DCRA-McCourt Partnership Insight Report

Lily Yue

Executive Summary

By conducting category-specific text analysis on the `complaint_confidential` column from the `complaint_inspections` dataset, I identified key items or areas to focus on when inspecting different types of property, and showed that the possibility of developing different inspection procedures for different types of property. Among the 5 categories of properties identified - 'Apartment', 'Cooperative Association', 'One Family Rental', 'Two Family Rental', and ' Hotel', it is clear that DCRA should focus on the kitchen and bathroom inspection for apartments, office area inspection for cooperative association, kitchen, floor and, ceiling problems for family rental, and leaking and elevator problems for hotel.

Possible insights extract from analysis are: 1) Endorsement category-specific inspection procedure could be established for furture proactive inspection to improve efficiency based on property type; 2) quicker and more comprehensive complaint-based inspection will be done if further access is given, and thus less future complaints may come from identified high-profile properties; 3)The 40,000+ high-profile properties identified by merging datasets could be turned into a database focusing on repetitive violations or identifying high-rish areas based on address_id, and assign special inspection unit for regular inspection and compliance enforcement.

Current Approach (if applicable)

The inspection process are *type-specific*. With **proactive inspections**, there is a list of frequently cited violations that the inspectors work through, and the inspectors are also trained on how to recognize approximately 300 violation. The current proactive process selects randomly from the pool of eligible buildings that haven't been inspected in the last 4 years. The 1st address is selected randomly across the city, the following addresses are randomly selected from that ward/clusters and assigned to inspectors. With **reactive inspections**, the inspectors usually only evaluate the complaint items but extend throughout the property if access is provided. The reactive inspections are usually first-come-first-serve.

The inspection process are also *time-dependent*. Re-inspections look for whether initially found violations were abated, but will have new inspection CAP if new violations are found.

Proposed Approach

I propose an approach to identify endorsement category-based inspection process. In the cleaned and merged dataset below, there are 5 categories of residential property: 'Apartment', 'One Family Rental', 'Two Family Rental', 'Cooperative Association', and 'Hotel'. By conducting text analysis on the `complaint_confidential` column from the `complaint_inspections` dataset, I identify key category-specific item or area indicated by frequently shown words in the text analysis vocabulary, and identify common items for inspection across different categories and also property-specific items for inspection.

Data Sources

I merged four datasets: `residential_license` , `proactive_inspections` , `violation_data_count_fines_severity` , and `complaint_inspections` .

`residential_license` provides information on cap status, license status, types or organization, unit, and address for each property. `proactive_inspection` provides information on the results and status of proactive inspections on randomly selected properties. `complaint_inspection` provides information on the results and status of complaint inspections on the first-come-first-serve base. `violation_data_count_fines_severity` provides information on violation severity shown by initial violations, 1 day violations, 30 day violations, and totol fines.

My final merged datasets includes the following columns: `CAP_ID` , `ENDORESEMENT_CATEGORY` , `inspection_result_comment` , `complaint_confidential` . The `ENDORESEMENT_CATEGORY` provides the categorization information of the property, and due to data cleaning and merging, we ended up with four categories: 'Apartment', 'Cooperative Association', 'One Family Rental', 'Two Family Rental', and ' Hotel'. The `inspection_result_comment` and `complaint_confidential` columns are both from the `complaint_inspection` dataset. After mering, these observations are properties that have residential licenses, underwent both proactive and complaint inspection process, and fined by DCRA. These characteristics indicate that there is a high possibility of repetitive violations and thus make them high-profile targets of further inspection and regulation compliance enforcement.

Techniques Applied

Natural Language Processing is a set of techniques used to process and analyze texts. In the merged dataset below, I conducted text analysis on the `confidential_inspection` column for each property type to extract useful information on common violation-related items in each residential categories by ranking words frequency from the highest to the lowest.

The two python tools used are **CountVectorizer** and **Tfidftransformer** from sckit-learn package. CountVectorizer parses the text to remove words, a process called *tokenization*, and then encodes the words as intergers or floating points values as input for *feature extraction (or vectorization)* that gives a document term matrix. Tfidftransformer then transforms the matrix to a normalized tf-idf representation, which scales down the impact of frequently occured tokens in the text corpus (such as violation, inspection, etc, in this context) and that are less informative compared to tokens occured in a small fraction of the text corpus.

Findings

'Apartment： the top violation-related items are wire, window, water, wall, tub, trash, toilet, stove, sink, shower, sewer, and refrigerator. Since aparment observations are much more than other categories of property, I only picked words whose frequency is above 650. From this list of items, we can properly infer that inspection should focus on kitchen and bathroom area.

'Cooperative Association': the top violation-related items are water, wall, tub, trash, stove, smoke, sink, pipe, light, leaking, heat, floor, ceiling, and carpet. From this list of items, we can properly infer that inspection should focus on the office area. In fact, the word 'office' is a high-frequency word appeared in the vocabulary.

