nyc311_complaints

October 8, 2022

[1]:
```
!pip install missingno
```

```
Requirement already satisfied: missingno in /opt/anaconda3/lib/python3.8/site-
packages (0.5.1)
Requirement already satisfied: numpy in /opt/anaconda3/lib/python3.8/site-
packages (from missingno) (1.22.4)
Requirement already satisfied: matplotlib in /opt/anaconda3/lib/python3.8/site-
packages (from missingno) (3.2.2)
Requirement already satisfied: scipy in /opt/anaconda3/lib/python3.8/site-
packages (from missingno) (1.5.0)
Requirement already satisfied: seaborn in /opt/anaconda3/lib/python3.8/site-
packages (from missingno) (0.10.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/anaconda3/lib/python3.8/site-packages (from matplotlib->missingno) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/opt/anaconda3/lib/python3.8/site-packages (from matplotlib->missingno) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in
/opt/anaconda3/lib/python3.8/site-packages (from matplotlib->missingno) (2.8.1)
Requirement already satisfied: cycler>=0.10 in
/opt/anaconda3/lib/python3.8/site-packages (from matplotlib->missingno) (0.10.0)
Requirement already satisfied: pandas>=0.22.0 in
/opt/anaconda3/lib/python3.8/site-packages (from seaborn->missingno) (1.4.4)
Requirement already satisfied: six>=1.5 in /opt/anaconda3/lib/python3.8/site-
packages (from python-dateutil>=2.1->matplotlib->missingno) (1.15.0)
Requirement already satisfied: pytz>=2020.1 in
/opt/anaconda3/lib/python3.8/site-packages (from
pandas>=0.22.0->seaborn->missingno) (2020.1)
```

[2]:
```
!pip install TextBlob
```

```
Requirement already satisfied: TextBlob in /opt/anaconda3/lib/python3.8/site-
packages (0.17.1)
Requirement already satisfied: nltk>=3.1; python_version >= "3" in
/opt/anaconda3/lib/python3.8/site-packages (from TextBlob) (3.5)
Requirement already satisfied: tqdm in /opt/anaconda3/lib/python3.8/site-
packages (from nltk>=3.1; python_version >= "3"->TextBlob) (4.64.1)
Requirement already satisfied: regex in /opt/anaconda3/lib/python3.8/site-
packages (from nltk>=3.1; python_version >= "3"->TextBlob) (2020.6.8)
```

```
Requirement already satisfied: joblib in /opt/anaconda3/lib/python3.8/site-
packages (from nltk>=3.1; python_version >= "3"->TextBlob) (1.1.0)
Requirement already satisfied: click in /opt/anaconda3/lib/python3.8/site-
packages (from nltk>=3.1; python_version >= "3"->TextBlob) (7.1.2)
```

[3]:
```
import pandas as pd
import numpy as np
import scipy
import os
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as ms
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import Binarizer
from IPython.display import display
%matplotlib inline

pd.options.display.float_format = '{:20,.3f}'.format
pd.options.display.max_columns = None
pd.options.display.max_colwidth = 1000
np.set_printoptions(precision=3)
```

0.1 Question 1 - Which type of complaints Department of Housing Preservation and Development should focus first ?

[4]:
```
df_complaint = pd.read_csv("data/fhrw-4uyv.csv")
```

[5]:
```
df_complaint.head()
```

[5]:
```
              created_date  unique_key        complaint_type  \
0  2019-08-23T12:35:54.000    43624241         HEAT/HOT WATER
1  2019-08-23T08:43:58.000    43623659  UNSANITARY CONDITION
2  2019-08-23T09:08:09.000    43624463  UNSANITARY CONDITION
3  2019-08-23T16:36:08.000    43625072           DOOR/WINDOW
4  2019-08-23T11:15:00.000    43623738  UNSANITARY CONDITION

          incident_zip      incident_address       street_name address_type  \
0           10,032.000  560 WEST   160 STREET  WEST   160 STREET      ADDRESS
1           11,208.000    261 MONTAUK AVENUE    MONTAUK AVENUE      ADDRESS
2           10,002.000    125 MADISON STREET    MADISON STREET      ADDRESS
3           11,211.000      525 UNION AVENUE      UNION AVENUE      ADDRESS
4           11,372.000      35-52F 73 STREET         73 STREET      ADDRESS

              city  \
0         NEW YORK
```

```
1         BROOKLYN
2         NEW YORK
3         BROOKLYN
4  Jackson Heights

resolution_description  \
0  The complaint you filed is a duplicate of a condition already reported by
another tenant for a building-wide condition. The original complaint is still
open. HPD may attempt to contact you to verify the correction of the condition
or may conduct an inspection of your unit if the original complainant is not
available for verification.
1
The following complaint conditions are still open. HPD may attempt to contact
you to verify the correction of the condition or may conduct an inspection.
2
The following complaint conditions are still open. HPD may attempt to contact
you to verify the correction of the condition or may conduct an inspection.
3
The following complaint conditions are still open. HPD may attempt to contact
you to verify the correction of the condition or may conduct an inspection.
4
NaN

     borough             latitude            longitude closed_date  \
0  MANHATTAN               40.835              -73.942         NaN
1   BROOKLYN               40.672              -73.878         NaN
2  MANHATTAN               40.712              -73.994         NaN
3   BROOKLYN               40.716              -73.952         NaN
4     QUEENS               40.751              -73.893         NaN

          location_type status
0  RESIDENTIAL BUILDING   Open
1  RESIDENTIAL BUILDING   Open
2  RESIDENTIAL BUILDING   Open
3  RESIDENTIAL BUILDING   Open
4  RESIDENTIAL BUILDING   Open
```

[6]:
```
df_complaint.shape
```

[6]:
```
(5846787, 15)
```

[7]:
```
df_complaint['complaint_type'].value_counts()
```

[7]:
```
HEAT/HOT WATER          1149978
HEATING                  887869
PLUMBING                 702046
GENERAL CONSTRUCTION     500863
```

```
UNSANITARY CONDITION      434830
PAINT - PLASTER           361258
PAINT/PLASTER             340753
ELECTRIC                  303115
NONCONST                  260890
DOOR/WINDOW               199443
WATER LEAK                186913
GENERAL                   145825
FLOORING/STAIRS           135159
APPLIANCE                 109480
HPD Literature Request     52830
SAFETY                     49904
OUTSIDE BUILDING            7015
ELEVATOR                    6397
Unsanitary Condition        5499
CONSTRUCTION                5078
General                     1163
Safety                       424
STRUCTURAL                    16
Plumbing                      11
AGENCY                         9
VACANT APARTMENT               8
Outside Building               6
Appliance                      4
Mold                           1
Name: complaint_type, dtype: int64
```

1 New York City Open Data data file web site indicated that the complaint type "HEAT/HOT Water" was renamed from "HEATING" after 2014. So we should combine these two types into one.

[8]:
```
df_complaint['complaint_type'].replace({'HEAT/HOT WATER': 'HEATING'}, inplace =
↪True)
```

[9]:
```
df_complaint.isnull().sum()
```

[9]:
```
created_date            0
unique_key              0
complaint_type          0
incident_zip        80613
incident_address    52831
street_name         52831
address_type        84779
city                80212
```

```
resolution_description     13190
borough                        0
latitude                   80587
longitude                  80587
closed_date               123460
location_type              52830
status                         0
dtype: int64
```

[10]:
```
type(df_complaint['created_date'][0])
```

[10]:
```
str
```

[11]:
```
df_complaint['created_year'] = df_complaint['created_date'].map(lambda x: x[0:
↪4]).astype(int)
```

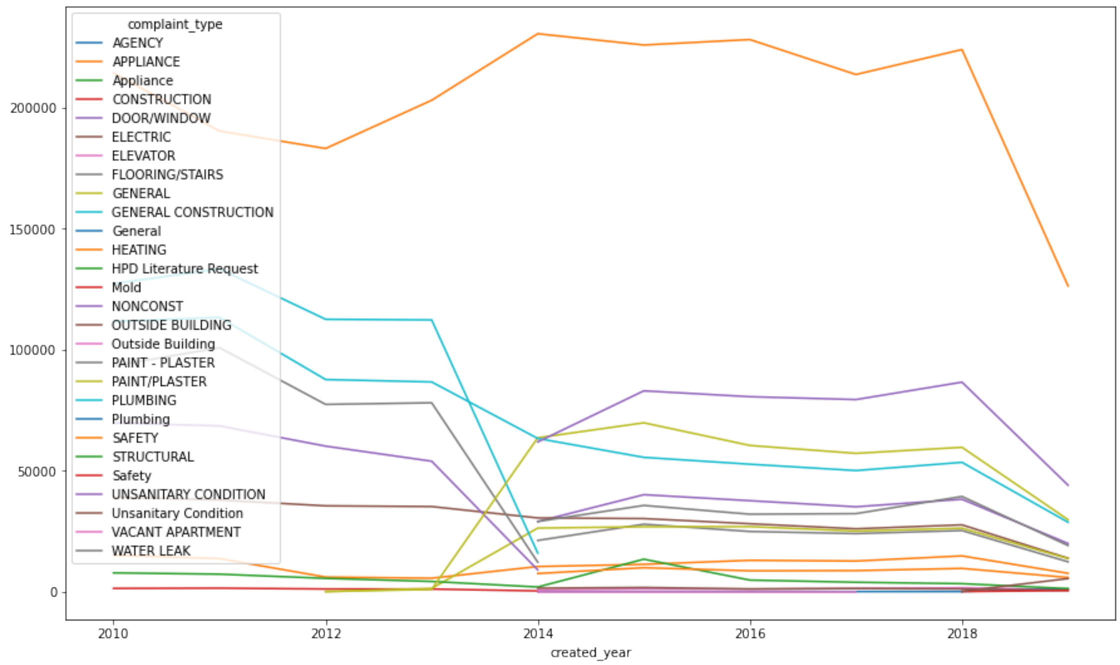
[12]:
```
df_complaint['created_month'] = df_complaint['created_date'].map(lambda x: x[5:
↪7]).astype(int)
```

[13]:
```
df_complaint.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5846787 entries, 0 to 5846786
Data columns (total 17 columns):
 #   Column                  Dtype
---  ------                  -----
 0   created_date            object
 1   unique_key              int64
 2   complaint_type          object
 3   incident_zip            float64
 4   incident_address        object
 5   street_name             object
 6   address_type            object
 7   city                    object
 8   resolution_description  object
 9   borough                 object
 10  latitude                float64
 11  longitude               float64
 12  closed_date             object
 13  location_type           object
 14  status                  object
 15  created_year            int64
 16  created_month           int64
dtypes: float64(3), int64(3), object(11)
memory usage: 758.3+ MB
```

