Capstone Project - The Battle of the Neighborhoods (Week 2)

Applied Data Science Capstone by IBM/Coursera

Table of contents

Introduction: Business Problem

The aim of this project is to find a safe and secure location for opening of commercial establishments in Vancouver, Canada. Specifically, this report will be targeted to stakeholders interested in opening any business place like **Grocery Store** in **Vancouver City**, Canada.

The first task would be to **choose the safest borough** by analysing crime data for opening a grocery store and **short listing a neighbourhood**, where grocery stores are not amongst the most commom venues, and yet **as close to the city as possible**.

We will make use of our data science tools to analyse data and focus on the safest borough and explore its neighborhoods and the 10 most common venues in each neighborhood so that the best neighborhood where grocery store is not amongst the most common venue can be selected.

Data

Based on definition of our problem, factors that will influence our decission are:

- finding the safest borough based on crime statistics
- finding the most common venues
- choosing the right neighbourhood within the borough

We will be using the geographical coordinates of Vancouver to plot neighbourhoods in a borough that is safe and in the city's vicinity, and finally cluster our neighborhoods and present our findings.

Following data sources will be needed to extract/generate the required information:

- **Part 1**: Using a real world data set from Kaggle containing the Vancouver Crimes from 2003 to 2019: A dataset consisting of the crime statistics of each Neighbourhoof in Vancouver along with type of crime, recorded year, month and hour.
- **Part 2**: Gathering additional information of the list of officially categorized boroughs in Vancouver from Wikipedia.: Borough information will be used to map the existing data where each neighbourhood can be assigned with the right borough.
- **Part 3**: Creating a new consolidated dataset of the Neighborhoods, along with their boroughs, crime data and the respective Neighbourhood's co-ordinates.: This data will be fetched using OpenCage Geocoder to find the safest borough and explore the neighbourhood by plotting it on maps using Folium and perform exploratory data analysis.
- **Part 4**: Creating a new consolidated dataset of the Neighborhoods, boroughs, and the most common venues and the respective Neighbourhood along with co-ordinates.: This data will be fetched using Four Square API to explore the neighbourhood venues and to apply machine learning algorithm to cluster the neighbourhoods and present the findings by plotting it on maps using Folium.

Part 1: Using a real world data set from Kaggle containing the Vancouver Crimes from 2003 to 2019

Vancouver Crime Report

Properties of the Crime Report

- TYPE - Crime type
- YEAR - Recorded year
- MONTH - Recorded month
- DAY - Recorded day
- HOUR - Recorded hour
- MINUTE - Recorded minute
- HUNDRED_BLOCK - Recorded block
- NEIGHBOURHOOD - Recorded neighborhood
- X - GPS longtitude
- Y - GPS latitude

Data set URL: https://www.kaggle.com/agilesifaka/vancouver-crime-report/version/2

Importing all the necessary Libraries

In [3]:
```
import numpy as np
import pandas as pd

#Command to install OpenCage Geocoder for fetching Lat and Lng of Neigh
borhood
!pip install opencage
!pip install folium

#Importing OpenCage Geocoder
from opencage.geocoder import OpenCageGeocode

# use the inline backend to generate the plots within the browser
%matplotlib inline
```

```
#Importing Matplot lib and associated packages to perform Data Visualis
ation and Exploratory Data Analysis
import matplotlib as mpl
import matplotlib.pyplot as plt

mpl.style.use('ggplot') # optional: for ggplot-like style

# check for latest version of Matplotlib
print ('Matplotlib version: ', mpl.__version__) # >= 2.0.0

# Matplotlib and associated plotting modules
import matplotlib.cm as cm
import matplotlib.colors as colors

#Importing folium to visualise Maps and plot based on Lat and Lng
import folium

#Requests to request web pages by making get requests to FourSquare RES
T Client
import requests

#To normalise data returned by FourSquare API
from pandas.io.json import json_normalize

#Importing KMeans from SciKit library to Classify neighborhoods into cl
usters
from sklearn.cluster import KMeans

print('Libraries imported')
```

```
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstora
ge/dhcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is de
precated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstora
ge/util.py:25: CryptographyDeprecationWarning: int_from_bytes is deprec
ated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
Requirement already satisfied: opencage in /opt/conda/envs/Python-3.7-m
```

```
ain/lib/python3.7/site-packages (1.2.2)
Requirement already satisfied: backoff>=1.10.0 in /opt/conda/envs/Pytho
n-3.7-main/lib/python3.7/site-packages (from opencage) (1.10.0)

Requirement already satisfied: Requests>=2.2.0 in /opt/conda/envs/Pytho
n-3.7-main/lib/python3.7/site-packages (from opencage) (2.24.0)
Requirement already satisfied: pyopenssl>=0.15.1 in /opt/conda/envs/Pyt
hon-3.7-main/lib/python3.7/site-packages (from opencage) (19.1.0)
Requirement already satisfied: six>=1.4.0 in /opt/conda/envs/Python-3.7
-main/lib/python3.7/site-packages (from opencage) (1.15.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Py
thon-3.7-main/lib/python3.7/site-packages (from Requests>=2.2.0->openca
ge) (2020.12.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1
in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from Re
quests>=2.2.0->opencage) (1.25.9)
Requirement already satisfied: chardet<4,>=3.0.2 in /opt/conda/envs/Pyt
hon-3.7-main/lib/python3.7/site-packages (from Requests>=2.2.0->opencag
e) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/envs/Python-
3.7-main/lib/python3.7/site-packages (from Requests>=2.2.0->opencage)
(2.9)
Requirement already satisfied: cryptography>=2.8 in /opt/conda/envs/Pyt
hon-3.7-main/lib/python3.7/site-packages (from pyopenssl>=0.15.1->openc
age) (3.4.7)
Requirement already satisfied: cffi>=1.12 in /opt/conda/envs/Python-3.7
-main/lib/python3.7/site-packages (from cryptography>=2.8->pyopenssl>=
0.15.1->opencage) (1.14.0)
Requirement already satisfied: pycparser in /opt/conda/envs/Python-3.7-
main/lib/python3.7/site-packages (from cffi>=1.12->cryptography>=2.8->p
yopenssl>=0.15.1->opencage) (2.20)
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstora
ge/dhcrypto.py:16: CryptographyDeprecationWarning: int_from_bytes is de
precated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
/opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages/secretstora
ge/util.py:25: CryptographyDeprecationWarning: int_from_bytes is deprec
ated, use int.from_bytes instead
  from cryptography.utils import int_from_bytes
Collecting folium
  Downloading folium-0.12.1-py2.py3-none-any.whl (94 kB)
```

```
  Downloading folium-0.12.1-py2.py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 6.1 MB/s  eta 0:00:01
Requirement already satisfied: requests in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from folium) (2.24.0)
Requirement already satisfied: jinja2>=2.9 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from folium) (2.11.2)
Collecting branca>=0.3.0
  Downloading branca-0.4.2-py3-none-any.whl (24 kB)
Requirement already satisfied: numpy in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from folium) (1.18.5)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->folium) (1.25.9)
Requirement already satisfied: chardet<4,>=3.0.2 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->folium) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->folium) (2.9)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from requests->folium) (2020.12.5)
Requirement already satisfied: MarkupSafe>=0.23 in /opt/conda/envs/Python-3.7-main/lib/python3.7/site-packages (from jinja2>=2.9->folium) (1.1.1)
Installing collected packages: branca, folium
Successfully installed branca-0.4.2 folium-0.12.1
Matplotlib version:  3.2.2
Libraries imported
```

Reading from the Dataset

Due to sheer amount of data(~ 600,000 rows), it was not possible to process all of them and instead for this project we will be considering the recent crime report of the 2018.

