FinalCode

April 5, 2021

[1]:
```
import pandas as pd
pd.options.mode.chained_assignment = None  # default='warn'

import requests
import numpy as np

!pip install imblearn
!pip install delayed

!pip install pydotplus
import pydotplus

import matplotlib.pyplot as plt
plt.rc("font", size=14)

import matplotlib.pyplot as plt
import sklearn

import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from imblearn.over_sampling import SMOTE
from sklearn.tree import export_graphviz
from sklearn import preprocessing
from sklearn.model_selection import train_test_split

from numpy import mean
from numpy import std
from sklearn.datasets import make_classification
from sklearn.model_selection import RepeatedKFold
from sklearn.model_selection import cross_val_score
```

```python
from sklearn.model_selection import KFold
from sklearn.preprocessing import MinMaxScaler
from sklearn.svm import SVR

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from math import sqrt
from sklearn import model_selection
from sklearn import metrics
from sklearn.model_selection import LeaveOneOut
from sklearn.model_selection import LeavePOut
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

#Importing all the necessary libraries
```

```
Collecting imblearn
  Using cached imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Collecting imbalanced-learn
  Using cached imbalanced_learn-0.8.0-py3-none-any.whl (206 kB)
Requirement already satisfied: scipy>=0.19.1 in /opt/conda/lib/python3.7/site-
packages (from imbalanced-learn->imblearn) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /opt/conda/lib/python3.7/site-
packages (from imbalanced-learn->imblearn) (1.18.4)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.7/site-
packages (from imbalanced-learn->imblearn) (0.15.1)
Collecting scikit-learn>=0.24
  Using cached scikit_learn-0.24.1-cp37-cp37m-manylinux2010_x86_64.whl (22.3 MB)
Collecting threadpoolctl>=2.0.0
  Using cached threadpoolctl-2.1.0-py3-none-any.whl (12 kB)
Installing collected packages: threadpoolctl, scikit-learn, imbalanced-learn,
imblearn
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 0.22.2.post1
    Uninstalling scikit-learn-0.22.2.post1:
      Successfully uninstalled scikit-learn-0.22.2.post1
Successfully installed imbalanced-learn-0.8.0 imblearn-0.0 scikit-learn-0.24.1
threadpoolctl-2.1.0
Collecting delayed
  Using cached delayed-0.11.0b1-py2.py3-none-any.whl (19 kB)
Collecting redis
  Using cached redis-3.5.3-py2.py3-none-any.whl (72 kB)
```

```
Collecting hiredis
  Using cached hiredis-2.0.0-cp37-cp37m-manylinux2010_x86_64.whl (85 kB)
Installing collected packages: redis, hiredis, delayed
Successfully installed delayed-0.11.0b1 hiredis-2.0.0 redis-3.5.3
Processing /home/jovyan/.cache/pip/wheels/1e/7b/04/7387cf6cc9e48b4a96e361b0be812
f0708b394b821bf8c9c50/pydotplus-2.0.2-py3-none-any.whl
Requirement already satisfied: pyparsing>=2.0.1 in
/opt/conda/lib/python3.7/site-packages (from pydotplus) (2.4.7)
Installing collected packages: pydotplus
Successfully installed pydotplus-2.0.2
```

[2]:

```
COVIDdf = pd.read_csv('COVID19 cases.csv')

#Creating data frame for first dataset
#Importing and reading CSV file
```

[3]:

```
NPdf = pd.read_csv('neighbourhood-profiles-2016-csv.csv', index_col= "_id")

#Creating data frame for second dataset
#Importing and reading CSV file
```

[4]:

```
COVIDdf.info()

#Collecting information on dataset, columns, and data types
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112025 entries, 0 to 112024
Data columns (total 18 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   _id                    112025 non-null  int64
 1   Assigned_ID            112025 non-null  int64
 2   Outbreak Associated    112025 non-null  object
 3   Age Group              111945 non-null  object
 4   Neighbourhood Name     110061 non-null  object
 5   FSA                    110773 non-null  object
 6   Source of Infection    112025 non-null  object
 7   Classification         112025 non-null  object
 8   Episode Date           112025 non-null  object
 9   Reported Date          112025 non-null  object
 10  Client Gender          112025 non-null  object
 11  Outcome                112025 non-null  object
 12  Currently Hospitalized 112025 non-null  object
 13  Currently in ICU       112025 non-null  object
 14  Currently Intubated    112025 non-null  object
 15  Ever Hospitalized      112025 non-null  object
 16  Ever in ICU            112025 non-null  object
 17  Ever Intubated         112025 non-null  object
```

```
dtypes: int64(2), object(16)
memory usage: 15.4+ MB
```