'One Family Rental': the top violation-related items are water, sink, roof, refrigerator, pipe, kitchen, electric, door, ceiling, carpet. From this list of items, we can properly infer that the inspection should focus on kitchen, ceiling, and floor problems.

'Two Family Rental': the top violation-related items are leaking and ceiling. These items reinforce the issues of ceiling and floor problems shared by family rental.

'Hotel': the top violation-related items are leak, elevator, and ceiling.

We can see that water, sink, ceiling, and leaking problems are common across categories. This finding not only reiterate the importance of close inspections on these issues but also points out the importance of potentially reinforcing regular maintenance on these items to decreaes future complaints.

Discussion

1) The missing values filter out many potential observations for analysis - there are more than 5 categories for ENDORSEMENTS_CATEGORY - and unequal sample size for each property type may affect the analysis. For example, for the 'Hotel' category, there are high-frequency words like *zoning, parking, grease, shelter, fence, hallway* appeared if we don't filter missing values.

2) The data still includes many not-so-useful information such as names and numbers. Even though attampted were made removing numbers (seen below in th implementation section), but the results turned out to be less desirable for identifying item or area problems for each category.

3) The use of address_id rather than CAP_ID filters out illegal addresses that DCRA could pick up during inspection.

4) It is very hard to define stop words and justify restrictions on max_df or min_df when we define CountVectorizer besides eyeballing the usefulness of the bag of words produced by the algorithm.

5) The list of words and text analysis criteria can be highly dependent on the dataset itself. For example, for the two family rental unit data, I was not able to use a comparatively restrictive CounterVectorize function with the same max_df (0.95) and min_df(0.02) as applied in other categories because there is no observation left after pruning. The list of words are also susceptible with regards to merging criteria and restrictions. For example, if I only merge the residential_license and complaint_inspections datasets without the others used in the codes below, the number of observations after data cleaning are very different, and there are more ENDORSEMENT_CATEGORY values left.