In [4]:
```python
vnc_crime_df = pd.read_csv('https://raw.githubusercontent.com/RamanujaSVL/Coursera_Capstone/master/vancouver_crime_records_2018.csv', index_col=None)
```

```
#Dropping X,Y which represents Lat, Lng data as Coordinates, the data s
eems to be corrupt
vnc_crime_df.drop(['Unnamed: 0','MINUTE', 'HUNDRED_BLOCK', 'X', 'Y'], a
xis = 1, inplace = True)

#vnc_crime_df.columns

vnc_crime_df.head()
```

Out[4]:

	TYPE	YEAR	MONTH	DAY	HOUR	NEIGHBOURHOOD
0	Break and Enter Commercial	2018	3	2	6	West End
1	Break and Enter Commercial	2018	6	16	18	West End
2	Break and Enter Commercial	2018	12	12	0	West End
3	Break and Enter Commercial	2018	4	9	6	Central Business District
4	Break and Enter Commercial	2018	10	2	18	Central Business District

Changing the name of columns to lowercase

In [5]:

```
vnc_crime_df.columns = ['Type', 'Year','Month','Day','Hour','Neighbourh
ood']
vnc_crime_df.head()
```

Out[5]:

	Type	Year	Month	Day	Hour	Neighbourhood
0	Break and Enter Commercial	2018	3	2	6	West End
1	Break and Enter Commercial	2018	6	16	18	West End
2	Break and Enter Commercial	2018	12	12	0	West End
3	Break and Enter Commercial	2018	4	9	6	Central Business District
4	Break and Enter Commercial	2018	10	2	18	Central Business District

Total Crimes in different Neighborhoods

In [6]:
```
vnc_crime_df['Neighbourhood'].value_counts()
```

Out[6]:
```
Central Business District    10857
West End                      3031
Mount Pleasant                2396
Strathcona                    1987
Kitsilano                     1802
Fairview                      1795
Renfrew-Collingwood           1762
Grandview-Woodland            1761
Kensington-Cedar Cottage      1391
Hastings-Sunrise              1270
Sunset                         967
Riley Park                     866
Marpole                        828
Victoria-Fraserview            600
Killarney                      565
Oakridge                       499
Dunbar-Southlands              474
Kerrisdale                     417
Shaughnessy                    414
West Point Grey                372
Arbutus Ridge                  311
South Cambie                   292
Stanley Park                   154
Musqueam                        17
Name: Neighbourhood, dtype: int64
```

Part 2: Gathering additional information about the Neighborhood from Wikipedia

As part of data set Borough which the neighborhood was part of was not categorized, so we will create a dictionary of Neighborhood and based on data in the following [Wikipedia page](#).

In [7]:

```
# define the dataframe columns
column_names = ['Neighbourhood', 'Borough']

# instantiate the dataframe
vnc_neigh_bor = pd.DataFrame(columns=column_names)

vnc_neigh_bor['Neighbourhood'] = vnc_crime_df['Neighbourhood'].unique()

neigh_bor_dict = {'Central Business District':'Central', 'West End':'Ce
ntral', 'Stanley Park':'Central', 'Victoria-Fraserview':'South Vancouve
r',
                  'Killarney':'South Vancouver', 'Musqueam':'South Vanc
ouver', 'Mount Pleasant':'East Side', 'Strathcona':'East Side',
                  'Renfrew-Collingwood':'East Side', 'Grandview-Woodlan
d':'East Side', 'Kensington-Cedar Cottage':'East Side', 'Hastings-Sunri
se':'East Side',
                  'Sunset':'East Side', 'Riley Park':'East Side', 'Kits
ilano':'West Side', 'Fairview':'West Side',
                  'Marpole':'West Side', 'Oakridge':'West Side', 'Dunba
r-Southlands':'West Side', 'Kerrisdale':'West Side',
                  'Shaughnessy':'West Side', 'West Point Grey':'West Si
de', 'Arbutus Ridge':'West Side', 'South Cambie':'West Side'}

for row, neigh in zip(neigh_bor_dict, vnc_neigh_bor['Neighbourhood']):
  vnc_neigh_bor.loc[vnc_neigh_bor.Neighbourhood == row, 'Borough'] = ne
igh_bor_dict.get(row)

vnc_neigh_bor.dropna(inplace=True)

print("Total Neighbourhood Count",len(vnc_neigh_bor['Neighbourhood']),
"Borough Count",len(vnc_neigh_bor['Borough'].unique()))

vnc_neigh_bor.head()
```

```
Total Neighbourhood Count 24 Borough Count 4
```

Out[7]:

	Neighbourhood	Borough
0	West End	Central
1	Central Business District	Central

	Neighbourhood	Borough
1	Central Business District	Central
2	Hastings-Sunrise	East Side
3	Grandview-Woodland	East Side
4	Mount Pleasant	East Side

Merging the Crime data Table to include Boroughs

In [8]:

```
vnc_boroughs_crime = pd.merge(vnc_crime_df,vnc_neigh_bor, on='Neighbourhood')

vnc_boroughs_crime.head()
```

Out[8]:

	Type	Year	Month	Day	Hour	Neighbourhood	Borough
0	Break and Enter Commercial	2018	3	2	6	West End	Central
1	Break and Enter Commercial	2018	6	16	18	West End	Central
2	Break and Enter Commercial	2018	12	12	0	West End	Central
3	Break and Enter Commercial	2018	3	2	3	West End	Central
4	Break and Enter Commercial	2018	3	17	11	West End	Central

Further Cleaning the data by dropping rows with invalid data

In [9]:

```
vnc_boroughs_crime.dropna(inplace=True)
vnc_boroughs_crime['Borough'].value_counts()
```

Out[9]:

```
Central            14042
East Side          12400
West Side           7204
South Vancouver     1182
Name: Borough, dtype: int64
```

Methodology

Categorized the methodologysection into two parts:

- **Exploratory Data Analysis:** Visualise the crime repots in different Vancouver boroughs to idenity the safest borough and normalise the neighborhoods of that borough. We will Use the resulting data and find 10 most common venues in each neighborhood.

- **Modelling:** To help stakeholders choose the right neighborhood within a borough we will be clustering similar neighborhoods using K - means clustering which is a form of unsupervised machine learning algorithm that clusters data based on predefined cluster size. We will use K-Means clustering to address this problem so as to group data based on existing venues which will help in the decision making process.