[5]:
```
COVIDdf = COVIDdf.drop(["_id","Assigned_ID", "Outbreak Associated", "FSA",␣
↪"Source of Infection", "Classification",
                         "Episode Date","Currently Hospitalized","Currently in␣
↪ICU","Currently Intubated", "Ever Hospitalized",
                         "Reported Date", "Ever in ICU","Ever Intubated"], axis␣
↪=1)

#Cleaning the dataset
#Removing unnecessary columns in the dataset
```

[6]:
```
print(COVIDdf.isnull().sum())

#Checking dataset for any null values
```

```
Age Group               80
Neighbourhood Name    1964
Client Gender            0
Outcome                  0
dtype: int64
```

[7]:
```
COVIDdf = COVIDdf.dropna()
COVIDdf.count()

#Dropping null values to clean dataset
```

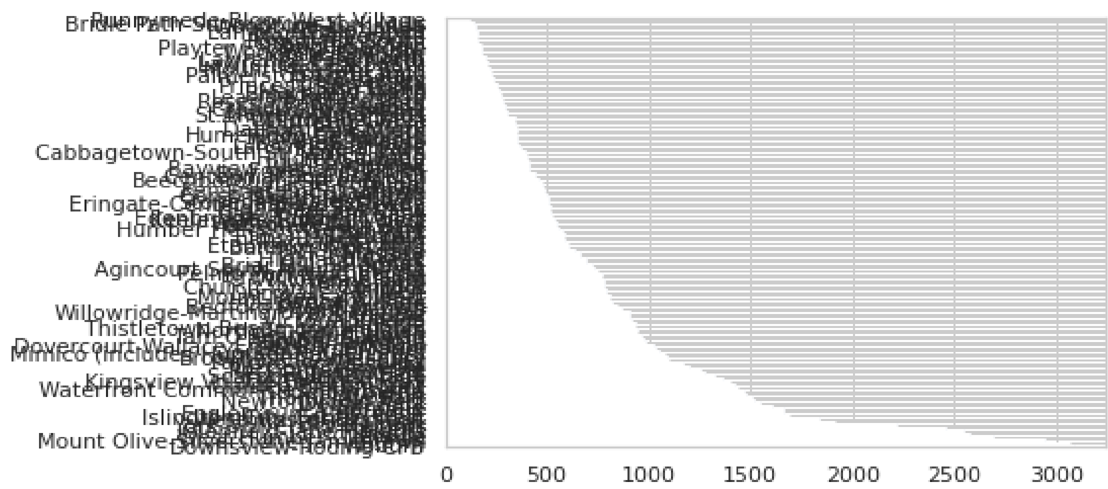
[7]:
```
Age Group             110006
Neighbourhood Name    110006
Client Gender         110006
Outcome               110006
dtype: int64
```

[8]:
```
print(COVIDdf["Neighbourhood Name"].value_counts())
COVIDdf["Neighbourhood Name"].value_counts().plot(kind = 'barh')

#Finding value count of each neighbourhood
#Plotting neighbourhood value count as a graph
#Plotting to see COVID contraction density within each neighbourhood
```

```
Downsview-Roding-CFB                   3092
Woburn                                 3068
Mount Olive-Silverstone-Jamestown      2940
West Humber-Clairville                 2689
Rouge                                  2576
                                       …
Lambton Baby Point                      156
```

```
Blake-Jones                              154
Woodbine-Lumsden                         154
Bridle Path-Sunnybrook-York Mills        141
Runnymede-Bloor West Village             117
Name: Neighbourhood Name, Length: 140, dtype: int64
```

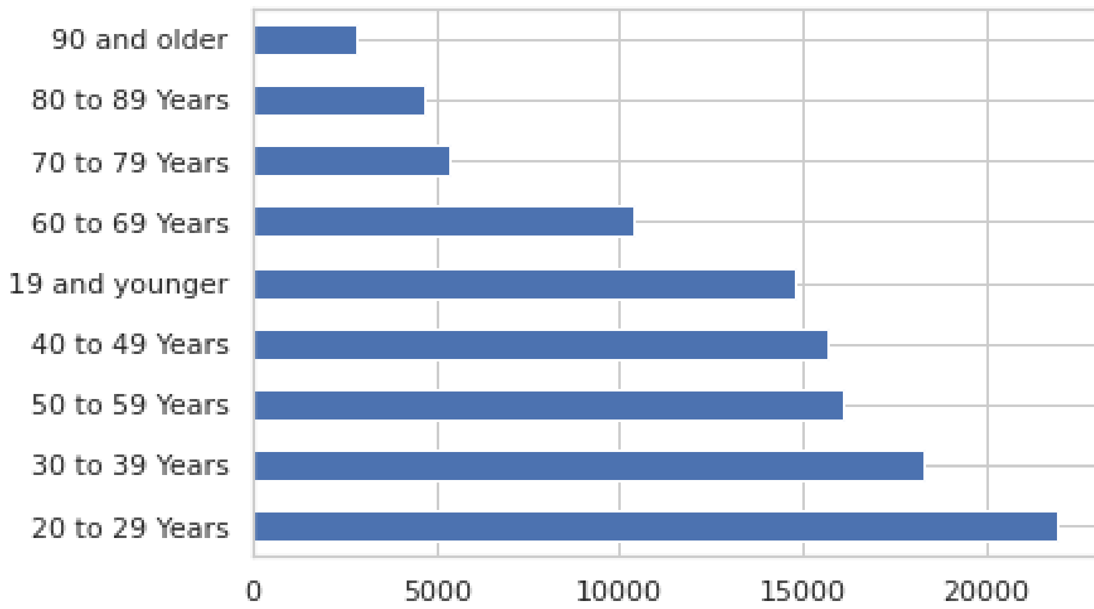
[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50f19159d0>