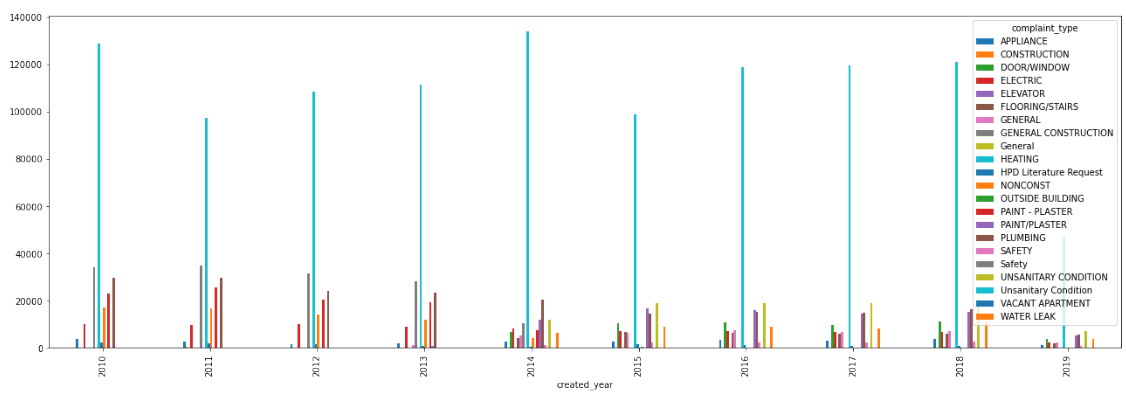
```
[14]: fig, ax = plt.subplots(figsize = (15, 9))
df_complaint.groupby(['created_year', 'complaint_type']).count()['unique_key'].
 ↪unstack().plot(ax = ax)
```

```
[14]: <matplotlib.axes._subplots.AxesSubplot at 0x7f79cb1fb370>
```




```
[15]: fig, ax = plt.subplots(figsize = (22, 7))
df_complaint[df_complaint['created_month'].isin([11, 12, 1])].
 ↪groupby(['created_year', 'complaint_type']).count()['unique_key'].unstack().
 ↪plot.bar(ax = ax)
```

```
[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7f79bd535bb0>
```

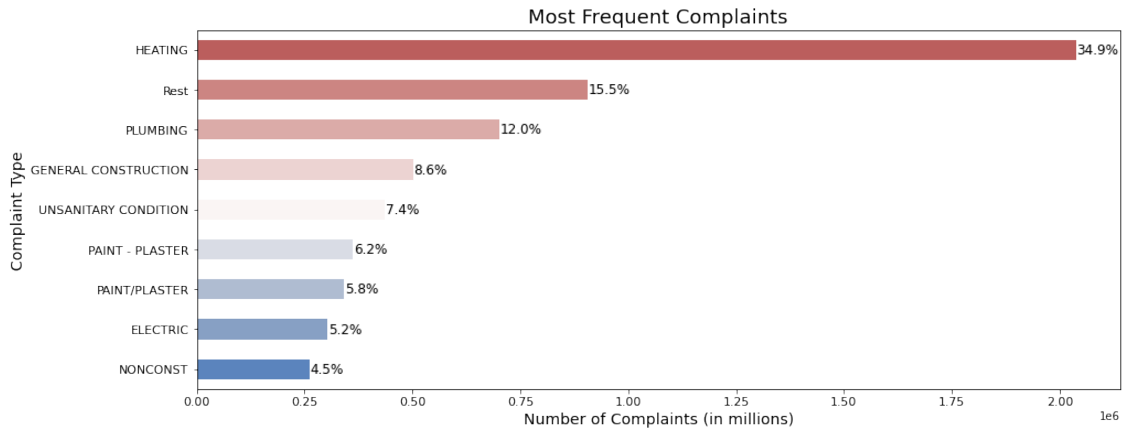


1.0.1 The above chart proves that though there may be little drop in total Heating problem counts from 2015-2018 if we consider the winter months together the number of heating complaints are still increasting between 2015 and 2016 and almost the same between 2016-2018.

[16]:
```
df_complaint['complaint_type'].value_counts().nlargest(10)
```

[16]:
```
HEATING                 2037847
PLUMBING                 702046
GENERAL CONSTRUCTION     500863
UNSANITARY CONDITION     434830
PAINT - PLASTER          361258
PAINT/PLASTER            340753
ELECTRIC                 303115
NONCONST                 260890
DOOR/WINDOW              199443
WATER LEAK               186913
Name: complaint_type, dtype: int64
```

[17]:
```
df_new = df_complaint.copy()

idx = df_new['complaint_type'].value_counts().sort_values().head(20).index
df_new.loc[df_new['complaint_type'].isin(idx), 'complaint_type'] = 'Rest'

df_new = df_new['complaint_type'].value_counts().sort_values()

df_new.plot(kind = 'barh', figsize = (15,6), fontsize = 11,
            color = sns.color_palette("vlag", len(df_new)))
plt.ylabel('Complaint Type', fontsize = 14)
plt.xlabel('Number of Complaints (in millions)', fontsize = 14)
plt.title("Most Frequent Complaints", fontsize = 18)


for index, value in enumerate(df_new):
    label = '{}%'.format(round((value/df_new.sum())*100, 1))
    plt.annotate(label, xy = (value + 2000, index - 0.1), fontsize = 12)
```



Conclusion: Given the above analysis it makes sebse for the Department of Housing Preservation and Development in New York City to first focus on solving Heating complaints among other NYC 311 complaint types.

1.1 Question 2 - Should the Department of Housing Preservation and Development focus on any particular set of Boroughs or Zip Code or Streets (where the complaints are severe) for that specific type of Complaints ?

[18]:
```
df_heating = df_complaint[df_complaint['complaint_type'] == 'HEATING']
```

[19]:
```
df_heating = df_heating.dropna(subset=['latitude', 'longitude'])
```

1.1.1 Visualizing Top "HEATING" complaint by using map of New York

[20]:
```
import folium
from folium.plugins import HeatMap

testCoo=df_heating[['latitude','longitude']].values

NY_COORDINATES = [40.796015,-73.947288]
map_ = folium.Map(location=NY_COORDINATES, zoom_start=12)

HeatMap(testCoo, radius=15, blur=20).add_to(map_)

display(map_)
```

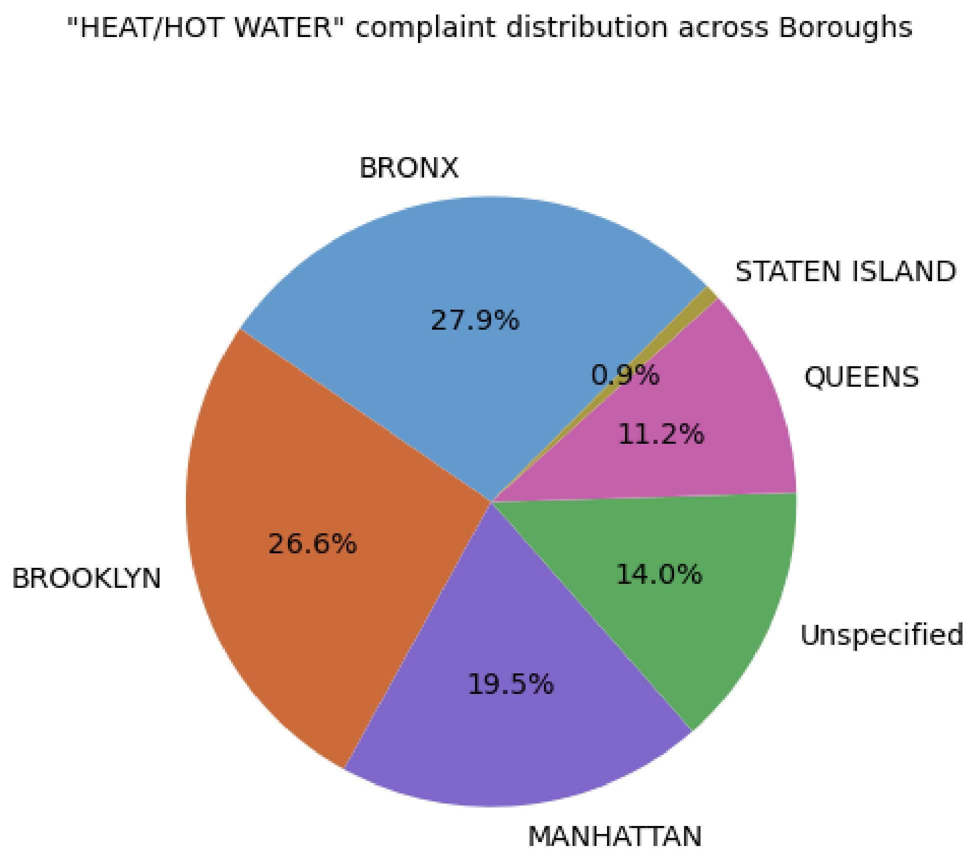
```
<folium.folium.Map at 0x7f79af8abc40>
```

[21]:
```
df_heating['borough'].value_counts()
```

[21]:
```
BRONX            563237
BROOKLYN         536711
MANHATTAN        393330
Unspecified      282648
QUEENS           225729
STATEN ISLAND     17227
Name: borough, dtype: int64
```

2 "HEATING" Complaint distribution across Boroughs

[22]:
```python
colors = ['#639ace','#ca6b39','#7f67ca','#5ba85f','#c360aa','#a7993f','#cc566a']
df_heating['borough'].value_counts().plot(kind='pie',
                                        autopct='%1.1f%%',
                                        startangle=45,
                                        shadow=False,
                                        colors = colors,
                                        figsize = (8,6),
                                       textprops={'fontsize': 14})
plt.ylabel('')
plt.title('"HEAT/HOT WATER" complaint distribution across Boroughs\n', y=1.12,␣
 ↪fontsize=14)
plt.axis('equal')
plt.tight_layout()
plt.show()
```