Implementation

Load packages and datasets

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransf
ormer
warnings.filterwarnings('ignore')
```

In [78]:
```
residential = pd.read_csv('residential_licenses.csv')
proactive=pd.read_csv('proactive_inspections.csv')
violation=pd.read_csv('violation_data_count_fines_severity.csv')
complaint=pd.read_csv('complaint_inspections.csv')
```

Data wrangling

In [79]:
```
# Select columns from complaint dataset
comments=complaint[['CAP_ID','inspection_result_comment','complaint_confidential','address_id']]
comments['address_id']=comments['address_id'].astype(float);
comments.head()
```

Out[79]:

	CAP_ID	inspection_result_comment	complaint_confidential	address_id
0	18ENF-HOUS-02209	Date of inspection: 09/18/2018 \r\nTime of I...	RUST ON WALLS AND STEPS ON 8TH FLOOR OF BUILDI...	277853.0
1	18ENF-HOUS-02209	Date of Re-inspection: 12/04/2018 \r\nProperty...	RUST ON WALLS AND STEPS ON 8TH FLOOR OF BUILDI...	277853.0
2	19ENF-HOUS-03456	No contact information. Arrived at the propert...	NaN	296110.0
3	19ENF-HOUS-01733	DCRA completed inspections for DCHA potential ...	NaN	310937.0
4	CRM1201303	On 5-23-17 An Re-inspection was conducted on t...	carport has collapsed	43059.0

In [98]:
```
# Transfer data type from integer to float for dataset merge
proactive = proactive.drop(['complaint_confidential', 'inspection_result_comment'], axis=1)
proactive['address_id']=proactive['address_id'].astype(float)
# Check result
proactive.columns
```

Out[98]:
```
Index(['As_Of_Date', 'CAP_CreateDate', 'CAP_ID', 'CAP_STATUS',
       'CAP_STATUS_DATE', 'CAP_ALIAS', 'CAP_CREATED_BY_confidential',
       'G6_Unique_ID', 'R3_BUREAU_CODE', 'Schedule_Date', 'Action_Group',
       'Action_Type', 'Action_Designation', 'Completed_Date', 'Request_DD',
       'G6_STATUS', 'REC_DATE', 'R3_DIVISION_CODE', 'INSP_RESULT_TYPE',
       'inspector_user_id_confidential', 'action_by_confidential',
       'BLDGNO_confidential', 'STNAME', 'STTYPE', 'QUAD', 'CITY', 'STATE',
       'ZIP', 'Address_confidential', 'group', 'Ward', 'GIS_CLUSTER', 'Anc',
       'crm_id', 'inspector_group', 'Inspector_Full_Name_confidential',
       'status_des', 'inspection_type', 'square', 'lot', 'address_id'],
      dtype='object')
```

In [99]:
```
# Rename cap_id column in violation dataset
violation_new=violation.rename(columns={"cap_id": "CAP_ID"})
# Check result
violation_new.head()
```

Out[99]:

	CAP_ID	initial_violations	total_fine	1_day_violations	30_day_violations	other_violations
0	-19UNIT-INS-001854	2	2594	1	1	0
1	18ENF-HOUS-00046	7	4172	2	5	0
2	18ENF-HOUS-00050	8	5189	3	5	0
3	18ENF-HOUS-00051	12	10783	6	6	0
4	18ENF-HOUS-00053	2	1018	1	1	0

In [100]:
```
# Merge the three datasets together based on address_id and CAP_ID
merge1=pd.merge(residential, proactive, on='address_id')
```

In [101]:
```
merge2=pd.merge(merge1, violation_new, on='CAP_ID')
```

In [102]:
```
merge3=pd.merge(merge2, comments, on='address_id')
merge3.head()
```

Out[102]:

	CUST_NUM	TYPE	BILL_ADDR1_confidential	BILL_ADDR2_confidential	BILL_ADDR3_confide
0	65004475	Business License	17009 LONGLEAF DR	NaN	
1	65004475	Business License	17009 LONGLEAF DR	NaN	
2	65004475	Business License	17009 LONGLEAF DR	NaN	
3	65004475	Business License	17009 LONGLEAF DR	NaN	
4	65004475	Business License	17009 LONGLEAF DR	NaN	

5 rows × 89 columns

In [103]:
```
# Rename duplicate x columns
merge3.rename(columns={'CAP_ID_x': 'CAP_ID'}, inplace=True)
# Check result
merge3.head()
```

Out[103]:

	CUST_NUM	TYPE	BILL_ADDR1_confidential	BILL_ADDR2_confidential	BILL_ADDR3_confide
0	65004475	Business License	17009 LONGLEAF DR	NaN	
1	65004475	Business License	17009 LONGLEAF DR	NaN	
2	65004475	Business License	17009 LONGLEAF DR	NaN	
3	65004475	Business License	17009 LONGLEAF DR	NaN	
4	65004475	Business License	17009 LONGLEAF DR	NaN	

5 rows × 89 columns

In [104]:
```
# Select columns to create new dataset
data = merge3[['CAP_ID','address_id','ENDORSEMENTS_CATEGORY','inspection
_result_comment', 'complaint_confidential']]
```

In [105]:
```
# Drop NaN values
data.dropna(how='any',inplace=True)
len(data)
```

Out[105]:
```
40404
```

In [106]:
```
# Get a list of categories of endorsement_category
data['ENDORSEMENTS_CATEGORY'].unique()
```

Out[106]:
```
array(['Apartment', 'Cooperative Association', 'Two Family Rental',
       'One Family Rental', 'Hotel'], dtype=object)
```

In [108]:
```
# Create new datasets based on endorsement_category
one_family_rental = data[data['ENDORSEMENTS_CATEGORY']=='One Family Rent
al']
apartment = data[data['ENDORSEMENTS_CATEGORY']=='Apartment']
two_family_rental = data[data['ENDORSEMENTS_CATEGORY']=='Two Family Rent
al']
cooperative_association = data[data['ENDORSEMENTS_CATEGORY']=='Cooperati
ve Association']
hotel = data[data['ENDORSEMENTS_CATEGORY']=='Hotel']
```

In [109]:
```
# Create lists of complaints for each category
one_family_rental_complaints = one_family_rental['complaint_confidentia
l'].dropna().tolist()
apartment_complaints = apartment['complaint_confidential'].dropna().toli
st()
two_family_rental_complaints = two_family_rental['complaint_confidentia
l'].dropna().tolist()
cooperative_association_complaints = cooperative_association['complaint_
confidential'].dropna().tolist()
hotel_complaints = hotel['complaint_confidential'].dropna().tolist()
```

Modeling Tasks

In [110]:
```
# Show full dataframe
pd.set_option('display.max_rows', None)
```

If one wants to define a list of stop words in the future:

1) Find and create a list of stop words excluding all numbers from the complaint_confidential column
category = data['complaint_confidential']
matrix_category = vectorizer.fit_transform(category)

2) Transform the document-term matrix
matrix_category_2 = TfidfT.fit_transform(matrix_category)

3) Examine the vocabulary
bag_of_words = vectorizer.get_feature_names()

4) Find all numbers in the vocabulary
number = [item for item in bag_of_words if item.isdigit()]

5) Create a vectorizer
vectorizer = CountVectorizer(stop_words=number, min_df=0.005)

In [111]:
```
# Create a vectorizer and a transformer
vectorizer = CountVectorizer(stop_words='english', min_df=0.005)
TfidfT = TfidfTransformer()
```

In [63]:
```
# Conduct text analysis on apartment complaints
# Create a document term matrix
matrix_apartment_complaints = vectorizer.fit_transform(apartment_complai
nts)
# Transform the document-term matrix
matrix_apartment_complaints_2 = TfidfT.fit_transform(matrix_apartment_co
mplaints)
# Examine the vocabulary
apartment_vocab=vectorizer.vocabulary_.items()
pd.DataFrame(list(apartment_vocab), columns=['Word', 'Frequency']).sort_
values(by=['Frequency'], ascending=False)
```