[9]:
```
print(COVIDdf["Age Group"].value_counts())
COVIDdf["Age Group"].value_counts().plot(kind = 'barh')

#Finding value count of each age group
#Plotting as bar graph
#Plotting to see age distrubution among COVID cases
```

```
20 to 29 Years    21948
30 to 39 Years    18272
50 to 59 Years    16119
40 to 49 Years    15693
19 and younger    14781
60 to 69 Years    10380
70 to 79 Years     5345
80 to 89 Years     4654
90 and older       2814
Name: Age Group, dtype: int64
```

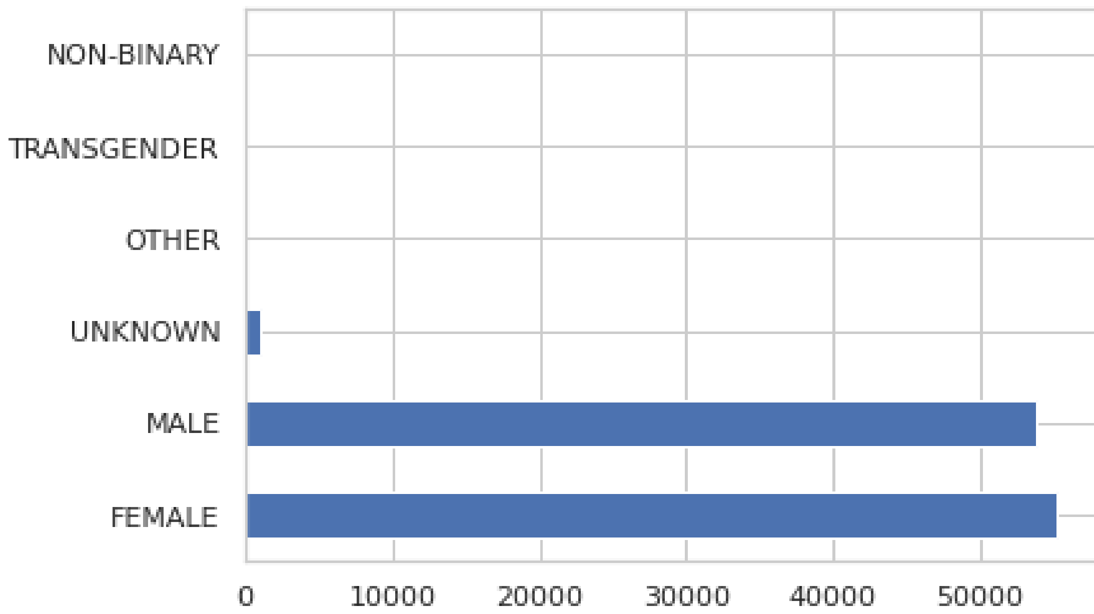
[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50f11648d0>



[10]:
```
print(COVIDdf["Client Gender"].value_counts())
COVIDdf["Client Gender"].value_counts().plot(kind = 'barh')

#Finding value count of each gender group
#Plotting as bar graph
#Plotting to find gender breakdown among COVID cases
```

```
FEMALE         55220
MALE           53821
UNKNOWN          918
OTHER             20
TRANSGENDER       19
NON-BINARY         8
Name: Client Gender, dtype: int64
```