Exploratory Data Analysis

Pivoting the table to better understand the data by crimes per borough

In [13]:

```
vnc_crime_cat = pd.pivot_table(vnc_boroughs_crime,
                               values=['Year'],
                               index=['Borough'],
                               columns=['Type'],
                               aggfunc=len,
                               fill_value=0,
                               margins=True)
vnc_crime_cat
```

Out[13]:

Year

Type	Break and Enter Commercial	Break and Enter Residential/Other	Mischief	Other Theft	Theft from Vehicle	Theft of Bicycle	Theft of Vehicle	Vehicle Collision or Pedestrian Struck (with Fatality)
Borough								
Central	787	198	2280	2489	6871	857	245	1
East Side	786	1043	2192	1674	4754	678	605	8
South Vancouver	49	156	187	88	483	36	71	1
West Side	403	1000	1062	696	2838	588	225	3
All	2025	2397	5721	4947	14946	2159	1146	13

Merging the Pivoted Column with other columns

In [14]:

```
vnc_crime_cat.reset_index(inplace = True)
vnc_crime_cat.columns = vnc_crime_cat.columns.map(''.join)
vnc_crime_cat.rename(columns={'YearAll':'Total'}, inplace=True)
# To ignore bottom All in Borough
vnc_crime_cat = vnc_crime_cat.head(4)
vnc_crime_cat
```

Out[14]:

	Borough	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle	YearT Vel
0	Central	787	198	2280	2489	6871	857	
1	East Side	786	1043	2192	1674	4754	678	

	Borough	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle	YearT Veh
2	South Vancouver	49	156	187	88	483	36	
3	West Side	403	1000	1062	696	2838	588	

Pivoting the table to better understand the data by crimes per neighborhood

In [15]:

```
vnc_crime_neigh = pd.pivot_table(vnc_boroughs_crime,
                                 values=['Year'],
                                 index=['Neighbourhood'],
                                 columns=['Type'],
                                 aggfunc=len,
                                 fill_value=0,
                                 margins=True)
vnc_crime_neigh
```

Out[15]:

	Year							
Type	**Break and Enter Commercial**	**Break and Enter Residential/Other**	**Mischief**	**Other Theft**	**Theft from Vehicle**	**Theft of Bicycle**	**Theft of Vehicle**	**Vehicle Collisio or Pedest Struck (with Fatality**
Neighbourhood								
Arbutus Ridge	12	78	49	18	111	12	12	
Central Business District	551	124	1812	2034	5301	640	165	

	Year							
Type	Break and Enter Commercial	Break and Enter Residential/Other	Mischief	Other Theft	Theft from Vehicle	Theft of Bicycle	Theft of Vehicle	Vehicle Collisio or Pedest Struck (with Fatality
Neighbourhood								
Dunbar-Southlands	8	106	81	31	199	16	9	
Fairview	138	73	233	297	692	245	55	
Grandview-Woodland	148	162	304	215	634	110	123	
Hastings-Sunrise	48	117	195	107	607	52	74	
Kensington-Cedar Cottage	62	145	255	148	541	69	71	
Kerrisdale	24	97	49	9	172	13	11	
Killarney	34	72	90	31	240	19	33	
Kitsilano	106	165	320	154	755	189	51	
Marpole	44	125	134	75	290	34	39	
Mount Pleasant	205	124	353	493	822	232	67	
Musqueam	0	4	3	0	4	2	2	
Oakridge	19	123	64	63	164	18	18	
Renfrew-Collingwood	91	156	243	472	569	37	92	
Riley Park	35	122	140	53	378	52	39	
Shaughnessy	12	120	41	0	187	10	11	
South Cambie	22	42	41	38	111	19	8	
Stanley Park	6	2	8	0	109	14	3	

Year	Break and Enter Commercial	Break and Enter Residential/Other	Mischief	Other Theft	Theft from Vehicle	Theft of Bicycle	Theft of Vehicle	Vehicle Collisi or Pedest Struck (with Fatality
Type								
Neighbourhood								
Strathcona	160	124	527	81	821	108	76	
Sunset	37	93	175	105	382	18	63	
Victoria-Fraserview	15	80	94	57	239	15	36	
West End	230	72	460	455	1461	203	77	
West Point Grey	18	71	50	11	157	32	11	
All	2025	2397	5721	4947	14946	2159	1146	

Merging the Pivoted Column with other columns

In [20]:

```
vnc_crime_neigh.reset_index(inplace = True)
vnc_crime_neigh.columns = vnc_crime_neigh.columns.map(''.join)
vnc_crime_neigh.rename(columns={'YearAll':'Total'}, inplace=True)

vnc_crime_neigh.head()
```

Out[20]:

	Neighbourhood	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle
0	Arbutus Ridge	12	78	49	18	111	12

	Neighbourhood	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle
1	Central Business District	551	124	1812	2034	5301	640
2	Dunbar-Southlands	8	106	81	31	199	16
3	Fairview	138	73	233	297	692	245
4	Grandview-Woodland	148	162	304	215	634	110

Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.

In [21]:
```
vnc_crime_cat.describe()
```

Out[21]:

	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle	Year of Ve
count	4.000000	4.000000	4.00000	4.000000	4.000000	4.000000	4.00
mean	506.250000	599.250000	1430.25000	1236.750000	3736.500000	539.750000	286.50
std	354.409721	488.189427	997.26572	1060.087221	2723.536977	353.955153	226.11
min	49.000000	156.000000	187.00000	88.000000	483.000000	36.000000	71.00
25%	314.500000	187.500000	843.25000	544.000000	2249.250000	450.000000	186.50
50%	594.500000	599.000000	1627.00000	1185.000000	3796.000000	633.000000	235.00
75%	786.250000	1010.750000	2214.00000	1877.750000	5283.250000	722.750000	335.00

	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle	Year' of Ve
max	787.000000	1043.000000	2280.00000	2489.000000	6871.000000	857.000000	605.00

Expolring the data by Visualising

Sorting the data by crimes per neighborhood

In [22]:

```
vnc_crime_neigh.sort_values(['Total'], ascending = False, axis = 0, inp
lace = True )

crime_neigh_top5 = vnc_crime_neigh.iloc[1:6]
crime_neigh_top5
```

