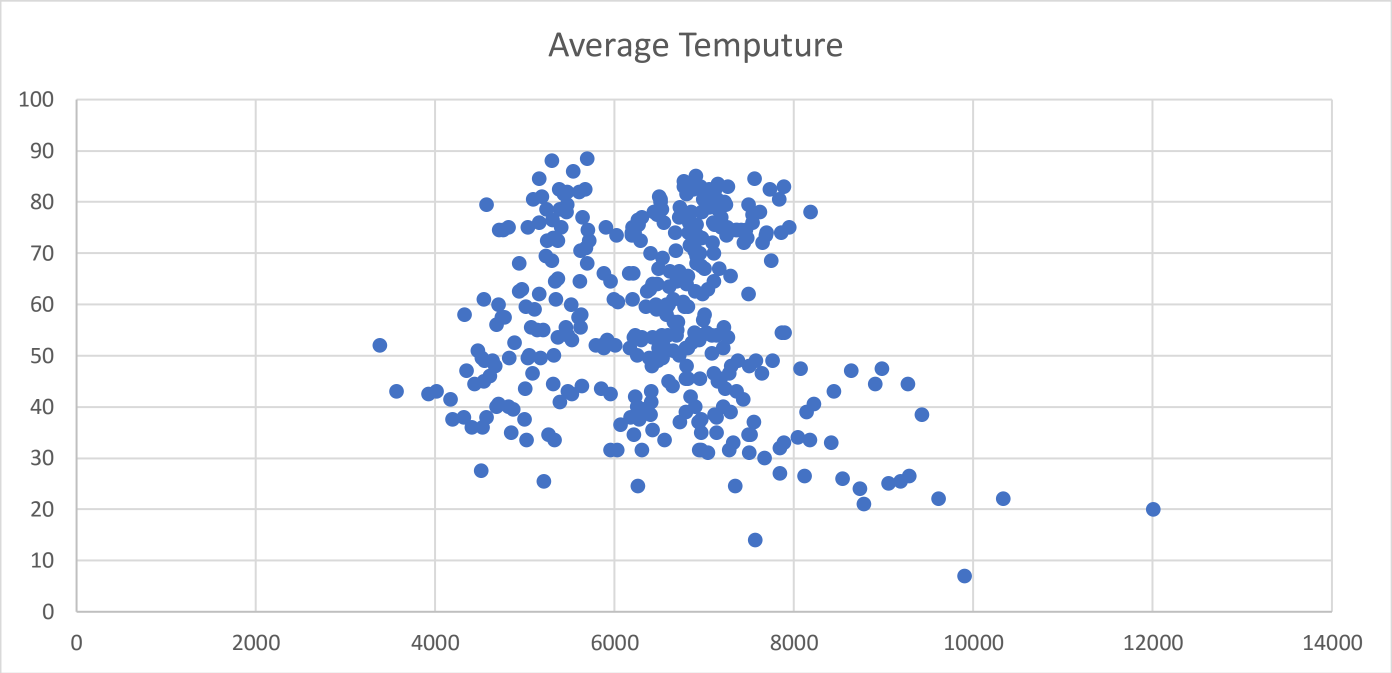
4.1 Weather

As we notice that the complaints count has some peak values during the cold days, we start the investigation from the relationship between weather and complaint numbers.