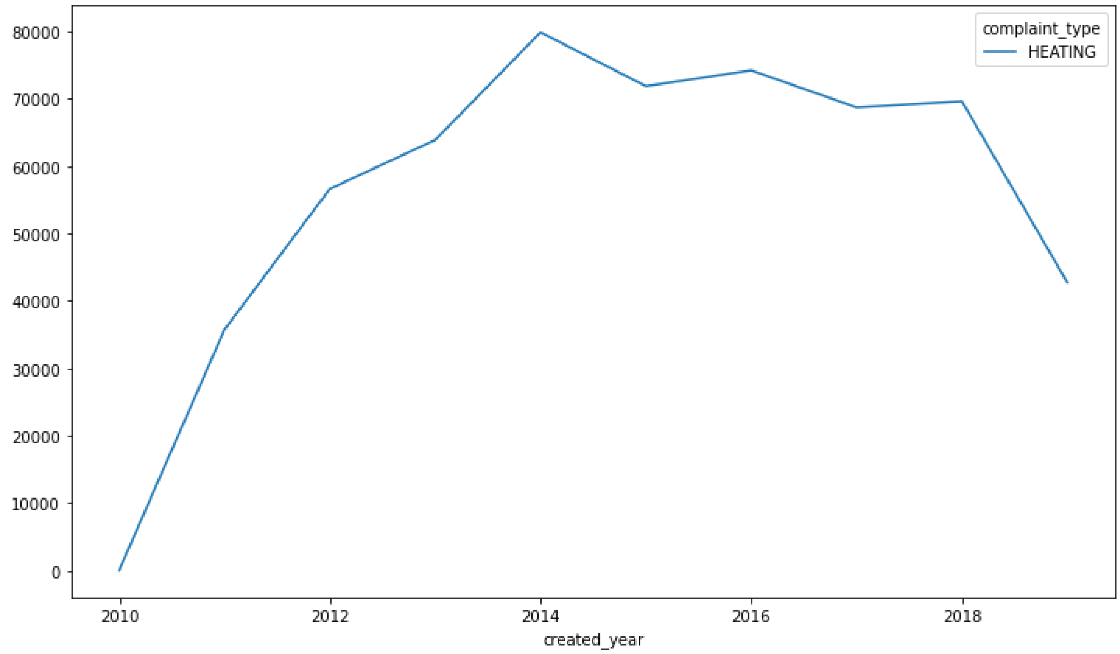


Given the fact that BRONX has majority of Heating Problems, let us focus there. So we are filtering the data further for the borough BRONX

[23]:
```
df_bronx = df_heating[df_heating['borough'] == "BRONX"]
```

[24]:
```
fig, ax = plt.subplots(figsize = (12, 7))
df_bronx.groupby(['created_year', 'complaint_type']).count()['unique_key'].
↪unstack().plot(ax = ax)
```