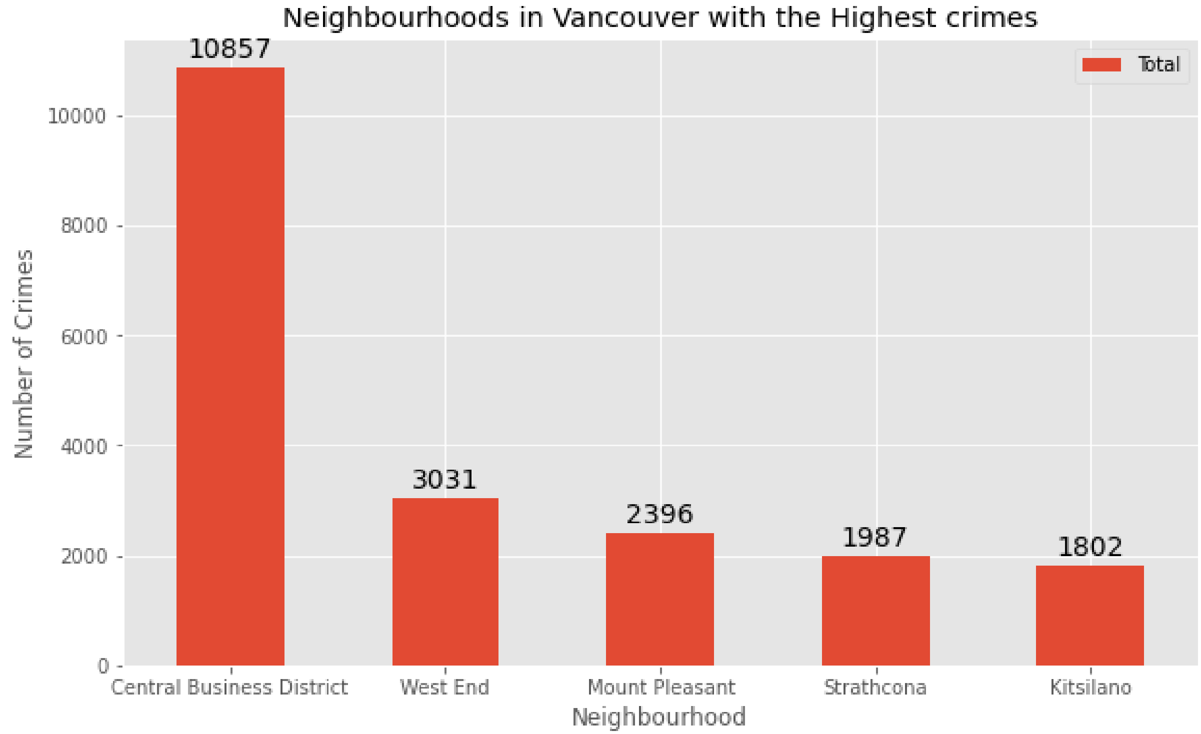
Out[22]:

	Neighbourhood	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle
1	Central Business District	551	124	1812	2034	5301	640
22	West End	230	72	460	455	1461	203
11	Mount Pleasant	205	124	353	493	822	232
19	Strathcona	160	124	527	81	821	108
9	Kitsilano	106	165	320	154	755	189

Five Neighborhoods with highest crime

In [23]:

```
per_neigh = crime_neigh_top5[['Neighbourhood','Total']]

per_neigh.set_index('Neighbourhood',inplace = True)

ax = per_neigh.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('Neighbourhood')
ax.set_title('Neighbourhoods in Vancouver with the Highest crimes')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2),
                (p.get_x()+p.get_width()/2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points',
                fontsize = 14,
               )

plt.show()
```



Five Neighborhoods with lowest crime

In [24]:
```
crime_neigh_low = vnc_crime_neigh.tail(5)
crime_neigh_low
```

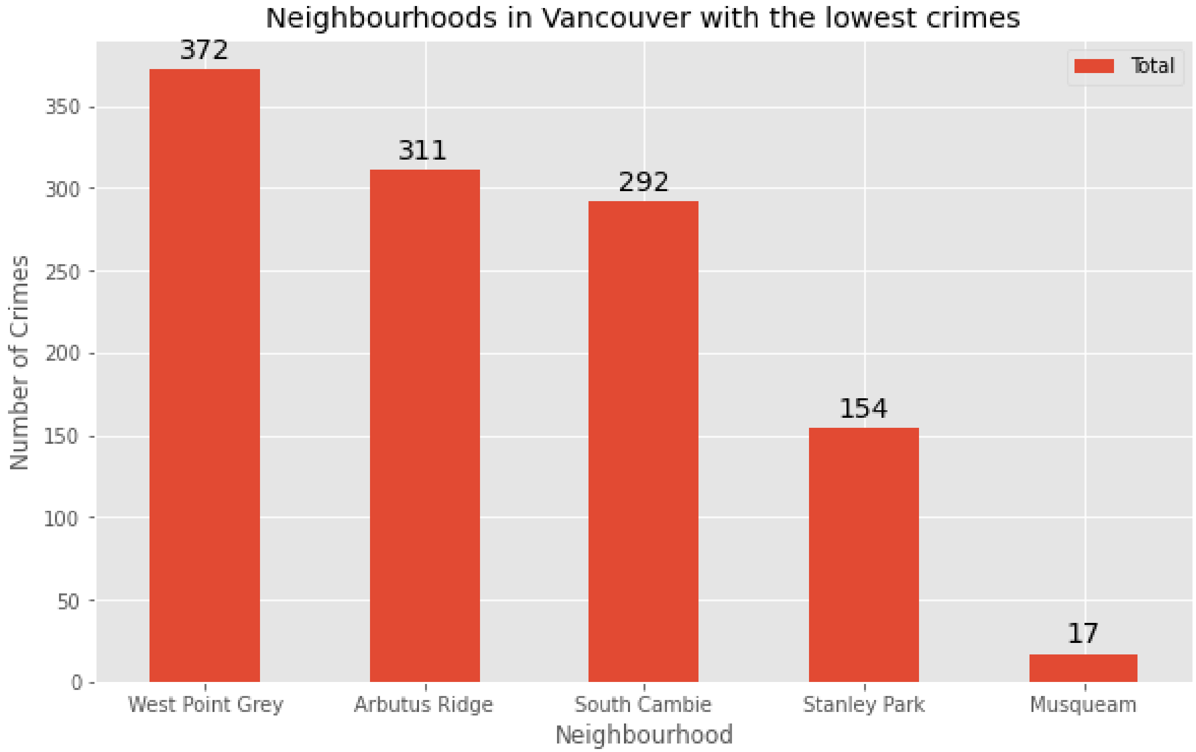
Out[24]:

	Neighbourhood	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle
23	West Point Grey	18	71	50	11	157	32
0	Arbutus Ridge	12	78	49	18	111	12

	Neighbourhood	YearBreak and Enter Commercial	YearBreak and Enter Residential/Other	YearMischief	YearOther Theft	YearTheft from Vehicle	YearTheft of Bicycle
17	South Cambie	22	42	41	38	111	19
18	Stanley Park	6	2	8	0	109	14
12	Musqueam	0	4	3	0	4	2

In [25]:

```
per_neigh = crime_neigh_low[['Neighbourhood','Total']]

per_neigh.set_index('Neighbourhood',inplace = True)

ax = per_neigh.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('Neighbourhood')
ax.set_title('Neighbourhoods in Vancouver with the lowest crimes')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2),
                (p.get_x()+p.get_width()/2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points',
                fontsize = 14,
               )

plt.show()
```



Borough is Vancouver with Highest Crime

In [26]:
```
vnc_crime_cat = pd.pivot_table(vnc_boroughs_crime,
                               values=['Year'],
                               index=['Borough'],
                               columns=['Type'],
                               aggfunc=len,
                               fill_value=0,
                               margins=True)
vnc_crime_cat
```

Out[26]:

Year

Type	Break and Enter Commercial	Break and Enter Residential/Other	Mischief	Other Theft	Theft from Vehicle	Theft of Bicycle	Theft of Vehicle	Vehicle Collision or Pedestrian Struck (with Fatality)
Borough								
Central	787	198	2280	2489	6871	857	245	1
East Side	786	1043	2192	1674	4754	678	605	8
South Vancouver	49	156	187	88	483	36	71	1
West Side	403	1000	1062	696	2838	588	225	3
All	2025	2397	5721	4947	14946	2159	1146	13