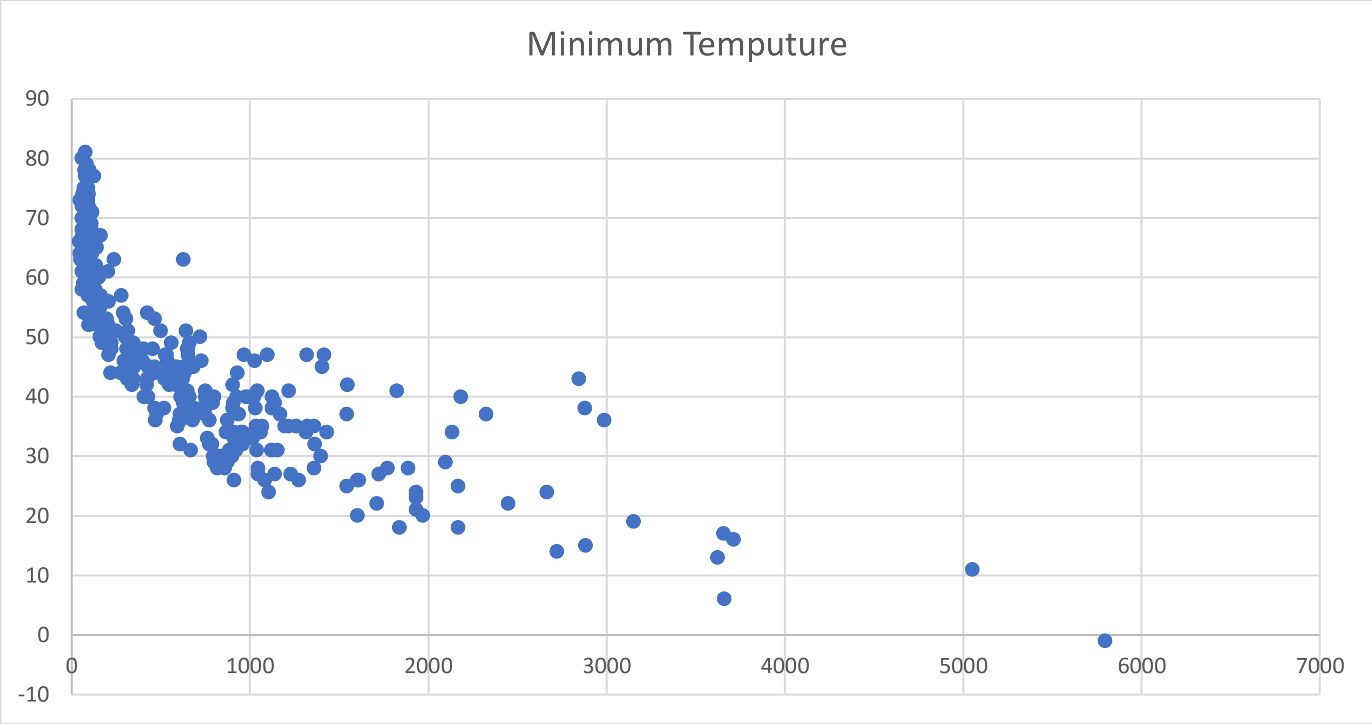
We get a weather dataset for year 2016, filter 2016 complaints records, group by day and count the numbers for each day. Compared with the weather dataset.



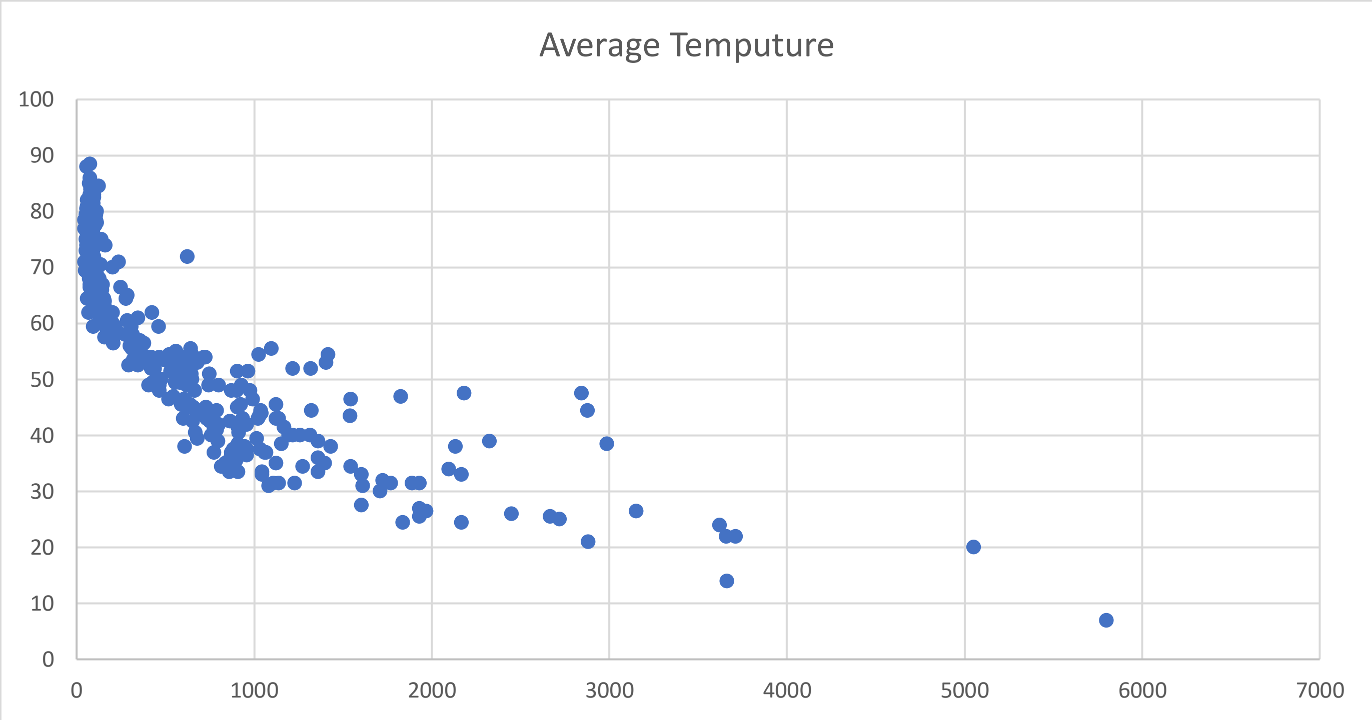
As shown in the diagram, the low temperature might have impact on those peak days, but most of the day complaints does not have any significant relationship with the temperature.