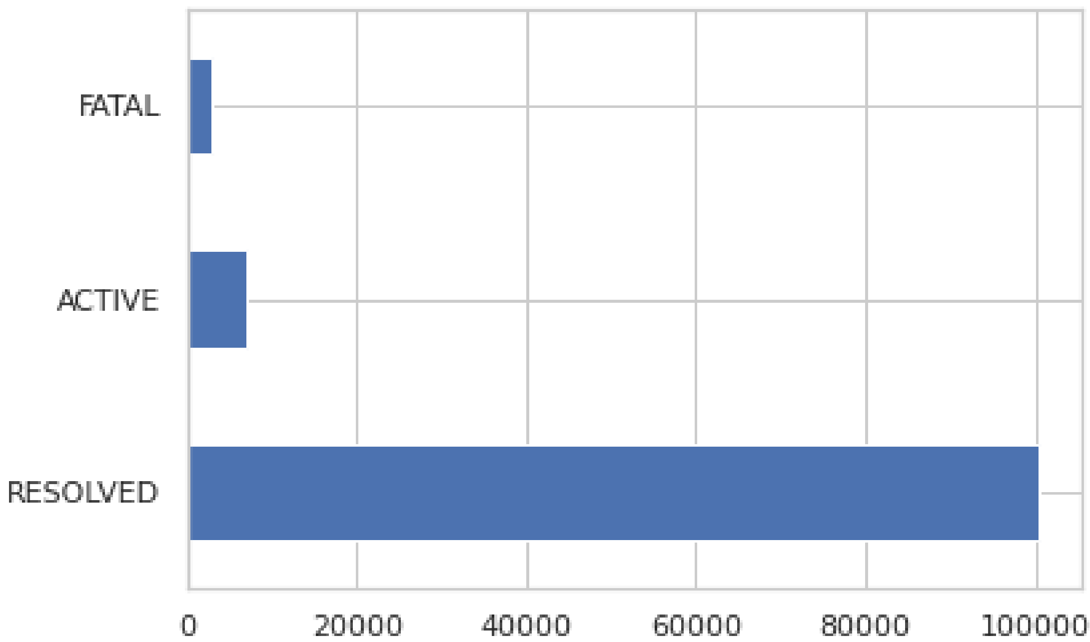
[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50f10c8a10>



[11]:
```
print(COVIDdf["Outcome"].value_counts())
COVIDdf["Outcome"].value_counts().plot(kind = 'barh')

#Finding value count of each outcome group
#Plotting as bar graph
#Plotting to find outcome breakdown among COVID contracted cases
```

```
RESOLVED    100346
ACTIVE        6901
FATAL         2759
Name: Outcome, dtype: int64
```

[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50f1043b10>



[12]:
```
agesexTable = COVIDdf.groupby(["Age Group"])["Client Gender"].value_counts()
agesexTable

#Finding the gender group breakdown within each age
```

[12]:
```
Age Group       Client Gender
19 and younger  MALE              7704
                FEMALE            6941
                UNKNOWN            133
                OTHER                2
                TRANSGENDER          1
20 to 29 Years  MALE             11405
                FEMALE           10339
                UNKNOWN            188
                TRANSGENDER          6
                NON-BINARY           5
                OTHER                5
30 to 39 Years  MALE              9247
                FEMALE            8858
                UNKNOWN            154
                OTHER                5
                TRANSGENDER          5
                NON-BINARY           3
40 to 49 Years  FEMALE            8215
                MALE              7328
```

```
                  UNKNOWN              141
                  TRANSGENDER            5
                  OTHER                  4
50 to 59 Years    FEMALE              8401
                  MALE                7598
                  UNKNOWN              117
                  OTHER                  2
                  TRANSGENDER            1
60 to 69 Years    MALE                5353
                  FEMALE              4971
                  UNKNOWN               55
                  OTHER                  1
70 to 79 Years    FEMALE              2655
                  MALE                2652
                  UNKNOWN               36
                  OTHER                  1
                  TRANSGENDER            1
80 to 89 Years    FEMALE              2847
                  MALE                1759
                  UNKNOWN               48
90 and older      FEMALE              1993
                  MALE                 775
                  UNKNOWN               46
Name: Client Gender, dtype: int64
```

[13]:

```
NPagesexTable = COVIDdf.groupby(["Neighbourhood Name", "Age Group"])["Client␣
↪Gender"].value_counts()
print(NPagesexTable)

#Finding the age and gender group breakdown within each neighbourhood
```

```
Neighbourhood Name  Age Group       Client Gender
Agincourt North     19 and younger  FEMALE              52
                                    MALE                43
                    20 to 29 Years  FEMALE             105
                                    MALE               104
                                    UNKNOWN              1
                                                      …
Yorkdale-Glen Park  70 to 79 Years  FEMALE              27
                    80 to 89 Years  FEMALE              81
                                    MALE                41
                    90 and older    FEMALE              63
                                    MALE                16
Name: Client Gender, Length: 2989, dtype: int64
```

[14]:

```
NPdf.reset_index(drop=True, inplace=True)
NPdf.set_index("Category", inplace=True)
```

```python
NPdf.head()

#Cleaning the dataset
#Resetting index to narrow down unwanted columns + rows
```