[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f79b0ecaee0>
```

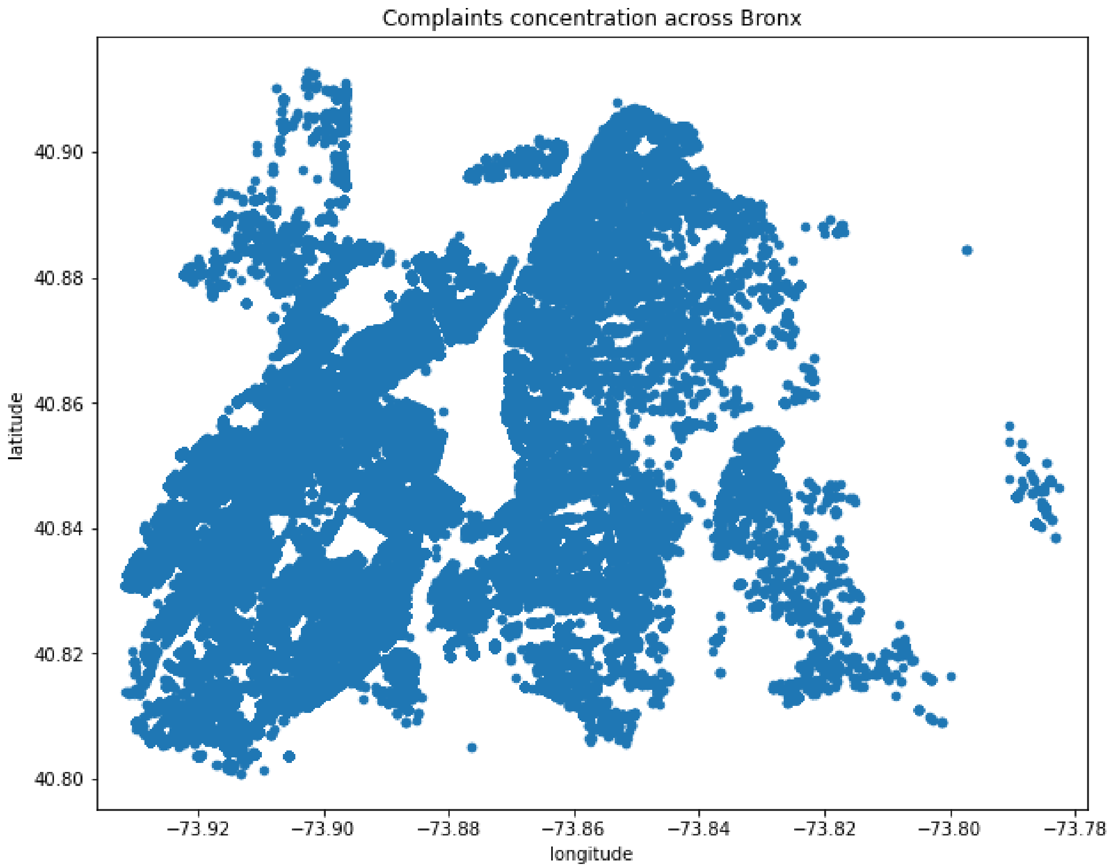


2.0.1 There was a sharp inrease in number of "HEAR/HOT WATER" Complaints from 2010 to 2014 where it reached its peak (approximately 80,000), and then it slightly felt to 70,000 and remained level. According to the graph, the number of "HEAT/HOT WATER" decreased from 70271 in 2018 to 38014 in 2019. onsidering the fact that we do not have full data for 2019, we can not draw any conclusions from this decrease

3 Let's look at complaints concentration across Bronx

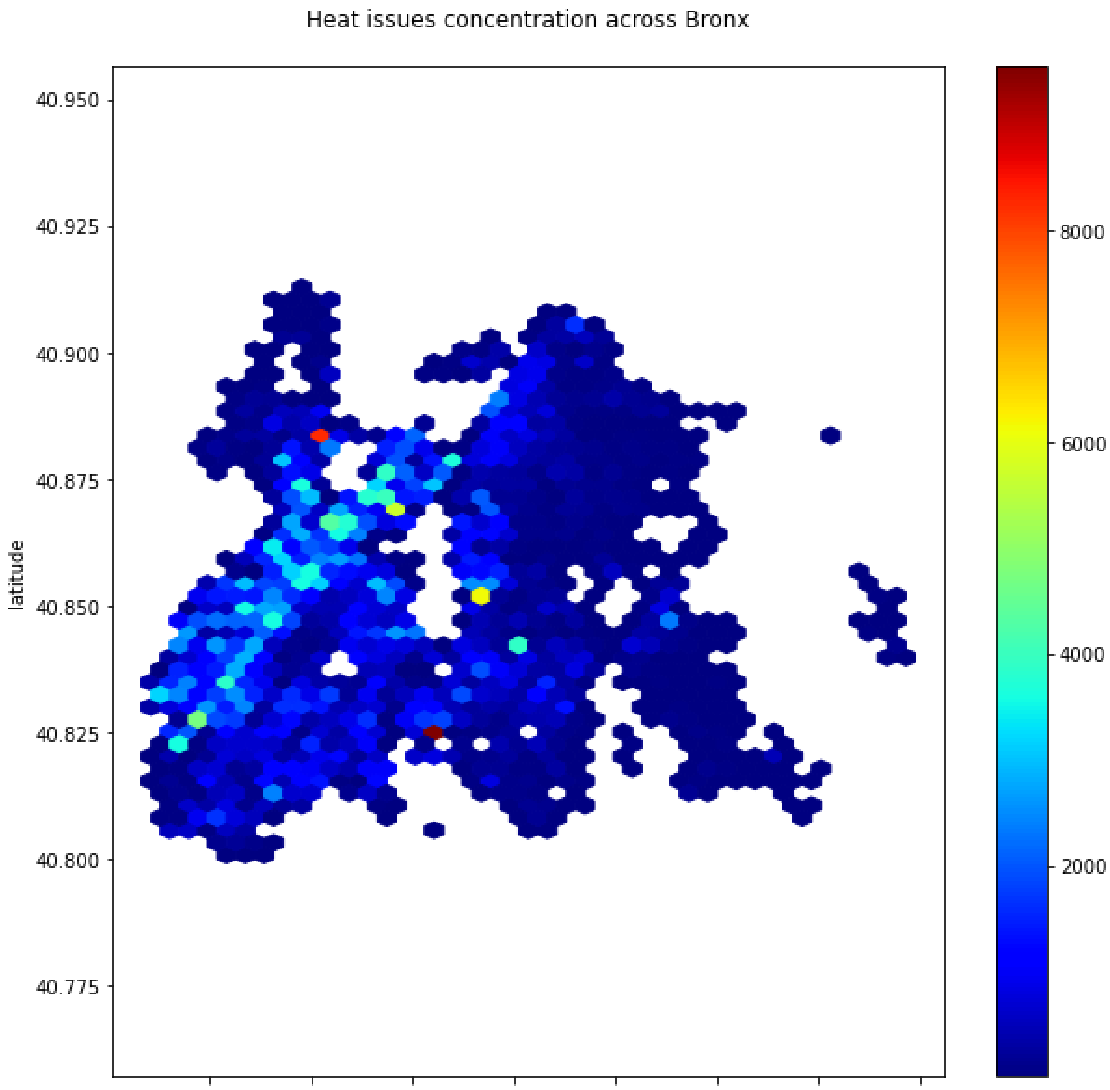
[25]:
```
df_bronx[['longitude', 'latitude']].plot(kind='scatter',
                                          x = 'longitude',
                                          y = 'latitude',
                                          xlim = (min(df_bronx['longitude']),␣
 ↪max(df_bronx['longitude'])),
                                          figsize = (10,8),
                                          title = 'Complaints concentration␣
 ↪across Bronx').axis('equal')
```

[25]: (-73.93913578371806, -73.77515854599136, 40.79524590542191, 40.918469885311644)



[26]:
```
df_bronx.plot(
    kind='hexbin', x='longitude', y='latitude', gridsize=40,title = 'Heat␣
↪issues concentration across Bronx\n',
    colormap='jet', mincnt=1, figsize=(10,10)).axis('equal')
```

[26]: (-73.93913578388204, -73.77515854582738, 40.79524590542191, 40.918469885311644)

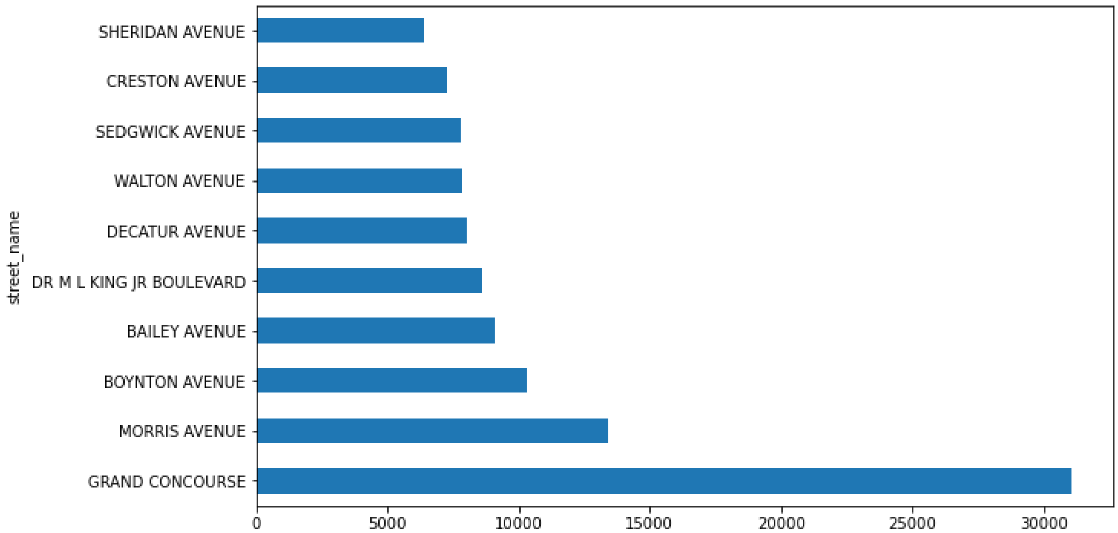


3.0.1 From these graphs, we can understand that there are several places where complaints come from more often.

4 Let's identify "HEAT/HOT WATER" Complaint distibution on street level

[27]:
```
df_bronx.groupby(['street_name']).count()['unique_key'].nlargest(10).plot(kind␣
 ↪= 'barh',
                                                                            figsize␣
 ↪= (10, 6))
```

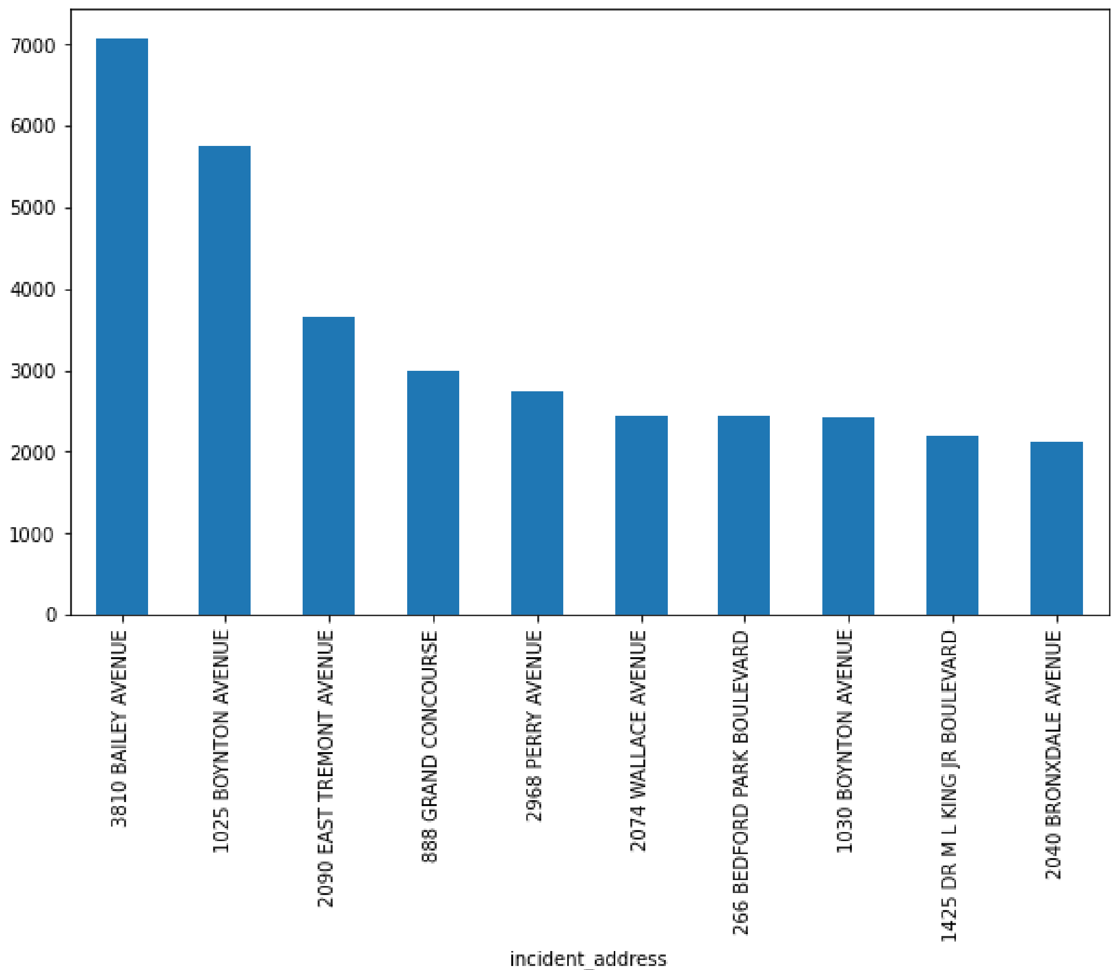
[27]: <matplotlib.axes._subplots.AxesSubplot at 0x7f79b0ed35e0>



There is one street where the complaints are most severe: GRAND CONCOURSE Let's find adresses where the complaints occur more often.

[28]:
```
df_bronx.groupby(['incident_address']).count()['unique_key'].nlargest(10).
↪plot(kind = 'bar',
                                                                              ␣
↪figsize = (10, 6))
```

[28]: <matplotlib.axes._subplots.AxesSubplot at 0x7f79b0f284f0>



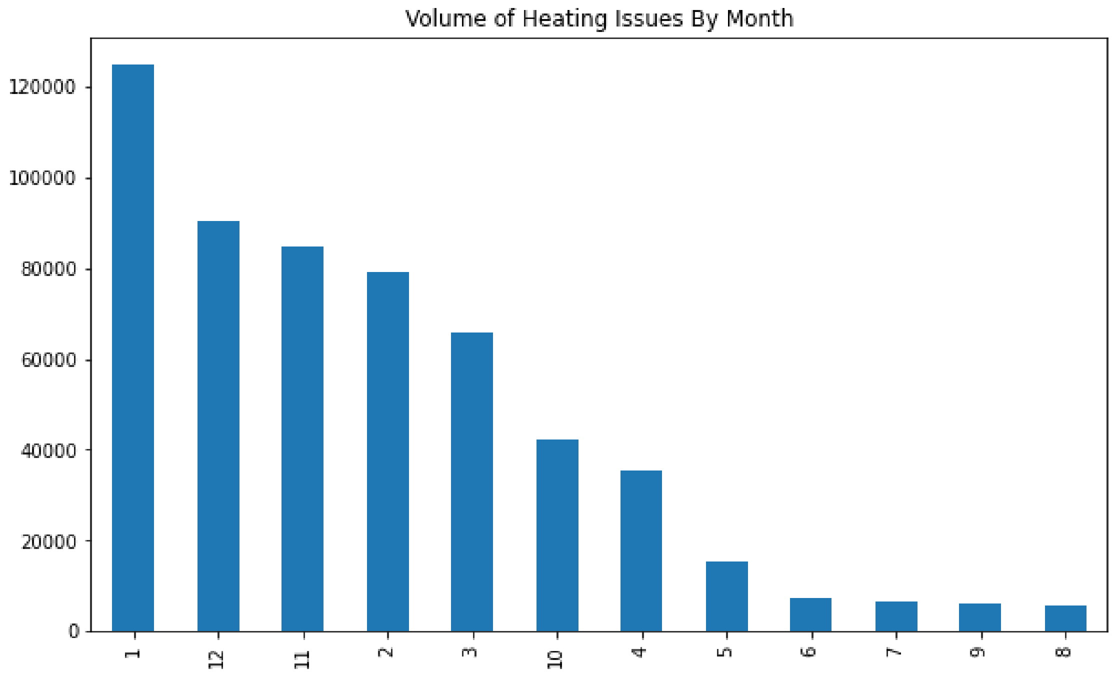
The bar chart shows that there are two addresses where the complaints come from most often: 3810 Bailey Avenue and 1025 Boynton Avenue

4.0.1 Additional insights

Volume of Heat Issue By Month

[29]:
```
df_bronx['created_month'].value_counts().plot(kind = 'bar',
                                               figsize = (10, 6),
                                               title = 'Volume of Heating Issues␣
↪By Month')
```

[29]: <matplotlib.axes._subplots.AxesSubplot at 0x7f79aff8eac0>

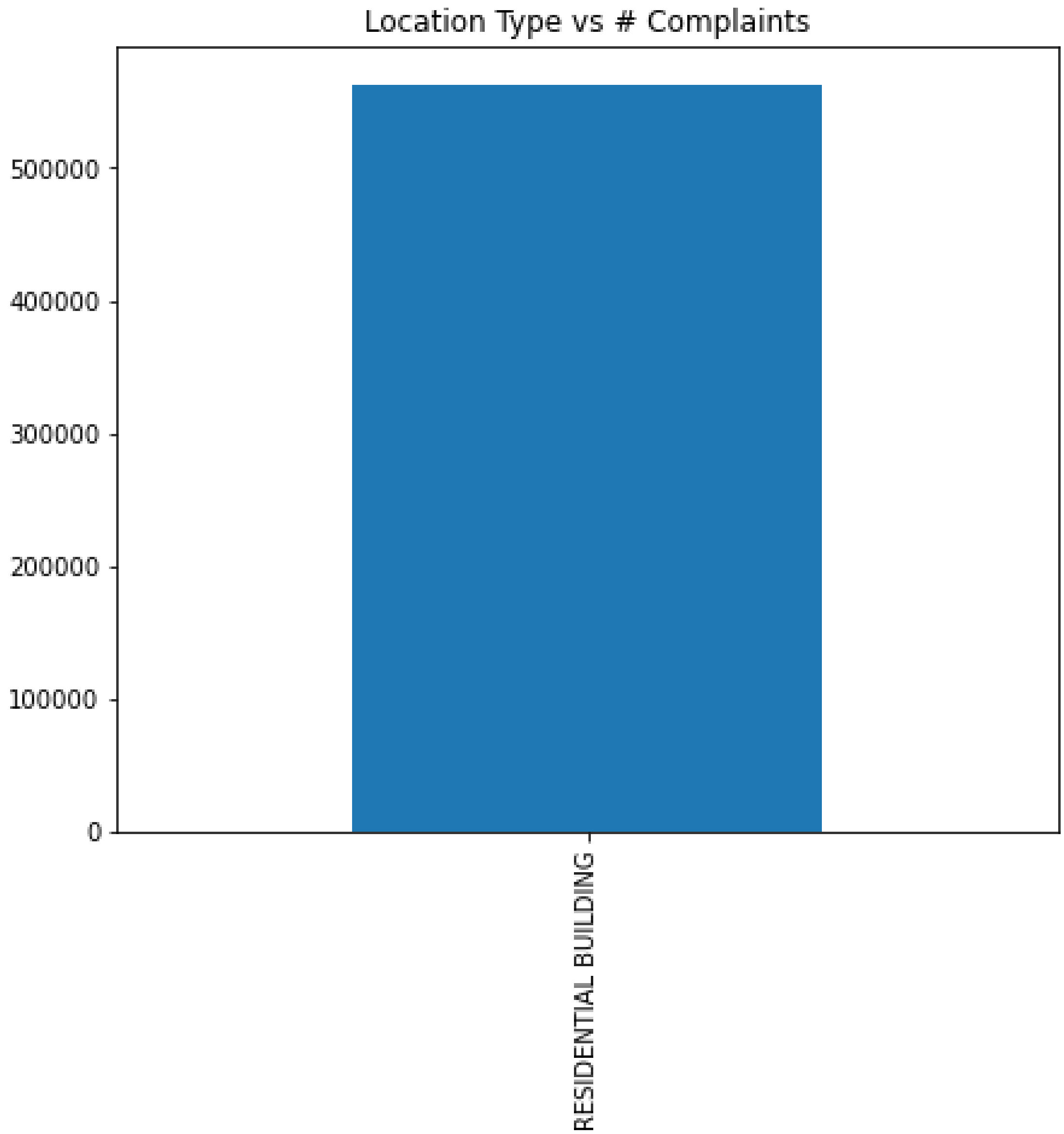


Heating issues are most common in the months of January and December

Location Type vs Number of Heating Complaints

[30]:
```
(df_bronx['location_type'].value_counts()).head(25).plot(kind='bar',
                                                          figsize=(7,6),
                                                          title='Location Type vs␣
 ↪# Complaints')
```

[30]: <matplotlib.axes._subplots.AxesSubplot at 0x7f79aff07fd0>


Location Type vs # Complaints
500000
400000
300000
200000
100000
0
RESIDENTIAL BUILDING

4.1 Conclusion: Given the above analysis it makes sense for the Department of Housing Preservation and Development in New York City to focus on Bronx Borought where "HEATING" complaints are most severe. It was found out that most part of the complaints come from GRAND CONCOURSE street; however, there are two certain adresses where complaints occur most often: 3810 BAILEY AVENUE and 1025 BOYTON AVENUE

4.2 Additional Insight: The most number of "HEATING" complaints came from residential buildings during months of January and December

5 Question 3 - Does the type of Complaints have obvious relationship with any particular characteristic(s) of the Houses ?

For this question, we also need to import the PLUTO dataset for the Bronx. Since we ahave already identifiedthat Bronx has the highest heating complaints we shall only download the data for Bronx borough

[31]:
```python
df_bx = pd.read_csv("data/BX_18v1.csv")
df_bx.head()
```