By filtering the heating complaints. We get the relationship between the heating complaints count and weather as below.



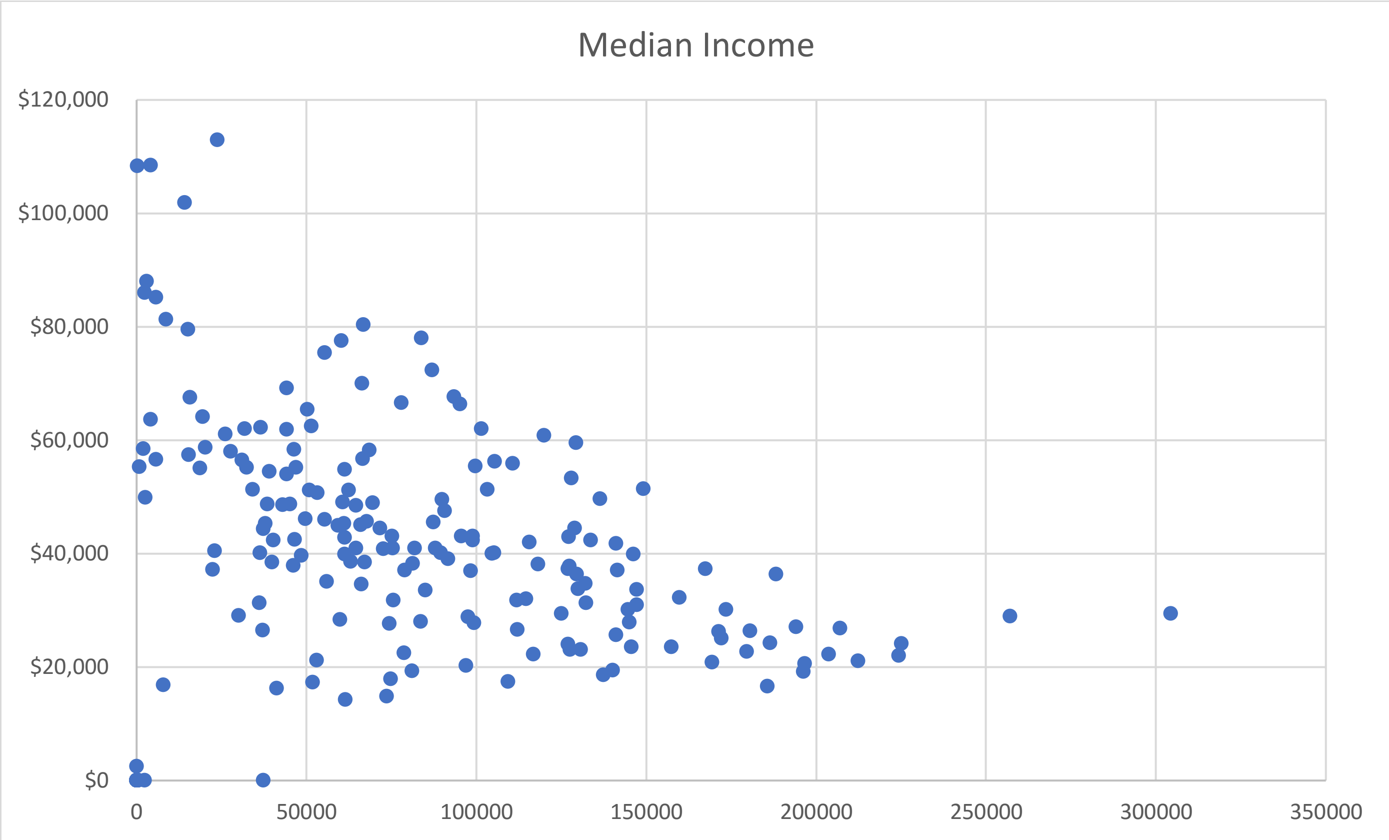
Clearly, the heating complaints have relationship between weather. The colder the weather is, the more the heating complaints are.