In [27]:
```
vnc_crime_cat.reset_index(inplace = True)
vnc_crime_cat.columns = vnc_crime_cat.columns.map(''.join)
vnc_crime_cat.rename(columns={'YearAll':'Total',
                              'YearBreak and Enter Commercial' : 'Break
and Enter Commercial',
                              'YearBreak and Enter Residential/Other' :
'Break and Enter Residential',
                              'YearMischief' : 'Mischief',
                              'YearOther Theft' : 'Other',
                              'YearTheft from Vehicle' : 'Theft from Ve
hicle',
                              'YearTheft of Bicycle' : 'Theft of Bicycl
e',
                              'YearTheft of Vehicle' : 'Theft of Vehicl
e',
                              'YearVehicle Collision or Pedestrian Stru
ck (with Fatality)' : 'Vehicle Collision or Pedestrian Struck (with Fat
ality)',
                              'YearVehicle Collision or Pedestrian Stru
ck (with Injury)' : 'Vehicle Collision or Pedestrian Struck (with Injur
y)'}, inplace=True)
```

```
# To ignore bottom All in Borough
vnc_crime_cat = vnc_crime_cat.head(4)
vnc_crime_cat
```

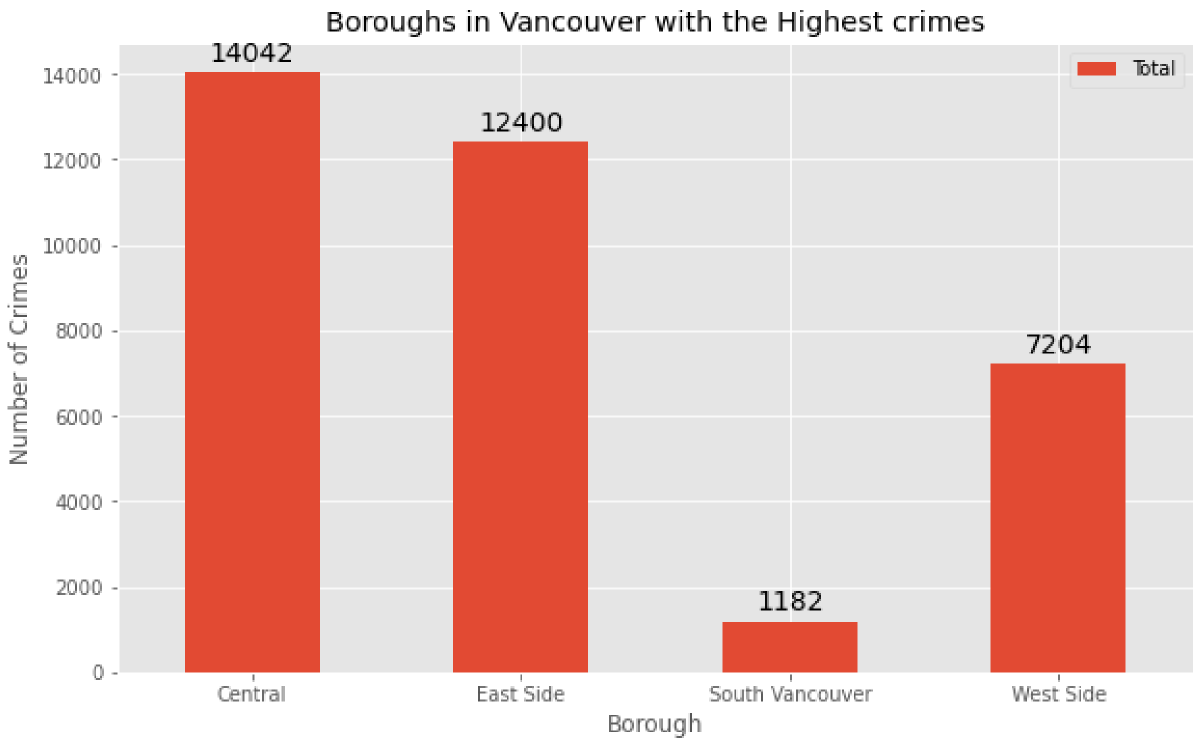
Out[27]:

	Borough	Break and Enter Commercial	Break and Enter Residential	Mischief	Other	Theft from Vehicle	Theft of Bicycle	Theft of Vehicle	Vehicle Collision or Pedestrian Struck (with Fatality)	(Pe
0	Central	787	198	2280	2489	6871	857	245	1	
1	East Side	786	1043	2192	1674	4754	678	605	8	
2	South Vancouver	49	156	187	88	483	36	71	1	
3	West Side	403	1000	1062	696	2838	588	225	3	

In [28]:

```
per_borough = vnc_crime_cat[['Borough','Total']]

per_borough.set_index('Borough',inplace = True)

ax = per_borough.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('Borough')
ax.set_title('Boroughs in Vancouver with the Highest crimes')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2),
                (p.get_x()+p.get_width()/2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points',
                fontsize = 14,
               )
```

```
plt.show()
```



Based on exploratory data analysis it is clear that South Vancouver has the lowest crimes

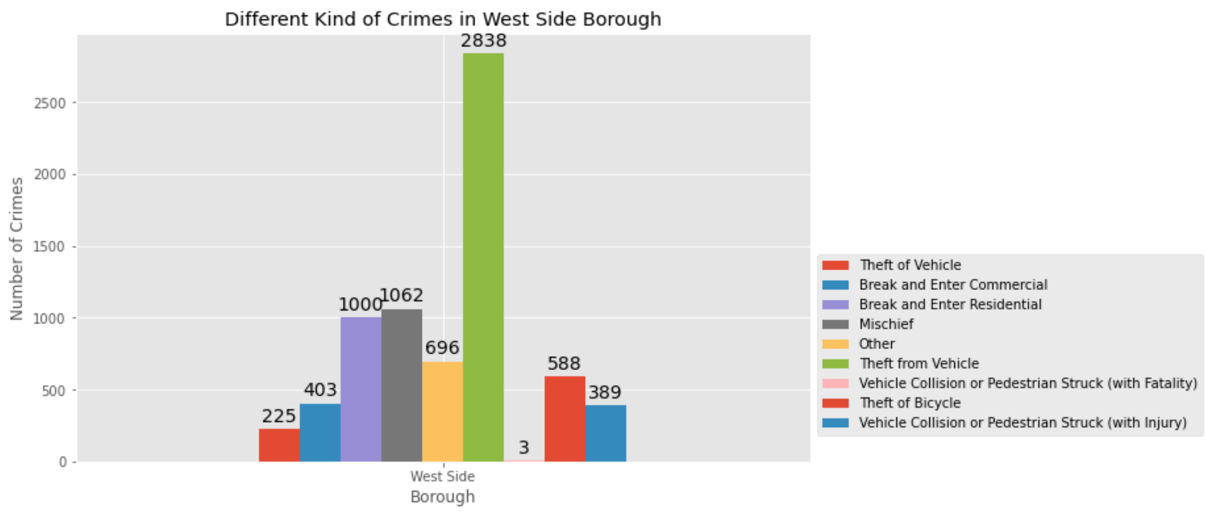
Since South Vancouver has very little number of neighborhoods and opening a commercial establishment would not be viable, we can choose the next borough with lowest crime which is West Side.