[14]:

```
                                              Topic  \
Category
Neighbourhood Information  Neighbourhood Information
Neighbourhood Information  Neighbourhood Information
Population                  Population and dwellings
Population                  Population and dwellings
Population                  Population and dwellings

                                                Data Source  \
Category
Neighbourhood Information                   City of Toronto
Neighbourhood Information                   City of Toronto
Population                 Census Profile 98-316-X2016001
Population                 Census Profile 98-316-X2016001
Population                 Census Profile 98-316-X2016001

                                     Characteristic City of Toronto  \
Category
Neighbourhood Information         Neighbourhood Number             NaN
Neighbourhood Information         TSNS2020 Designation             NaN
Population                            Population, 2016       2,731,571
Population                            Population, 2011       2,615,060
Population                 Population Change 2011-2016           4.50%

                          Agincourt North Agincourt South-Malvern West  \
Category
Neighbourhood Information             129                          128
Neighbourhood Information  No Designation               No Designation
Population                         29,113                       23,757
Population                         30,279                       21,988
Population                         -3.90%                        8.00%

                                Alderwood           Annex Banbury-Don Mills  \
Category
Neighbourhood Information              20              95                42
Neighbourhood Information  No Designation  No Designation    No Designation
Population                         12,054          30,526            27,695
Population                         11,904          29,177            26,918
Population                          1.30%           4.60%             2.90%

                           Bathurst Manor  … Willowdale West  \
Category                                   …
```

```
Neighbourhood Information               34  …            37
Neighbourhood Information    No Designation  …  No Designation
Population                          15,873  …        16,936
Population                          15,434  …        15,004
Population                           2.80%  …        12.90%

                          Willowridge-Martingrove-Richview  Woburn  \
Category
Neighbourhood Information                                7     137
Neighbourhood Information                   No Designation     NIA
Population                                          22,156  53,485
Population                                          21,343  53,350
Population                                           3.80%   0.30%

                          Woodbine Corridor Woodbine-Lumsden        Wychwood  \
Category
Neighbourhood Information                64               60              94
Neighbourhood Information    No Designation   No Designation  No Designation
Population                           12,541            7,865          14,349
Population                           11,703            7,826          13,986
Population                            7.20%            0.50%           2.60%

                           Yonge-Eglinton  Yonge-St.Clair  \
Category
Neighbourhood Information             100              97
Neighbourhood Information  No Designation  No Designation
Population                         11,817          12,528
Population                         10,578          11,652
Population                         11.70%           7.50%

                          York University Heights      Yorkdale-Glen Park
Category
Neighbourhood Information                      27                      31
Neighbourhood Information                     NIA  Emerging Neighbourhood
Population                                 27,593                  14,804
Population                                 27,713                  14,687
Population                                 -0.40%                   0.80%

[5 rows x 144 columns]
```

[15]:

```
NPdf = NPdf.drop(['Aboriginal peoples','Education', 'Families, households and␣
↪marital status','Housing','Immigration and citizenship',
                  'Journey to work', 'Labour','Language', 'Language of␣
↪work','Mobility', 'Neighbourhood Information', 'Population',
                  'Visible minority'])

#Dropping all the unwanted columns in the NPdf dataset
```

[16]:
```
NPdf.reset_index(drop=True, inplace=True)
NPdf.set_index("Topic", inplace=True)

#Resetting the index as "Topic" to further clean dataset
```

[17]:
```
NPdf = NPdf.drop(['Income of individuals in 2015', 'Income of economic families␣
↪in 2015', 'Income sources', 'Income taxes',
                  'Low income in 2015'])

#Data cleaning
#Dropping more unwanted columns
```

[18]:
```
NPdf.reset_index(drop=True, inplace=True)
NPdf = NPdf.drop(["Data Source"], axis=1)
NPdf = NPdf.drop_duplicates(subset = "Characteristic", keep = "first")

#Data cleaning
#Resetting the index
#Removing duplicate values under "Characteristic" column
```

[19]:
```
NPdf.set_index("Characteristic", inplace=True)

#Data cleaning
#Setting index column as "Characteristic"
```

[20]:
```
NPdf = NPdf.dropna()

#Dropping all null values
```

[21]:
```
NPdf = NPdf.drop(['Total - Income statistics in 2015 for private households by␣
↪household size - 100% data',
                  '    Total - Income statistics in 2015 for one-person private␣
↪households - 100% data'])

NPdf = NPdf.drop(['    Total - Income statistics in 2015 for two-or-more-person␣
↪private households - 100% data',
                  'Total - Income statistics in 2015for private households by␣
↪household size - 25% sampledata'])

NPdf = NPdf.drop(['  Average after-tax income of households in 2015 ($)',
                  '    Total - Income statistics in 2015 for one-person private␣
↪households - 25% sample data'])

NPdf= NPdf.drop(['Total - Household total income groups in 2015 for private␣
↪households - 100% data',
```

```
                 'Total - Household after-tax income groups in 2015 for private␣
↪households - 100% data',
                 '    Total - Income statistics in 2015 for two-or-more-person␣
↪private households - 25% sample data'])

#Dropping further columns to filter out specific income related data values
```