4.2 Income

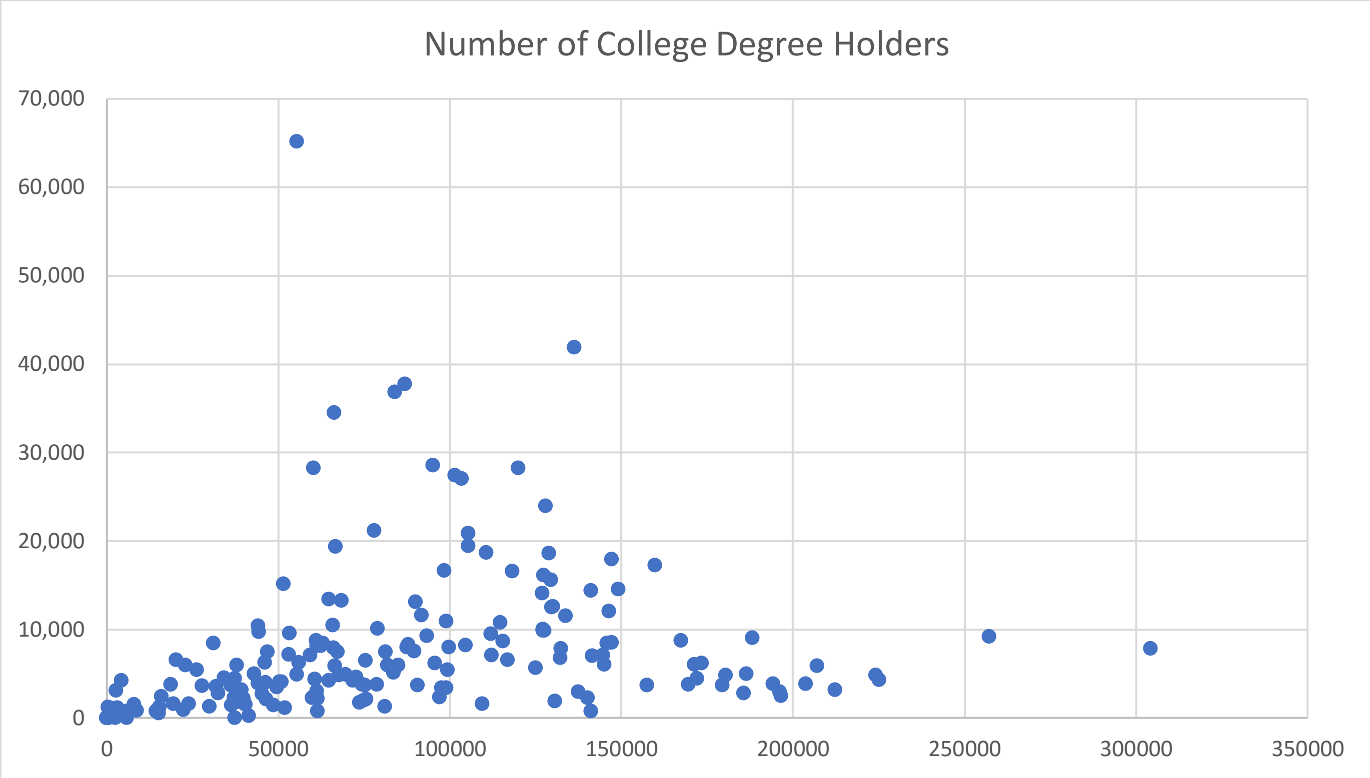
We apply join operation on income data frame and count data frame grouped by zip code. As we see on the following chart, zip code areas with higher median income are likely to have less complaints.