Different types of crimes recorded in the West Side Borough

West side was chosen because crime type Break and enter Commercial is also low amongst other crimes types which makes West Side ideal destination for opening of

commercial establishments

In [29]:
```
 vnc_ws_df = vnc_crime_cat[vnc_crime_cat['Borough'] == 'West Side']

 vnc_ws_df = vnc_ws_df.sort_values(['Total'], ascending = True, axis = 
0)

vnc_ws = vnc_ws_df[['Borough','Theft of Vehicle', 'Break and Enter Comm
ercial','Break and Enter Residential','Mischief','Other',
                 'Theft from Vehicle','Vehicle Collision or Pedestrian
 Struck (with Fatality)','Theft of Bicycle',
                 'Vehicle Collision or Pedestrian Struck (with Injury)'
]]


vnc_ws.set_index('Borough',inplace = True)

ax = vnc_ws.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('Borough')
ax.set_title('Different Kind of Crimes in West Side Borough')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=3),
                (p.get_x()+p.get_width()/3., p.get_height()),
                ha='center',
                va='center',
                xytext=(5, 10),
                textcoords='offset points',
                fontsize = 14
               )
    ax.legend(loc='upper left', bbox_to_anchor=(1.00, 0.5))

plt.show()
```



Part 3: Creating a new consolidated dataset of the Neighborhoods, along with their boroughs, crime data and the respective Neighbourhood's co-ordinates.:

This data will be fetched using OpenCage Geocoder to find the safest borough and explore the neighbourhood by plotting it on maps using Folium and perform exploratory data analysis.

Restricting the rows in the data frame to only those with West side as Borough

In [30]:

```
vnc_ws_neigh = vnc_boroughs_crime

#vnc_ws_neigh.drop(['Type','Year', 'Month', 'Day', 'Hour'], axis = 1, i
nplace = True)
vnc_ws_neigh = vnc_ws_neigh[vnc_ws_neigh['Borough'] == 'West Side']
vnc_ws_neigh.reset_index(inplace=True, drop=True)

print('Number of Neighbourhoods in West Side Borough', len(vnc_ws_neigh
['Neighbourhood'].unique()))
```

```
vnc_ws_neigh['Neighbourhood'].unique()
```

```
Number of Neighbourhoods in West Side Borough 10
```

Out[30]:
```
array(['Shaughnessy', 'Fairview', 'Oakridge', 'Marpole', 'Kitsilano',
       'Kerrisdale', 'West Point Grey', 'Arbutus Ridge', 'South Cambi
e',
       'Dunbar-Southlands'], dtype=object)
```

Creating a new Data frame with Lat, Lng being fetched from OpenCage geocoder

In [31]:
```
Latitude = []
Longitude = []
Borough = []
Neighbourhood = vnc_ws_neigh['Neighbourhood'].unique()



key = '830323b5ca694362904814ff0a11b803'
geocoder = OpenCageGeocode(key)

for i in range(len(Neighbourhood)):
    address = '{}, Vancouver, BC, Canada'.format(Neighbourhood[i])
    location = geocoder.geocode(address)
    Latitude.append(location[0]['geometry']['lat'])
    Longitude.append(location[0]['geometry']['lng'])
    Borough.append('West Side')
print(Latitude, Longitude)

#print('The geograpical coordinate of Vancouver City are {}, {}.'.forma
t(latitude, longitude))
```

```
[49.2463051, 49.2619557, 49.2266149, 49.2092233, 49.2694099, 49.220984
8, 49.2681022, 49.2463051, 49.2464639, 49.237864] [-123.1384051, -123.1
304084, -123.1229433, -123.1361495, -123.155267, -123.1595484, -123.202
6425, -123.159636, -123.1216027, -123.1843544]
```

Glimpse of the new Data Frame with Neighborhoods in West Side Borough of Vancoouver along with centroid of their co-ordinates

In [32]:
```
ws_neig_dict = {'Neighbourhood': Neighbourhood,'Borough':Borough,'Latit
ude': Latitude,'Longitude':Longitude}
ws_neig_geo = pd.DataFrame(data=ws_neig_dict, columns=['Neighbourhood',
'Borough', 'Latitude', 'Longitude'], index=None)

ws_neig_geo
```

Out[32]:

	Neighbourhood	Borough	Latitude	Longitude
0	Shaughnessy	West Side	49.246305	-123.138405
1	Fairview	West Side	49.261956	-123.130408
2	Oakridge	West Side	49.226615	-123.122943
3	Marpole	West Side	49.209223	-123.136150
4	Kitsilano	West Side	49.269410	-123.155267
5	Kerrisdale	West Side	49.220985	-123.159548
6	West Point Grey	West Side	49.268102	-123.202642
7	Arbutus Ridge	West Side	49.246305	-123.159636
8	South Cambie	West Side	49.246464	-123.121603
9	Dunbar-Southlands	West Side	49.237864	-123.184354

Fetching the Geographical co-ordiantes of Vancouver to plot on Map

In [33]:
```
address = 'Vancouver, BC, Canada'

location = geocoder.geocode(address)
latitude = location[0]['geometry']['lat']
longitude = location[0]['geometry']['lng']
```

```
print('The geographical coordinate of Vancouver, Canada are {}, {}.'.format(latitude, longitude))
```

```
The geograpical coordinate of Vancouver, Canada are 49.2608724, -123.1139529.
```

Using Folium to plot Vancouver City's West Side Borough and it's Neighborhoods

In [34]:

```
van_map = folium.Map(location=[latitude, longitude], zoom_start=12)

# add markers to map
for lat, lng, borough, neighborhood in zip(ws_neig_geo['Latitude'], ws_neig_geo['Longitude'], ws_neig_geo['Borough'], ws_neig_geo['Neighbourhood']):
    label = '{}, {}'.format(neighborhood, borough)
    label = folium.Popup(label, parse_html=True)
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        popup=label,
        color='red',
        fill=True,
        fill_color='#3186cc',
        fill_opacity=0.7,
        parse_html=False).add_to(van_map)

van_map
```

Out[34]:

Part 4: Creating a new consolidated dataset of the Neighborhoods, boroughs, and the most common venues and the respective Neighbourhood along with co-ordinates.:

This data will be fetched using Four Square API to explore the neighbourhood venues and to apply machine learning algorithm to cluster the neighbourhoods and present the findings by plotting it on maps using Folium.