```
<ipython-input-31-71ae1697db37>:1: DtypeWarning: Columns (19,20,22,23,64,65,80)
have mixed types. Specify dtype option on import or set low_memory=False.
  df_bx = pd.read_csv("data/BX_18v1.csv")
```

[31]:
```
  Borough  Block  Lot   CD              CT2010            CB2010  \
0      BX   2260    1  201              19.000         1,022.000
1      BX   2260    4  201              19.000         1,022.000
2      BX   2260   10  201              19.000         1,022.000
3      BX   2260   17  201              19.000         1,022.000
4      BX   2260   18  201              19.000         1,022.000

            SchoolDist              Council          ZipCode FireComp  \
0                7.000                8.000       10,454.000     L029
1                7.000                8.000       10,454.000     L029
2                7.000                8.000       10,454.000     L029
3                7.000                8.000       10,454.000     L029
4                7.000                8.000       10,454.000     L029

            PolicePrct  HealthCenterDistrict          HealthArea  \
0               40.000                23.000           4,700.000
1               40.000                23.000           4,700.000
2               40.000                23.000           4,700.000
3               40.000                23.000           4,700.000
4               40.000                23.000           4,700.000

             SanitBoro       SanitDistrict SanitSub                Address  \
0                2.000               1.000       2A  122 BRUCKNER BOULEVARD
1                2.000               1.000       2A  126 BRUCKNER BOULEVARD
```

```
2              2.000                  1.000       2A  138 BRUCKNER BOULEVARD
3              2.000                  1.000       2A  144 BRUCKNER BOULEVARD
4              2.000                  1.000       2A  148 BRUCKNER BOULEVARD

  ZoneDist1 ZoneDist2 ZoneDist3 ZoneDist4 Overlay1 Overlay2 SPDist1  SPDist2  \
0  M1-5/R8A       NaN       NaN       NaN      NaN      NaN    MX-1      NaN
1  M1-5/R8A       NaN       NaN       NaN      NaN      NaN    MX-1      NaN
2  M1-5/R8A       NaN       NaN       NaN      NaN      NaN    MX-1      NaN
3  M1-5/R8A       NaN       NaN       NaN      NaN      NaN    MX-1      NaN
4  M1-5/R8A       NaN       NaN       NaN      NaN      NaN    MX-1      NaN

   SPDist3  LtdHeight SplitZone BldgClass              LandUse  Easements  \
0      NaN        NaN         N        Z9                  NaN          0
1      NaN        NaN         N        G5                7.000          0
2      NaN        NaN         N        F5                6.000          0
3      NaN        NaN         N        C1                2.000          0
4      NaN        NaN         N        C7                2.000          0

  OwnerType              OwnerName  LotArea  BldgArea  ComArea  ResArea  \
0       NaN  122 BRUCKNER PARTNERS    15000         0        0        0
1       NaN  24 INDIAN HEAD HOLDIN    13770       752      752        0
2         P            ANJOST CORP    35000     39375    39375        0
3       NaN       144 BRUCKNER LLC     2500     12500        0    12500
4         P       148 BRUCKNER LLC     1875      8595     1719     6876

   OfficeArea  RetailArea  GarageArea  StrgeArea  FactryArea  OtherArea  \
0           0           0           0          0           0          0
1         272           0           0        480           0          0
2           0           0           0          0       39375          0
3           0           0           0          0           0          0
4           0        1719           0          0           0          0

   AreaSource  NumBldgs            NumFloors  UnitsRes  UnitsTotal  \
0           7         1                0.000         0           0
1           2         2                1.000         0           1
2           2         1                2.000         0           1
3           2         1                5.000        15          15
4           2         1                5.000         8          10

             LotFront             LotDepth            BldgFront  \
0              75.000              200.000                0.000
1             137.580              100.000               16.000
2             175.000              200.000              175.000
3              25.000              100.000               25.000
4              25.000               75.000               25.000

            BldgDepth  Ext             ProxCode IrrLotCode  \
```

```
0             0.000  NaN                0.000         N
1            16.000  NaN                0.000         N
2           200.000  NaN                0.000         N
3            85.000  NaN                0.000         N
4            70.000  NaN                0.000         N

             LotType            BsmtCode  AssessLand  AssessTot  \
0              3.000               0.000      130500     161100
1              5.000               0.000      117000     326700
2              4.000               0.000      153000     879300
3              5.000               0.000       51300     332550
4              3.000               2.000       17490     125304

   ExemptLand  ExemptTot  YearBuilt  YearAlter1  YearAlter2 HistDist Landmark  \
0           0          0          0           0           0      NaN      NaN
1           0          0       1931        1994           0      NaN      NaN
2           0          0       1931           0           0      NaN      NaN
3           0          0       1931        2001           0      NaN      NaN
4           0      52349       1920        2009           0      NaN      NaN

            BuiltFAR            ResidFAR            CommFAR  \
0              0.000               6.020              5.000
1              0.050               6.020              5.000
2              1.130               6.020              5.000
3              5.000               6.020              5.000
4              4.580               6.020              5.000

            FacilFAR  BoroCode         BBL  CondoNo  Tract2010  \
0              6.500         2  2022600001        0         19
1              6.500         2  2022600004        0         19
2              6.500         2  2022600010        0         19
3              6.500         2  2022600017        0         19
4              6.500         2  2022600018        0         19

              XCoord              YCoord ZoneMap ZMCode  Sanborn  \
0      1,005,957.000         232,162.000      6b    NaN  209S016
1      1,006,076.000         232,156.000      6b    NaN  209S016
2      1,006,187.000         232,036.000      6b    NaN  209S016
3      1,006,299.000         232,033.000      6b    NaN  209S016
4      1,006,363.000         232,040.000      6b    NaN  209S016

              TaxMap EDesigNum              APPBBL APPDate  PLUTOMapID  \
0         20,901.000     E-143               0.000     NaN           1
1         20,901.000     E-143               0.000     NaN           1
2         20,901.000     E-143               0.000     NaN           1
3         20,901.000     E-143               0.000     NaN           1
4         20,901.000     E-143               0.000     NaN           1
```

```
  FIRM07_FLAG  PFIRM15_FLAG Version
0         NaN           NaN    18V1
1         NaN           NaN    18V1
2         NaN           NaN    18V1
3         NaN           NaN    18V1
4         NaN           NaN    18V1
```

[32]:
```
df_bx = df_bx[['Address','BldgArea','BldgDepth','BuiltFAR',
               'CommFAR','FacilFAR',
               'Lot','LotArea','LotDepth','NumBldgs','NumFloors',
               'OfficeArea','ResArea','ResidFAR','RetailArea',
               'YearBuilt','YearAlter1','ZipCode', 'YCoord', 'XCoord'
               ]]
```

[33]:
```
df_bx.head(2)
```

[33]:
```
                  Address  BldgArea            BldgDepth             BuiltFAR  \
0  122 BRUCKNER BOULEVARD         0                0.000                0.000
1  126 BRUCKNER BOULEVARD       752               16.000                0.050