We also calculate the correlation coefficient, which is 0.0104. Technically, the two factors are not strongly linear correlated.

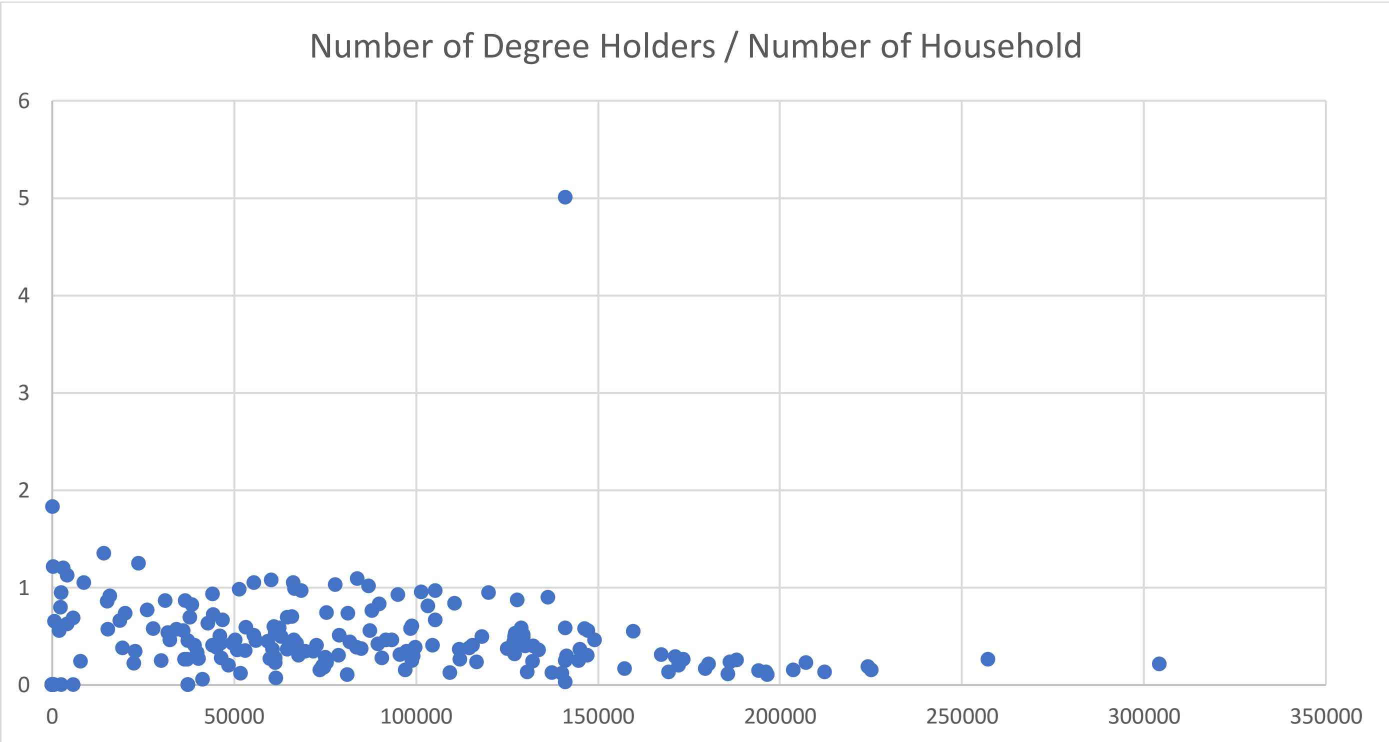


4.3 Education

We start our investigation by showing the relationship between the complaints counts and the number of people with college or higher degree in that zip code area.



We define a new index represented by number of degree holders divided by number of household to see the relationship between education level and complaints number in different zip code areas.