Setting Up Foursquare Credentials

In [56]:

```
#Four Square Credentials

CLIENT_ID = 'blabla'
CLIENT_SECRET = 'blabla'
VERSION = '20191101'
LIMIT = 100

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET:' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: blabla
CLIENT_SECRET:blabla
```

Defining a function to fetch top 10 venues around a given neighborhood

In [36]:
```
def getNearbyVenues(names, latitudes, longitudes, radius=500):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id=
{}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups'][0]['it
ems']

        # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list fo
r item in venue_list])
    nearby_venues.columns = ['Neighbourhood',
                  'Neighborhood Latitude',
                  'Neighborhood Longitude',
                  'Venue',
                  'Venue Category']
```

```
    return(nearby_venues)
```

Generating Venues

In [37]:
```
vnc_ws_venues = getNearbyVenues(names=ws_neig_geo['Neighbourhood'],
                                   latitudes=ws_neig_geo['Latitude'],
                                   longitudes=ws_neig_geo['Longitude']
                                  )
```

```
Shaughnessy
Fairview
Oakridge
Marpole
Kitsilano
Kerrisdale
West Point Grey
Arbutus Ridge
South Cambie
Dunbar-Southlands
```

Data frame containing venues for each neighborhood in West Side

In [38]:
```
print(vnc_ws_venues.shape)
vnc_ws_venues.head()
```

```
(161, 5)
```

Out[38]:

	Neighbourhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Category
0	Shaughnessy	49.246305	-123.138405	Printspot	Print Shop
1	Shaughnessy	49.246305	-123.138405	Devonshire Park	Park
2	Shaughnessy	49.246305	-123.138405	Bus Stop 50209 (10)	Bus Stop

	Neighbourhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Category
3	Shaughnessy	49.246305	-123.138405	Clever Wedding Photographer Vancouver	Photography Studio
4	Shaughnessy	49.246305	-123.138405	Bus Stop 50206 (10)	Bus Stop

Venue Count per neighborhood

In [39]:
```
vnc_ws_venues.groupby('Neighbourhood').count().drop(['Neighborhood Lati
tude','Neighborhood Longitude','Venue Category'], axis = 1)
```

Out[39]:

	Venue
Neighbourhood	
Arbutus Ridge	9
Dunbar-Southlands	12
Fairview	19
Kerrisdale	4
Kitsilano	47
Marpole	34
Oakridge	9
Shaughnessy	6
South Cambie	14
West Point Grey	7

In [40]:
```
print('There are {} uniques categories.'.format(len(vnc_ws_venues['Venu
e Category'].unique())))
```

```
There are 82 uniques categories.
```

Modelling

One Hot Encoding to Analyze Each Neighborhood

In [41]:
```
# one hot encoding
vnc_onehot = pd.get_dummies(vnc_ws_venues[['Venue Category']], prefix=
"", prefix_sep="")

# add neighborhood column back to dataframe
vnc_onehot['Neighbourhood'] = vnc_ws_venues['Neighbourhood']

# move neighborhood column to the first column
fixed_columns = [vnc_onehot.columns[-1]] + list(vnc_onehot.columns[:-1
])
vnc_onehot = vnc_onehot[fixed_columns]

vnc_onehot.head()
```

Out[41]:

	Neighbourhood	American Restaurant	Asian Restaurant	BBQ Joint	Bakery	Bank	Bar	Beach	Breakfast Spot	Bubble Tea Shop
0	Shaughnessy	0	0	0	0	0	0	0	0	0
1	Shaughnessy	0	0	0	0	0	0	0	0	0
2	Shaughnessy	0	0	0	0	0	0	0	0	0
3	Shaughnessy	0	0	0	0	0	0	0	0	0
4	Shaughnessy	0	0	0	0	0	0	0	0	0

5 rows × 83 columns

In [42]:
```
vnc_onehot.shape
```

Out[42]:
```
(161, 83)
```

In [43]:
```
vnc_ws_grouped = vnc_onehot.groupby('Neighbourhood').mean().reset_index
()
vnc_ws_grouped
```

Out[43]:

	Neighbourhood	American Restaurant	Asian Restaurant	BBQ Joint	Bakery	Bank	Bar	Beach	Bre
0	Arbutus Ridge	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
1	Dunbar-Southlands	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
2	Fairview	0.000000	0.052632	0.052632	0.000000	0.000000	0.000000	0.000000	0.(
3	Kerrisdale	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
4	Kitsilano	0.042553	0.021277	0.000000	0.085106	0.000000	0.000000	0.021277	0.(
5	Marpole	0.000000	0.000000	0.000000	0.000000	0.029412	0.029412	0.000000	0.(
6	Oakridge	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
7	Shaughnessy	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(
8	South Cambie	0.000000	0.000000	0.000000	0.000000	0.071429	0.000000	0.000000	0.(
9	West Point Grey	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.(

10 rows × 83 columns

In [44]:
```
vnc_ws_grouped.shape
```

Out[44]: (10, 83)

Top 5 most common venues across neighborhoods

In [45]:
```
num_top_venues = 5

for hood in vnc_ws_grouped['Neighbourhood']:
    print("----"+hood+"----")
```

```
    temp = vnc_ws_grouped[vnc_ws_grouped['Neighbourhood'] == hood].T.re
set_index()
    temp.columns = ['venue','freq']
    temp = temp.iloc[1:]
    temp['freq'] = temp['freq'].astype(float)
    temp = temp.round({'freq': 2})
    print(temp.sort_values('freq', ascending=False).reset_index(drop=Tr
ue).head(num_top_venues))
    print('\n')
```

```
----Arbutus Ridge----
            venue  freq
0    Liquor Store  0.11
1  Discount Store  0.11
2    Dance Studio  0.11
3     Coffee Shop  0.11
4  Sandwich Place  0.11


----Dunbar-Southlands----
                  venue  freq
0         Grocery Store  0.25
1          Liquor Store  0.17
2  Gym / Fitness Center  0.08
3   Japanese Restaurant  0.08
4           Coffee Shop  0.08


----Fairview----
                 venue  freq
0          Coffee Shop  0.11
1    Korean Restaurant  0.05
2     Asian Restaurant  0.05
3  Japanese Restaurant  0.05
4    Indian Restaurant  0.05


----Kerrisdale----
       venue  freq
0       Park  0.25
```

```
1         Café  0.25
2         Pool  0.25
3  Golf Course  0.25
4        Plaza  0.00


----Kitsilano----
                venue  freq
0              Bakery  0.09
1  American Restaurant  0.04
2     Sushi Restaurant  0.04
3           Food Truck  0.04
4       Ice Cream Shop  0.04


----Marpole----
                venue  freq
0     Sushi Restaurant  0.09
1             Bus Stop  0.09
2          Pizza Place  0.06
3  Japanese Restaurant  0.06
4         Dessert Shop  0.06


----Oakridge----
                venue  freq
0   Light Rail Station  0.11
1      Bubble Tea Shop  0.11
2       Sandwich Place  0.11
3          Coffee Shop  0.11
4  Sporting Goods Shop  0.11


----Shaughnessy----
               venue  freq
0            Bus Stop  0.33
1                Park  0.17
2          Print Shop  0.17
3  Photography Studio  0.17
4      Chocolate Shop  0.17
```

```
4        Chocolate Shop  0.17


----South Cambie----
                    venue  freq
0             Coffee Shop  0.29
1        Sushi Restaurant  0.07
2           Shopping Mall  0.07
3  Vietnamese Restaurant  0.07
4        Malay Restaurant  0.07


----West Point Grey----
                   venue  freq
0         Sandwich Place  0.14
1        Harbor / Marina  0.14
2  Gym / Fitness Center  0.14
3                    Gym  0.14
4              Disc Golf  0.14
```

Now let's create the new dataframe and display the top 10 venues for each neighborhood.