               CommFAR             FacilFAR  Lot  LotArea  \
0                5.000                6.500    1    15000
1                5.000                6.500    4    13770

              LotDepth  NumBldgs            NumFloors  OfficeArea  ResArea  \
0              200.000         1                0.000           0        0
1              100.000         2                1.000         272        0

              ResidFAR  RetailArea  YearBuilt  YearAlter1  \
0                6.020           0          0           0
1                6.020           0       1931        1994

               ZipCode               YCoord                XCoord
0           10,454.000          232,162.000         1,005,957.000
1           10,454.000          232,156.000         1,006,076.000
```

[34]:
```
df_bx['ZipCode']                = df_bx['ZipCode'].fillna(0).astype(int)
df_bx['NumFloors']              = df_bx['NumFloors'].fillna(0).astype(int)
df_bx['NumBldgs']               = df_bx['NumBldgs'].fillna(0).astype(int)
df_bx['LotDepth']               = df_bx['LotDepth'].fillna(0).astype(int)
df_bx['BldgDepth']              = df_bx['BldgDepth'].fillna(0).astype(int)
df_bx['BldgArea']               = df_bx['BldgArea'].fillna(0).astype(int)
df_bx['Lot']                    = df_bx['Lot'].fillna(0).astype(int)
df_bx['LotArea']                = df_bx['LotArea'].fillna(0).astype(int)
df_bx['BuiltFAR']               = df_bx['BuiltFAR'].fillna(0).astype(int)
df_bx['ResidFAR']               = df_bx['ResidFAR'].fillna(0).astype(int)
```

```
df_bx['CommFAR']                = df_bx['CommFAR'].fillna(0).astype(int)
df_bx['FacilFAR']               = df_bx['FacilFAR'].fillna(0).astype(int)
df_bx['OfficeArea']             = df_bx['OfficeArea'].fillna(0).astype(int)
df_bx['ResArea']                = df_bx['ResArea'].fillna(0).astype(int)
df_bx['RetailArea']             = df_bx['RetailArea'].fillna(0).astype(int)
df_bx['XCoord']                 = df_bx['XCoord'].fillna(0).astype(int)
df_bx['YCoord']                 = df_bx['YCoord'].fillna(0).astype(int)
```

Join PLUTO and Complaint datasets based on address Let's see how many addresses in complaints can be found in PLUTO dataset

[35]:
```
print(df_bronx.shape)
print(df_bx.shape)
```

```
(563237, 17)
(89854, 20)
```

[36]:
```
perc=((df_bronx['incident_address'].isin(df_bx['Address']).sum())/df_bronx.
 ↪shape[0])*100

print('Complaints incident address found in PLUTO dataset percentage: {:.3f}%'.
 ↪format(perc))
```

```
Complaints incident address found in PLUTO dataset percentage: 79.961%
```

There's 79.961% Bronx complaint incident adresses can be found in PLUTO dataset

5.0.1 Removing Duplicates

[37]:
```
df_pluto_bx = df_bx[df_bx["Address"].notnull()]
print(df_pluto_bx.shape)
df_pluto_bx = df_pluto_bx.drop_duplicates(subset=['Address','ZipCode'],␣
 ↪keep='first')
print(df_pluto_bx.shape)
```

```
(89785, 20)
(87384, 20)
```

Next let us aggregate the number of Complaints at each address level so that we can try finding out if there is any relationship between the number of complaints and building characteristics

[38]:
```
df_complaint_aggr = df_bronx.groupby(['incident_address'], as_index = False).
 ↪agg({'unique_key': 'count'})
df_complaint_aggr.columns = ["Address", 'complaintCnt']
```

[39]:
```
df_complaint_aggr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21998 entries, 0 to 21997
Data columns (total 2 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   Address       21998 non-null  object
 1   complaintCnt  21998 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 343.8+ KB
```

Now we merge the data between building dataset and aggregated complaint dataset

[40]:
```
df_combo = pd.merge(df_bx, df_complaint_aggr, left_on = ['Address'], right_on =␣
 ↪['Address'], how = "left")
print(df_combo.shape)
print(df_complaint_aggr.shape)
print(df_bx.shape)
```

```
(89854, 21)
(21998, 2)
(89854, 20)
```

[41]:
```
df_combo.head(2)
```

[41]:
```
                   Address  BldgArea  BldgDepth  BuiltFAR  CommFAR  FacilFAR  \
0  122 BRUCKNER BOULEVARD         0          0         0        5         6   
1  126 BRUCKNER BOULEVARD       752         16         0        5         6   

   Lot  LotArea  LotDepth  NumBldgs  NumFloors  OfficeArea  ResArea  ResidFAR  \
0    1    15000       200         1          0           0        0         6   
1    4    13770       100         2          1         272        0         6   

   RetailArea  YearBuilt  YearAlter1  ZipCode  YCoord   XCoord  complaintCnt  
0           0          0           0    10454  232162  1005957           NaN  
1           0       1931        1994    10454  232156  1006076           NaN  
```

[42]:
```
df_combo['complaintCnt'] = df_combo.complaintCnt.fillna('0').astype(int)
```

Let us filter out the data that has YearBuilt as 0 to remove bad data

[43]:
```
df_combo= df_combo[df_combo['YearBuilt'] > 0]
print(df_combo.shape)
```

```
(83487, 21)
```

Let us create 2 new features Building Age from YearBuilt and Builkding Altered Age from YearAlter1

```
[44]: df_combo['bldg_age'] = df_combo.apply(lambda x: 2022 - x['YearBuilt'], axis = 1)
print(df_combo.shape)
```

```
(83487, 22)
```

```
[45]: df_combo['bldg_alt_age'] = df_combo.apply(lambda x: 2018 - x['YearAlter1'] if␣
 ↪x['YearAlter1'] > 0 else x['bldg_age'], axis = 1)
print(df_combo.shape)
```

```
(83487, 23)
```

```
[46]: df_combo.head(2)
```

```
[46]:                     Address  BldgArea  BldgDepth  BuiltFAR  CommFAR  FacilFAR  \
1  126 BRUCKNER BOULEVARD       752         16         0        5         6
2  138 BRUCKNER BOULEVARD     39375        200         1        5         6

   Lot  LotArea  LotDepth  NumBldgs  NumFloors  OfficeArea  ResArea  ResidFAR  \
1    4    13770       100         2          1         272        0         6
2   10    35000       200         1          2           0        0         6

   RetailArea  YearBuilt  YearAlter1  ZipCode  YCoord   XCoord  complaintCnt  \
1           0       1931        1994    10454  232156  1006076             0
2           0       1931           0    10454  232036  1006187             0

   bldg_age  bldg_alt_age
1        91            24
2        91            91
```

Now let us divide the number of complaints to 4 ranges -

a) 0 (No Complaint),
b) 1 (Number of Complaints between 1 to 10)
c) 10 (Number of Complaints between 11 to 100) and
d) 100 (Number of Complaints above 100)

```
[47]: def cnt_range(cnt):
    if cnt <= 0:
        rng = 0
    elif cnt <= 10 and cnt > 0:
        rng = 1
    elif cnt <= 100 and cnt > 10:
        rng = 10
    elif cnt > 100:
        rng = 100
    else:
        rng = 10
    return rng
```

[48]:
```
df_combo['complaint_range'] = df_combo['complaintCnt'].apply(lambda x:␣
↪cnt_range(x))
df_combo.shape
```

[48]:
```
(83487, 24)
```

[49]:
```
df_combo.groupby(['complaint_range']).count()
```

[49]:
```
                 Address  BldgArea  BldgDepth  BuiltFAR  CommFAR  FacilFAR  \
complaint_range
0                  66738     66739      66739     66739    66739     66739
1                  11189     11189      11189     11189    11189     11189
10                  4537      4537       4537      4537     4537      4537
100                 1022      1022       1022      1022     1022      1022

                   Lot  LotArea  LotDepth  NumBldgs  NumFloors  OfficeArea  \
complaint_range
0                66739    66739     66739     66739      66739       66739
1                11189    11189     11189     11189      11189       11189
10                4537     4537      4537      4537       4537        4537
100               1022     1022      1022      1022       1022        1022

                 ResArea  ResidFAR  RetailArea  YearBuilt  YearAlter1  \
complaint_range
0                  66739     66739       66739      66739       66739
1                  11189     11189       11189      11189       11189
10                  4537      4537        4537       4537        4537
100                 1022      1022        1022       1022        1022

                 ZipCode  YCoord  XCoord  complaintCnt  bldg_age  bldg_alt_age
complaint_range
0                  66739   66739   66739         66739     66739         66739
1                  11189   11189   11189         11189     11189         11189
10                  4537    4537    4537          4537      4537          4537
100                 1022    1022    1022          1022      1022          1022
```

The above table clearly shows that 80% of the addresses are not having any Heating Complaints and 10% of them have complaints less than 10 in last 8 years

So let us try to investigate if there is any relationship with number of complaints and the characteristic of the buildings

[50]:
```
df_combo_sub = df_combo[['Address','BldgArea','BldgDepth','BuiltFAR',
                'CommFAR','FacilFAR',
                'Lot','LotArea','LotDepth','NumBldgs','NumFloors',
                'OfficeArea','ResArea','ResidFAR','RetailArea',
                'bldg_age','bldg_alt_age','ZipCode', 'YCoord', 'XCoord',
```

```
                'complaintCnt', 'complaint_range']]
df_combo_sub.shape
```

[50]: (83487, 22)

Let us move some fields out of feature list like YearBuilt, YearAlter1 which we are already taking care of as bld_age and bld_alt_age

[51]:
```
feats = df_combo.columns.to_list()
response = 'complaint_range'
feats.remove(response)
feats.remove('complaintCnt')
feats.remove('Address')
feats.remove('YearBuilt')
feats.remove('YearAlter1')