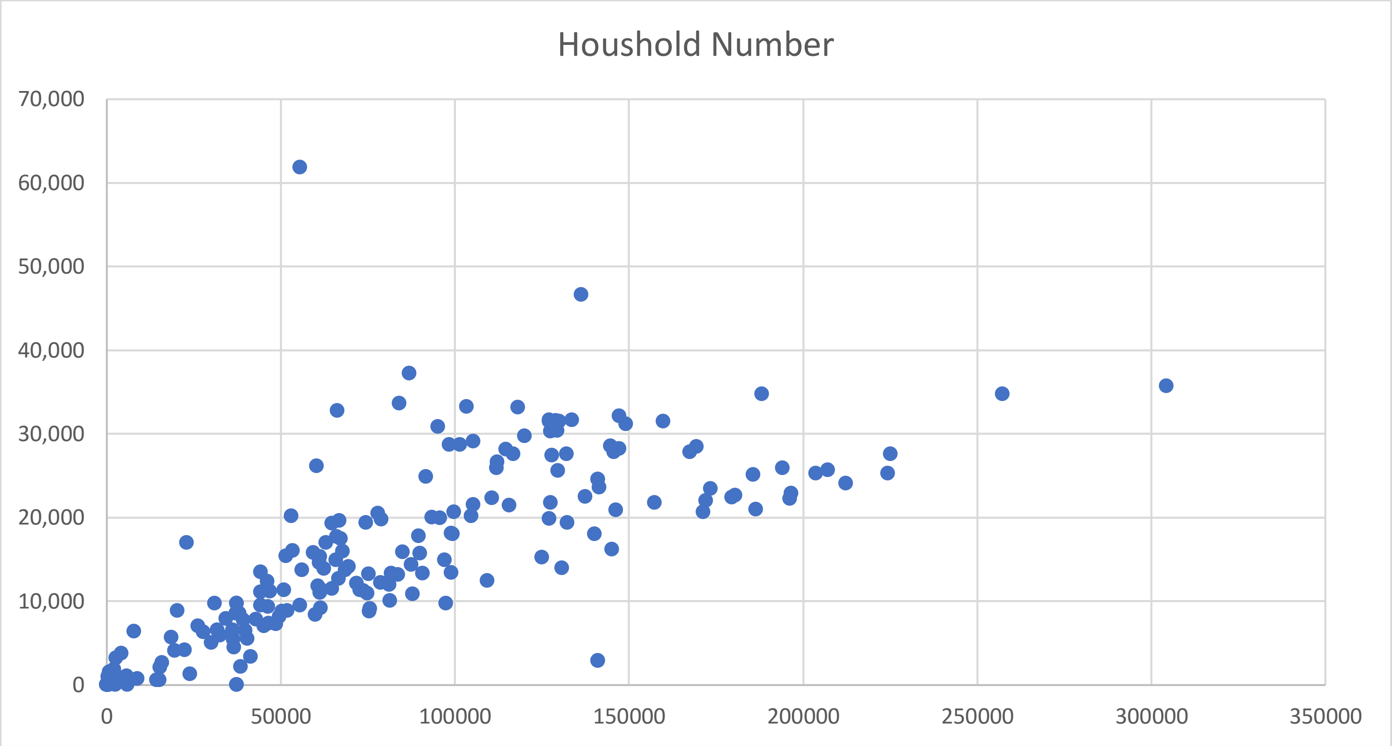


As we see, they do not have much strong relationship between each other. But a slight trend is the zip code areas with more bachelor holders are likely to have fewer complaints.

Then, we use linear regression model to calculate the correlation coefficient between median income and complaint number, which is about 0.0579. The calculation result shows that these two factors are not strongly related to each other, which proves the hypothesis we give from the chart.

4.4 Household

Household number is directly related to complaints number, which is apparent.



The correlation coefficient for household number and complaint number is about 0.6026, which confirms that the two number are strongly related to each other.

V. CODE

All of our code is stored on a GitHub repository (https://github.com/lilixu93/NYC-Data-Analysis).We mainly use Python for this project. All code is Hadoop or Spark related.

ACKNOWLEDGMENT

We would like to thank professor Claudio Silva for helping us with the idea, and advising on implementing various big data analysis techniques. We also thank for TAs for providing detailed instructions and queries answering.