[22]:
```
NPdf = NPdf.drop_duplicates(keep = "first")

#Dropping any duplicates within dataset
#Choosing the first value to stay within dataframe
```

[23]:
```
NPdf= NPdf.T

#Transposing dataframe to be able to get the characterisitcs as columns, with␣
↪neighbourhoods as rows
```

[24]:
```
NPdf = NPdf.drop(['    $150,000 to $199,999', '    $100,000 to $124,999', '   ␣
↪$125,000 to $149,999', '    $200,000 and over'], axis = 1)

#Dropping unnecessary columns to filter out specific income related data values␣
↪for dataset
```

[25]:
```
for col in NPdf.columns:
    print(col)

#Printing all the income related and ethnicity related columns within dataset
```

```
  $15,000 to $19,999
  Under $5,000
  $5,000 to $9,999
  $10,000 to $14,999
  $20,000 to $24,999
  $25,000 to $29,999
  $30,000 to $34,999
  $35,000 to $39,999
  $40,000 to $44,999
  $45,000 to $49,999
  $50,000 to $59,999
  $60,000 to $69,999
  $70,000 to $79,999
  $80,000 to $89,999
  $90,000 to $99,999
  $100,000 and over
 Guadeloupean
 Scottish
Total - Ethnic origin for the population in private households - 25% sample data
```

North American Aboriginal origins
First Nations (North American Indian)
Inuit
Mtis
Other North American origins
Acadian
American
Canadian
New Brunswicker
Newfoundlander
Nova Scotian
Ontarian
Qubcois
Portuguese
Other North American origins; n.i.e.
European origins
British Isles origins
Channel Islander
Cornish
English
Irish
Manx
Welsh
British Isles origins; n.i.e.
French origins
Alsatian
Breton
Corsican
French
Western European origins (except French origins)
Austrian
Bavarian
Belgian
Dutch
Flemish
Frisian
German
Luxembourger
Swiss
Western European origins; n.i.e.
Northern European origins (except British Isles origins)
Danish
Finnish
Icelandic
Norwegian
Swedish
Northern European origins; n.i.e.
Eastern European origins

Bulgarian
Caribbean origins
Byelorussian
Czech
Czechoslovakian; n.o.s.
Estonian
Hungarian
Latvian
Lithuanian
Moldovan
Polish
Romanian
Russian
Slovak
Ukrainian
Eastern European origins; n.i.e.
Southern European origins
Albanian
Bosnian
Catalan
Croatian
Cypriot
Greek
Italian
Kosovar
Macedonian
Maltese
Montenegrin
Serbian
Sicilian
Slovenian
Spanish
Yugoslavian; n.o.s.
Southern European origins; n.i.e.
Antiguan
Other European origins
Basque
Jewish
Roma (Gypsy)
Slavic; n.o.s.
Other European origins; n.i.e.
Bahamian
Barbadian
Bermudan
Carib
Cuban
Dominican
Grenadian

Haitian
Jamaican
Kittitian/Nevisian
Martinican
Montserratan
Puerto Rican
St. Lucian
Arawak
Trinidadian/Tobagonian
Vincentian/Grenadinian
West Indian; n.o.s.
Caribbean origins; n.i.e.
Latin; Central and South American origins
Aboriginal from Central/South America (except Arawak and Maya)
Argentinian
Belizean
Bolivian
Brazilian
Chilean
Colombian
Costa Rican
Ecuadorian
Guatemalan
Guyanese
Hispanic
Honduran
Maya
Beninese
Mexican
Nicaraguan
Panamanian
Paraguayan
Peruvian
Salvadorean
Uruguayan
Venezuelan
Latin; Central and South American origins; n.i.e.
African origins
Central and West African origins
Akan
Angolan
Ashanti
Burkinabe
Cameroonian
Chadian
Congolese
Edo
Ewe

Gabonese
Gambian
Ghanaian
Guinean
Ibo
Ivorian
Liberian
Malian
Malink
Nigerian
Peulh
Senegalese
Sierra Leonean
Togolese
Wolof
Yoruba
Central and West African origins; n.i.e.
Maure
North African origins
Algerian
Berber
Coptic
Dinka
Egyptian
Libyan
Moroccan
Sudanese
Tunisian
North African origins; n.i.e.
Southern and East African origins
Afrikaner
Amhara
Bantu; n.o.s.
Burundian
Djiboutian
Eritrean
Ethiopian
Harari
Kenyan
Malagasy
Mauritian
Oromo
Rwandan
Seychellois
Somali
South African
Tanzanian
Tigrian