X = df_combo[feats]
Y = df_combo[response]
```

Pearson's correlation

[52]:
```
corr_p = df_combo_sub.corr()['complaint_range'].sort_values(ascending = False)

display(corr_p.to_frame().style.background_gradient(cmap = 'Reds', axis = 0))
```

<pandas.io.formats.style.Styler at 0x7f79bd380610>

We notice weak positive linear correlation (coefficient between 0.20 and 0.50) between complaint_range and building characteristics, such as NumFloors, BuiltFar, ResidFAR, and BldgDepth.

Spearman's correlation

[53]:
```
corr_p = df_combo_sub.corr(method = 'spearman')['complaint_range'].
 ↪sort_values(ascending = False)

display(corr_p.to_frame().style.background_gradient(cmap = 'Reds', axis = 0))
```

<pandas.io.formats.style.Styler at 0x7f79bd3807f0>

The complaint_range of "HEATING" complaints has a weak (absolute correlation coefficient 0.20-0.39) correlation with FacilFar, ResidFar, BldgDepth, BldgArea, NumFloors, XCoord, and a moderate (absolute correlation coefficient 0.40-0.59) correlation with ResArea, and BuiltFar

Next, we validate this with few more approaches.Let's try to find Feature Importance using Ensemble based techniques like Random Forest and XGBoost

[54]:
```
from sklearn.metrics import mean_squared_error, r2_score, precision_score, \
    recall_score, confusion_matrix, classification_report, \
```

```
    accuracy_score, f1_score
from sklearn.metrics import confusion_matrix, accuracy_score, roc_curve,␣
 ↪roc_auc_score
from sklearn.model_selection import train_test_split, cross_val_score,␣
 ↪cross_val_predict, StratifiedKFold, learning_curve
from sklearn.preprocessing import RobustScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import Pipeline
```

[55]:
```
X, X_test, y, y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

[56]:
```
scaler = RobustScaler()
```

[57]:
```
clsfRF = RandomForestClassifier(n_estimators=100, random_state=5)
pipeRFC = Pipeline(steps=[('scaler', scaler), ('classification', clsfRF)])
```

[58]:
```
pipeRFC.fit(X, y)
```

[58]:
```
Pipeline(steps=[('scaler', RobustScaler()),
                ('classification', RandomForestClassifier(random_state=5))])
```

[59]:
```
model = pipeRFC.steps[1][1]
model
```

[59]:
```
RandomForestClassifier(random_state=5)
```

[60]:
```
total_importance = sum(model.feature_importances_)
col_names = X.columns.tolist()
feat_importance = model.feature_importances_

most_imp_ml_features = pd.DataFrame()
most_imp_ml_features['name'] = col_names
most_imp_ml_features['importance'] = feat_importance
most_imp_ml_features['percentage_importance'] =␣
 ↪most_imp_ml_features['importance'].apply(lambda x: 1.0 * x /␣
 ↪total_importance)

print(most_imp_ml_features.percentage_importance.sum())
```

```
1.0000000000000002
```

[61]:
```
most_imp_ml_features = most_imp_ml_features.
 ↪sort_values(by='importance',ascending=False)
most_imp_ml_features[['name','percentage_importance']].head(20)
```

[61]:

	name	percentage_importance
11	ResArea	0.136

```
16         XCoord              0.106
15         YCoord              0.104
5             Lot              0.096
0        BldgArea              0.093
6         LotArea              0.070
1       BldgDepth              0.067
9       NumFloors              0.054
18   bldg_alt_age              0.047
17       bldg_age              0.047
7        LotDepth              0.045
2        BuiltFAR              0.039
14        ZipCode              0.028
12       ResidFAR              0.026
4        FacilFAR              0.014
13     RetailArea              0.011
8        NumBldgs              0.010
10     OfficeArea              0.004
3         CommFAR              0.004
```

[62]:
```python
from xgboost import XGBRegressor
```

[63]:
```python
regXGBC = XGBRegressor(observation = 'multi:softmax')
pipeXGBC = Pipeline(steps=[('scaler', scaler), ('regression', regXGBC)])
```

[64]:
```python
pipeXGBC.fit(X, y)
```

```
[00:15:37] WARNING: /Users/runner/work/xgboost/xgboost/python-
package/build/temp.macosx-10.9-x86_64-3.7/xgboost/src/learner.cc:627:
Parameters: { "observation" } might not be used.

  This could be a false alarm, with some parameters getting used by language
bindings but
  then being mistakenly passed down to XGBoost core, or some parameter actually
being used
  but getting flagged wrongly here. Please open an issue if you find any such
cases.
```

[64]:
```
Pipeline(steps=[('scaler', RobustScaler()),
                ('regression',
                 XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
                              colsample_bylevel=1, colsample_bynode=1,
                              colsample_bytree=1, early_stopping_rounds=None,
                              enable_categorical=False, eval_metric=None,
                              gamma=0, gpu_id=-1, grow_policy='depthwise',
                              importance_type=None, interaction_constraints='',
```

```
                                  learning_rate=0.300000012, max_bin=256,
                                  max_cat_to_onehot=4, max_delta_step=0,
                                  max_depth=6, max_leaves=0, min_child_weight=1,
                                  missing=nan, monotone_constraints='()',
                                  n_estimators=100, n_jobs=0, num_parallel_tree=1,
                                  observation='multi:softmax', predictor='auto',
                                  random_state=0, reg_alpha=0, …))])
```

[65]:
```
modelXGB = pipeXGBC.steps[1][1]
modelXGB
```

[65]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwise',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight=1,
             missing=nan, monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, observation='multi:softmax', predictor='auto',
             random_state=0, reg_alpha=0, …)
```

[66]:
```
total_importanceXGB = sum(modelXGB.feature_importances_)
col_names = X.columns.tolist()
feat_importance = modelXGB.feature_importances_

most_imp_ml_features = pd.DataFrame()
most_imp_ml_features['name'] = col_names
most_imp_ml_features['importance'] = feat_importance
most_imp_ml_features['percentage_importance'] =␣
↪most_imp_ml_features['importance'].apply(lambda x: 1.0 * x /␣
↪total_importanceXGB)

print(most_imp_ml_features.percentage_importance.sum())
```

```
1.0
```

[67]:
```
most_imp_ml_features = most_imp_ml_features.
↪sort_values(by='importance',ascending=False)
most_imp_ml_features[['name','percentage_importance']].head(20)
```

[67]:

	name	percentage_importance
11	ResArea	0.168
18	bldg_alt_age	0.075
15	YCoord	0.068
16	XCoord	0.061
6	LotArea	0.060

```
17       bldg_age                 0.056
7        LotDepth                 0.055
14        ZipCode                 0.048
8        NumBldgs                 0.047
5             Lot                 0.046
13     RetailArea                 0.045
4         FacilFAR                0.039
10     OfficeArea                 0.039
1       BldgDepth                 0.039
12        ResidFAR                0.036
3          CommFAR                0.034
2         BuiltFAR                0.031
0         BldgArea                0.029
9        NumFloors                0.024
```

Both of Random Forest and XGBoost shows some relationship between the complaint_range and Building characteristics like ResArea, bldg_age, BuiltFar, etc.

5.1 Concluding Remarks - Some of the building characteristics like ResArea', 'NumFloors', 'BuiltFAR', 'BldgArea', 'BldgDepth', and 'LotArea' seems to have some relationship with Number of Heating Complaints as verified by two diffent algorithms.

6 Question 4 - Can a predictive model be built to predict possibility of Complaints of same type in future ?

[68]:
```python
df_model = df_combo.copy()
df_model.head(5)
```

[68]:
```
                    Address  BldgArea  BldgDepth  BuiltFAR  CommFAR  FacilFAR  \
1  126 BRUCKNER BOULEVARD       752         16         0        5         6
2  138 BRUCKNER BOULEVARD     39375        200         1        5         6
3  144 BRUCKNER BOULEVARD     12500         85         5        5         6
4  148 BRUCKNER BOULEVARD      8595         70         4        5         6
6     519 EAST 132 STREET      5316        100         0        5         6

   Lot  LotArea  LotDepth  NumBldgs  NumFloors  OfficeArea  ResArea  ResidFAR  \
1    4    13770       100         2          1         272        0         6
2   10    35000       200         1          2           0        0         6
3   17     2500       100         1          5           0    12500         6
4   18     1875        75         1          5           0     6876         6
6   34     8700       100         2          1           0        0         6

   RetailArea  YearBuilt  YearAlter1  ZipCode  YCoord   XCoord  complaintCnt  \
1           0       1931        1994    10454  232156  1006076             0
2           0       1931           0    10454  232036  1006187             0
```

```
3          0       1931        2001    10454   232033   1006299              2
4       1719       1920        2009    10454   232040   1006363             13
6          0       1931           0    10454   232055   1006046              0

   bldg_age  bldg_alt_age  complaint_range
1        91            24                0
2        91            91                0
3        91            17                1
4       102             9               10
6        91            91                0
```

[69]:
```python
df_model.isnull().sum().sort_values(ascending = False).head(5)
```

[69]:
```
Address         1
BldgArea        0
bldg_alt_age    0
bldg_age        0
complaintCnt    0
dtype: int64
```

[70]:
```python
df_model.dropna(inplace = True)
```

Since our target is to be able to predict possibility of 'heating' complaints in the future, we make new binary column 'target' indicating whether an address has 'heating' complaints

[71]:
```python
df_model['target'] = np.where(df_model['complaint_range'] >= 1, 1, 0)
```

[72]:
```python
df_model['target'].value_counts()
```

[72]:
```
0    66738
1    16748
Name: target, dtype: int64
```

6.0.1 Feature Selection

We select features with an absolute Spearman's correlation coefficient greater than ~0.15

[137]:
```python
y = df_model['target']
X = df_model[['ResArea', 'BuiltFAR', 'NumFloors',
              'BldgArea', 'BldgDepth', 'ResidFAR',
              'FacilFAR', 'LotArea', 'bldg_age']]
```

6.0.2 Splittign into test/train sets

[138]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 42)
```

```
print('The shape of X_train is ', X_train.shape)
print('The shape of y_train is ', y_train.shape)
print('The shape of X_test is ', X_test.shape)
print('The shape of y_test is ', y_test.shape)
```

```
The shape of X_train is  (66788, 9)
The shape of y_train is  (66788,)
The shape of X_test is  (16698, 9)
The shape of y_test is  (16698,)
```