In [46]:
```
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

In [47]:
```
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighbourhood']
for ind in np.arange(num_top_venues):
```

```
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indicator
s[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe
neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
neighborhoods_venues_sorted['Neighbourhood'] = vnc_ws_grouped['Neighbou
rhood']

for ind in np.arange(vnc_ws_grouped.shape[0]):
    neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venu
es(vnc_ws_grouped.iloc[ind, :], num_top_venues)

neighborhoods_venues_sorted.head()
```

Out[47]:

	Neighbourhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue
0	Arbutus Ridge	Coffee Shop	Shopping Mall	Discount Store	Fast Food Restaurant	Sandwich Place	Seafood Restaurant	Dance Studio
1	Dunbar-Southlands	Grocery Store	Liquor Store	Japanese Restaurant	Coffee Shop	Pet Store	Café	Bus Stop
2	Fairview	Coffee Shop	Japanese Restaurant	Park	Pet Store	Pharmacy	Pizza Place	Nail Salon
3	Kerrisdale	Café	Park	Golf Course	Pool	Yoga Studio	Disc Golf	Discount Store
4	Kitsilano	Bakery	Japanese Restaurant	Sushi Restaurant	Coffee Shop	Food Truck	French Restaurant	Ice Cream Shop

Cluster Neighbourhoods

In [48]:
```
# set number of clusters
kclusters = 5

vnc_grouped_clustering = vnc_ws_grouped.drop('Neighbourhood', 1)

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(vnc_grouped_c
lustering)

# check cluster labels generated for each row in the dataframe
kmeans.labels_[0:10]
```

Out[48]:
```
array([4, 4, 0, 2, 0, 0, 0, 1, 0, 3], dtype=int32)
```

In [49]:
```
# add clustering labels
neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

vancouver_merged = ws_neig_geo

# merge toronto_grouped with Vancouver data to add latitude/longitude f
or each neighborhood
vancouver_merged = vancouver_merged.join(neighborhoods_venues_sorted.se
t_index('Neighbourhood'), on='Neighbourhood')

vancouver_merged.head()
```

Out[49]:

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue
0	Shaughnessy	West Side	49.246305	-123.138405	1	Bus Stop	Park	Photography Studio
1	Fairview	West Side	49.261956	-123.130408	0	Coffee Shop	Japanese Restaurant	Park
2	Oakridge	West Side	49.226615	-123.122943	0	Coffee Shop	Sandwich Place	Vietnamese Restaurant
3	Marpole	West Side	49.209223	-123.136150	0	Sushi Restaurant	Bus Stop	Japanese Restaurant

	Neighbourhood	Borough	Latitude	Longitude	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue
4	Kitsilano	West Side	49.269410	-123.155267	0	Bakery	Japanese Restaurant	Sushi Restaurant

In [50]:

```
# create map
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=12
)

# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster in zip(vancouver_merged['Latitude'], vancouv
er_merged['Longitude'], vancouver_merged['Neighbourhood'], vancouver_me
rged['Cluster Labels']):
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_h
tml=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```

Out[50]:

Analysis

Examining the resulting Clusters

Cluster 1

In [51]:
```
vancouver_merged.loc[vancouver_merged['Cluster Labels'] == 0, vancouver
_merged.columns[[1] + list(range(5, vancouver_merged.shape[1]))]]
```

Out[51]:

	Borough	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8 C
1	West Side	Coffee Shop	Japanese Restaurant	Park	Pet Store	Pharmacy	Pizza Place	Nail Salon	Re

	Borough	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8 C
2	West Side	Coffee Shop	Sandwich Place	Vietnamese Restaurant	Light Rail Station	Fast Food Restaurant	Sushi Restaurant	Sporting Goods Shop	T
3	West Side	Sushi Restaurant	Bus Stop	Japanese Restaurant	Dessert Shop	Coffee Shop	Pizza Place	Chinese Restaurant	T
4	West Side	Bakery	Japanese Restaurant	Sushi Restaurant	Coffee Shop	Food Truck	French Restaurant	Ice Cream Shop	A Re
8	West Side	Coffee Shop	Shopping Mall	Cantonese Restaurant	Vietnamese Restaurant	Bank	Grocery Store	Park	Re

Cluster 2

In [52]:

```
vancouver_merged.loc[vancouver_merged['Cluster Labels'] == 1, vancouver_merged.columns[[1] + list(range(5, vancouver_merged.shape[1]))]]
```

Out[52]:

	Borough	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
0	West Side	Bus Stop	Park	Photography Studio	Chocolate Shop	Print Shop	Grocery Store	Gym	Disc Golf

Cluster 3

In [53]:

```
vancouver_merged.loc[vancouver_merged['Cluster Labels'] == 2, vancouver_merged.columns[[1] + list(range(5, vancouver_merged.shape[1]))]]
```

Out[53]:

	Borough	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
5	West Side	Café	Park	Golf Course	Pool	Yoga Studio	Disc Golf	Discount Store	Falafel Restaurant

Cluster 4

In [54]:
```
vancouver_merged.loc[vancouver_merged['Cluster Labels'] == 3, vancouver_merged.columns[[1] + list(range(5, vancouver_merged.shape[1]))]]
```

Out[54]:

	Borough	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
6	West Side	Harbor / Marina	Gym / Fitness Center	Gym	Disc Golf	Sandwich Place	Performing Arts Venue	Park	Yoga Studio

Cluster 5

In [55]:
```
vancouver_merged.loc[vancouver_merged['Cluster Labels'] == 4, vancouver_merged.columns[[1] + list(range(5, vancouver_merged.shape[1]))]]
```

Out[55]:

	Borough	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
7	West Side	Coffee Shop	Shopping Mall	Discount Store	Fast Food Restaurant	Sandwich Place	Seafood Restaurant	Dance Studio	Grocery Store

	Borough	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7th Most Common Venue	8th Most Common Venue
9	West Side	Grocery Store	Liquor Store	Japanese Restaurant	Coffee Shop	Pet Store	Café	Bus Stop	Gym

Results and Discussion

The objective of the business problem was to help stakeholders identify one of the safest borough in Vancouver, and an appropriate neighborhood within the borough to set up a commercial establishment especially a Grocery store. This has been achieved by first making use of Vancouver crime data to identify a safe borugh with considerable number of neighborhood for any business to be viable. After selecting the borough it was imperative to choose the right neighborhood where grocery shops were not among venues in a close proximity to each other. We achieved this by grouping the neighborhoods into clusters to assist the stakeholders by providing them with relavent data about venues and safety of a given neighborhood.

Conclusion

We have explored the crime data to understand different types of crimes in all neighborhoods of Vancouver and later categorized them into different boroughs, this helped us group the neighborhoods into boroughs and choose the safest borough first. Once we confirmed the borough the number of neighborhoods for consideration also comes down, we further shortlist the neighborhoods based on the common venues, to choose a neighborhood which best suits the business problem.