Ugandan
Zambian
Zimbabwean
Zulu
Southern and East African origins; n.i.e.
Other African origins
Black; n.o.s.
Other African origins; n.i.e.
Asian origins
West Central Asian and Middle Eastern origins
Afghan
Arab; n.o.s.
Armenian
Assyrian
Azerbaijani
Georgian
Hazara
Iranian
Hmong
Iraqi
Israeli
Jordanian
Kazakh
Kurd
Kuwaiti
Kyrgyz
Maori
Lebanese
Palestinian
Pashtun
Saudi Arabian
Syrian
Tajik
Tatar
Turk
Turkmen
Uighur
Uzbek
Yemeni
West Central Asian and Middle Eastern origins; n.i.e.
South Asian origins
Bangladeshi
Bengali
Bhutanese
East Indian
Goan
Gujarati
Kashmiri

```
Nepali
Pakistani
Punjabi
Sinhalese
Sri Lankan
Indonesian
Tamil
South Asian origins; n.i.e.
East and Southeast Asian origins
Burmese
Cambodian (Khmer)
Chinese
Filipino
Japanese
Karen
Korean
Laotian
Malaysian
Mongolian
Singaporean
Taiwanese
Thai
Tibetan
Vietnamese
East and Southeast Asian origins; n.i.e.
Hawaiian
Other Asian origins
Other Asian origins; n.i.e.
Oceania origins
Australian
New Zealander
Pacific Islands origins
Fijian
Polynesian; n.o.s.
Samoan
Pacific Islands origins; n.i.e.
```

[26]:

```
data = ['  Under $5,000', '  $5,000 to $9,999', '  $10,000 to $14,999', ' ␣
↪$15,000 to $19,999', '  $20,000 to $24,999',
                                  '  $25,000 to $29,999', '  $30,000 to␣
↪$34,999', '  $35,000 to $39,999','  $40,000 to $44,999','  $45,000 to␣
↪$49,999',
                                  '  $50,000 to $59,999', '  $60,000 to␣
↪$69,999', '  $70,000 to $79,999', '  $80,000 to $89,999', '  $90,000 to␣
↪$99,999',
                                  '  $100,000 and over']
IncomeProfile = NPdf.loc[:,data]
```

```
#Creating new dataframe specifically for Income Profile from Neighbourhood␣
↪Profile dataframe
```

[27]:
```
IncomeProfile.dtypes

#Returning data types of columns in dataframe
```

[27]:
```
Characteristic
  Under $5,000           object
  $5,000 to $9,999       object
  $10,000 to $14,999     object
  $15,000 to $19,999     object
  $20,000 to $24,999     object
  $25,000 to $29,999     object
  $30,000 to $34,999     object
  $35,000 to $39,999     object
  $40,000 to $44,999     object
  $45,000 to $49,999     object
  $50,000 to $59,999     object
  $60,000 to $69,999     object
  $70,000 to $79,999     object
  $80,000 to $89,999     object
  $90,000 to $99,999     object
  $100,000 and over      object
dtype: object
```

[28]:
```
IncomeProfile = IncomeProfile.replace(',', '', regex=True)

#Fixing cell values of IncomeProfile
#Removing the ',' in cell values
```

[29]:
```
c = IncomeProfile.select_dtypes(object).columns
IncomeProfile[c] = IncomeProfile[c].apply(pd.to_numeric,errors='coerce')

#Converting cell values from object into integer
```

[30]:
```
IncomeProfile.dtypes

#Checking to see values converted from object into integer
```

[30]:
```
Characteristic
  Under $5,000           int64
  $5,000 to $9,999       int64
  $10,000 to $14,999     int64
  $15,000 to $19,999     int64
  $20,000 to $24,999     int64
```

```
  $25,000 to $29,999    int64
  $30,000 to $34,999    int64
  $35,000 to $39,999    int64
  $40,000 to $44,999    int64
  $45,000 to $49,999    int64
  $50,000 to $59,999    int64
  $60,000 to $69,999    int64
  $70,000 to $79,999    int64
  $80,000 to $89,999    int64
  $90,000 to $99,999    int64
  $100,000 and over     int64
dtype: object
```

[31]:
```
IncomeProfile = IncomeProfile.T
```

[32]:
```
AvgIncomeProfile = IncomeProfile

#Creating new dataframe
#Copying IncomeProfile df into AvgIncomeProfile df
```

[33]:
```
AvgIncomeProfile.index

#Checking index column in new dataframe
```

[33]:
```
Index(['  Under $5,000', '  $5,000 to $9,999', '  $10,000 to $14,999',
       '  $15,000 to $19,999', '  $20,000 to $24,999', '  $25,000 to $29,999',
       '  $30,000 to $34,999', '  $35,000 to $39,999', '  $40,000 to $44,999',
       '  $45,000 to $49,999', '  $50,000 to $59,999', '  $60,000 to $69,999',
       '  $70,000 to $79,999', '  $80,000 to $89,999', '  $90,000 to $99,999',
       '  $100,000 and over'],
      dtype='object', name='Characteristic')
```