[139]:
```
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

6.0.3 Random Forest

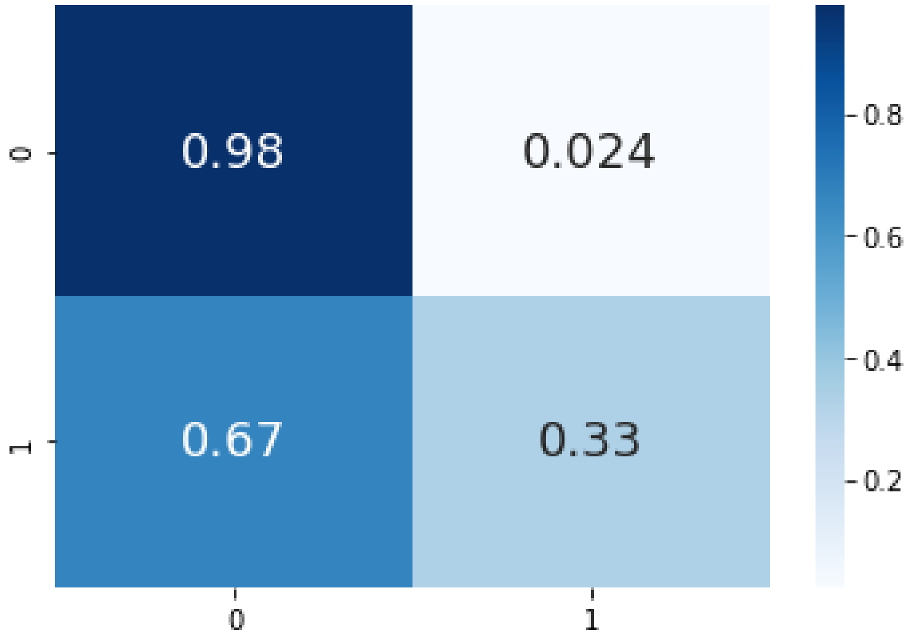
[140]:
```
forest = RandomForestClassifier(random_state = 42)
modelF = forest.fit(X_train, y_train)
y_predF = modelF.predict(X_test)
```

[141]:
```
print('Test Accuracy: %f'%(np.mean(y_test == y_predF) * 100))

print('Recall:', recall_score(y_test, y_predF, average = None))
print('Precision:', precision_score(y_test, y_predF, average = None))
print('Clasification report: \n', classification_report(y_test, y_predF))
print('Confussion matrix:\n',confusion_matrix(y_test, y_predF))
```

```
Test Accuracy: 84.788597
Recall: [0.976 0.331]
Precision: [0.855 0.771]
Clasification report:
               precision    recall  f1-score   support

           0       0.86      0.98      0.91     13388
           1       0.77      0.33      0.46      3310

    accuracy                           0.85     16698
   macro avg       0.81      0.65      0.69     16698
weighted avg       0.84      0.85      0.82     16698

Confussion matrix:
 [[13063   325]
 [ 2215  1095]]
```

[143]:
```
cm = confusion_matrix(y_test, y_predF, normalize = 'true')
sns.heatmap(cm, annot = True, cmap = 'Blues', annot_kws = {'fontsize': 18})
```

[143]: <matplotlib.axes._subplots.AxesSubplot at 0x7f79f01dbb20>



From the confusion matrix, we can see that 66% of y = 1 were predicted to be 0. One possible reason might be the class imbalance in train/test sets. To adress this problem, I decided to oversample the minority class to have 50% of the number of instances of majority class, then use random undersampling to reduce the number of instances in the majority class to have the same number of instances as the minority class.

[144]:
```
y_train.value_counts()
```

[144]:
```
0    53350
1    13438
Name: target, dtype: int64
```

[80]:
```
from imblearn.pipeline import Pipeline
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTE
```

[145]:
```
over = SMOTE(sampling_strategy = 0.5, random_state = 42)
under = RandomUnderSampler(sampling_strategy = 1, random_state = 42)

steps = [('o', over),
         ('u', under)]
pipeline = Pipeline(steps = steps)

X_train, y_train = pipeline.fit_resample(X_train, y_train)
```

```
print('y training set after SMOTE')
y_train.value_counts()
```

```
y training set after SMOTE
```

[145]:
```
0    26675
1    26675
Name: target, dtype: int64
```

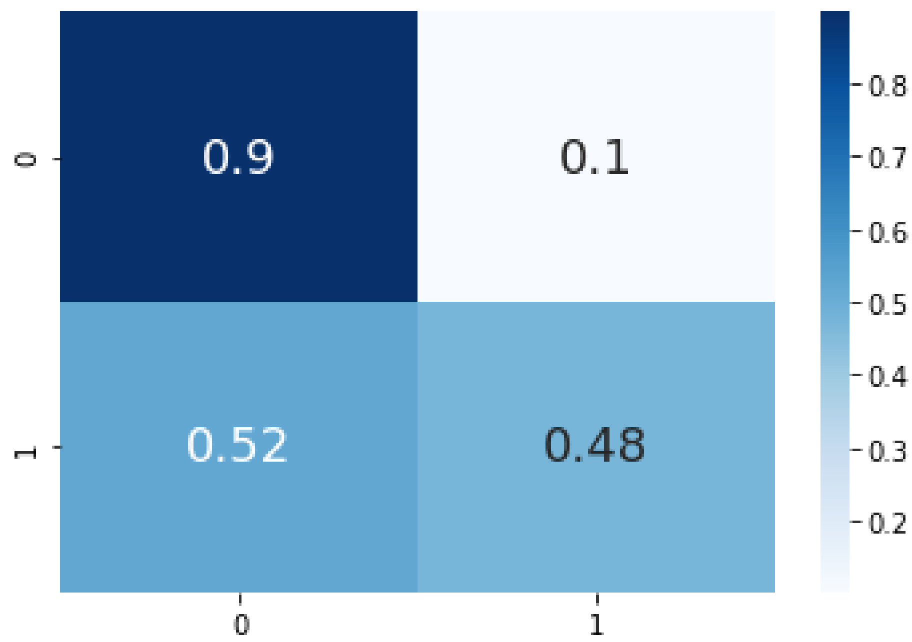
[150]:
```
forest = RandomForestClassifier(random_state = 42)
modelF = forest.fit(X_train, y_train)
y_predF = modelF.predict(X_test)
```

[151]:
```
print('Test Accuracy: %f'%(np.mean(y_test == y_predF) * 100))

print('Recall:', recall_score(y_test, y_predF, average = None))
print('Precision:', precision_score(y_test, y_predF, average = None))
print('Clasification report: \n', classification_report(y_test, y_predF))
print('Confussion matrix:\n',confusion_matrix(y_test, y_predF))
```

```
Test Accuracy: 81.458857
Recall: [0.898 0.478]
Precision: [0.874 0.536]
Clasification report:
               precision    recall  f1-score   support

           0       0.87      0.90      0.89     13388
           1       0.54      0.48      0.51      3310

    accuracy                           0.81     16698
   macro avg       0.71      0.69      0.70     16698
weighted avg       0.81      0.81      0.81     16698

Confussion matrix:
 [[12020  1368]
 [ 1728  1582]]
```

[152]:
```
cm = confusion_matrix(y_test, y_predF, normalize = 'true')
sns.heatmap(cm, annot = True, cmap = 'Blues', annot_kws = {'fontsize': 18})
```

[152]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7a029b3e20>
```



We see a slight improvement in random forest performance, however, the recall for value 1 is still low.

6.0.4 XGB Classifier

[112]:
```
from xgboost import XGBClassifier
```

[153]:
```
xgb = XGBClassifier(random_state = 42)
modelX = xgb.fit(X_train, y_train)
y_predX = modelX.predict(X_test)
```

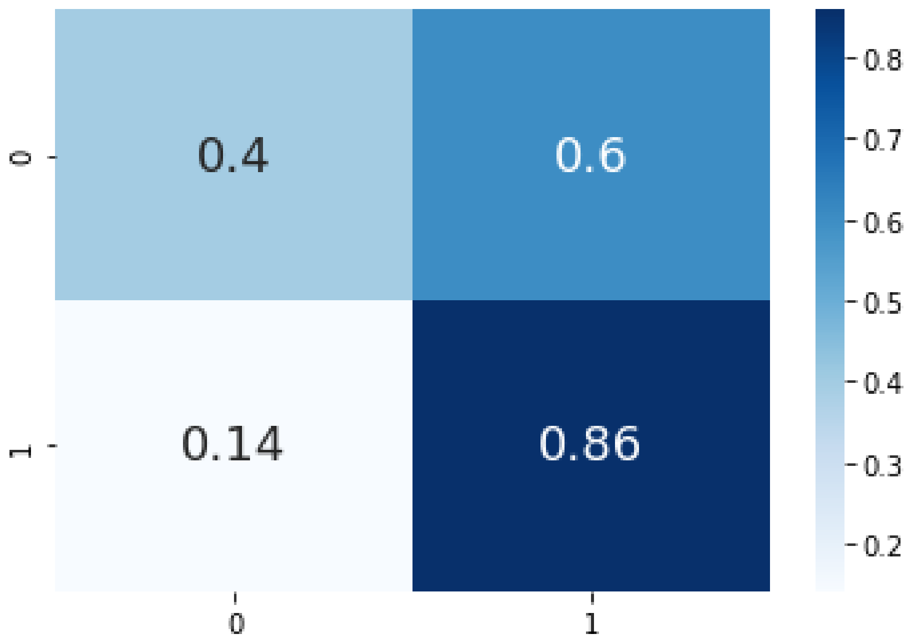
[154]:
```
print('Test Accuracy: %f'%(np.mean(y_test == y_predX) * 100))

print('Recall:', recall_score(y_test, y_predX, average = None))
print('Precision:', precision_score(y_test, y_predX, average = None))
print('Clasification report: \n', classification_report(y_test, y_predX))
print('Confussion matrix:\n',confusion_matrix(y_test, y_predX))
```

```
Test Accuracy: 49.011858
Recall: [0.399 0.858]
Precision: [0.919 0.261]
Clasification report:
                precision    recall  f1-score   support


           0       0.92      0.40      0.56     13388
           1       0.26      0.86      0.40      3310
```

```
    accuracy                           0.49     16698
   macro avg       0.59      0.63      0.48     16698
weighted avg       0.79      0.49      0.53     16698

Confussion matrix:
 [[5343 8045]
 [ 469 2841]]
```

[155]:
```
cm = confusion_matrix(y_test, y_predX, normalize = 'true')
sns.heatmap(cm, annot = True, cmap = 'Blues', annot_kws = {'fontsize': 18})
```

[155]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f7a02d67e50>`



XGBClassifier, on the other hand, shows the opposite picture. It has lower precision and higher recall for value of 1, but the opposite for value 0. Although model require further tuning, XGBClassifier would preferred as we desire to predice houses with potential 'heating' complaints, so it's better to predict y = 1 when in doubt.

6.1 Concluding Remarks - The performance of the predicitve models above is not great. The agency may start with those model but they should do more tuning or use some other datasets which have attributes those are better predictor of the complaint. Overall, there is a potential of building a model to predict possibility of identified complaint.

6.2 Future Development:

```
- Feature Engineering
- Try different techniques for addressing the class imbalance
- Add more classifiers
- Perfrom hyperparameter tuning and find the optimal set of paramters
```