[34]:
```
AvgIncomeProfile = AvgIncomeProfile.drop(['City of Toronto'], axis=1)

#Dropping unnecessary column for this dataframe
```

[35]:
```
AvgIncomeProfile['Average'] = 0

#Creating new column 'Average' for AvgIncomeProfile
```

[36]:
```
AvgIncomeProfile.index
```

[36]:
```
Index(['  Under $5,000', '  $5,000 to $9,999', '  $10,000 to $14,999',
       '  $15,000 to $19,999', '  $20,000 to $24,999', '  $25,000 to $29,999',
       '  $30,000 to $34,999', '  $35,000 to $39,999', '  $40,000 to $44,999',
       '  $45,000 to $49,999', '  $50,000 to $59,999', '  $60,000 to $69,999',
       '  $70,000 to $79,999', '  $80,000 to $89,999', '  $90,000 to $99,999',
```

```
       '  $100,000 and over'],
      dtype='object', name='Characteristic')
```

[37]:
```
AvgIncomeProfile.loc['  Under $5,000']['Average'] = 2500
AvgIncomeProfile.loc['  $5,000 to $9,999']['Average'] = 7500
AvgIncomeProfile.loc['  $10,000 to $14,999']['Average'] = 12500
AvgIncomeProfile.loc['  $15,000 to $19,999']['Average'] = 17500
AvgIncomeProfile.loc['  $20,000 to $24,999']['Average'] = 22500
AvgIncomeProfile.loc['  $25,000 to $29,999']['Average'] = 27500
AvgIncomeProfile.loc['  $30,000 to $34,999']['Average'] = 32500
AvgIncomeProfile.loc['  $35,000 to $39,999']['Average'] = 37500
AvgIncomeProfile.loc['  $40,000 to $44,999']['Average'] = 42500
AvgIncomeProfile.loc['  $45,000 to $49,999']['Average'] = 47500
AvgIncomeProfile.loc['  $50,000 to $59,999']['Average'] = 55000
AvgIncomeProfile.loc['  $60,000 to $69,999']['Average'] = 65000
AvgIncomeProfile.loc['  $70,000 to $79,999']['Average'] = 75000
AvgIncomeProfile.loc['  $80,000 to $89,999']['Average'] = 85000
AvgIncomeProfile.loc['  $90,000 to $99,999']['Average'] = 95000
AvgIncomeProfile.loc['  $100,000 and over']['Average'] = 100000

#Adding values into 'Average' column with median value of each Income range␣
 ↪(index column)
```

[38]:
```
for col in AvgIncomeProfile.columns:
    if col != 'Average':
        AvgIncomeProfile[col] *= AvgIncomeProfile['Average']

#Creating 'for' loop to calculate product of 'Average' column with each cell␣
 ↪value
#Cell value represents number of people that have x income range within each␣
 ↪neighbourhood
#First step to finding average income of each neighbourhood
```

[39]:
```
AvgIncomeProfile

#Printing dataframe to see if loop produced required results
```

[39]:
```
                      Agincourt North  Agincourt South-Malvern West  \
Characteristic
  Under $5,000                 387500                        787500
  $5,000 to $9,999             787500                       1050000
  $10,000 to $14,999          2000000                       2437500
  $15,000 to $19,999          7875000                       4637500
  $20,000 to $24,999          7200000                       7087500
  $25,000 to $29,999         14850000                      11000000
  $30,000 to $34,999         13650000                      12025000
  $35,000 to $39,999         17062500                      14437500
```

$40,000 to $44,999	17850000	15725000
$45,000 to $49,999	20662500	19712500
$50,000 to $59,999	44000000	42350000
$60,000 to $69,999	45500000	41925000
$70,000 to $79,999	47625000	44625000
$80,000 to $89,999	44625000	43350000
$90,000 to $99,999	48925000	38475000
$100,000 and over	250500000	203000000

	Alderwood	Annex	Banbury-Don Mills	Bathurst Manor	\
Characteristic					
Under $5,000	137500	2125000	662500	325000	
$5,000 to $9,999	337500	3637500	1162500	637500	
$10,000 to $14,999	1000000	8187500	2937500	1937500	
$15,000 to $19,999	1575000	12950000	7000000	7262500	
$20,000 to $24,999	3262500	13950000	8887500	5512500	
$25,000 to $29,999	4125000	14575000	12237500	6325000	
$30,000 to $34,999	5037500	17062500	13162500	7150000	
$35,000 to $39,999	6375000	20812500	17625000	9562500	
$40,000 to $44,999	6800000	22950000	19550000	10200000	
$45,000 to $49,999	7837500	23987500	19000000	11637500	
$50,000 to $59,999	18425000	55000000	51150000	22000000	
$60,000 to $69,999	19500000	58500000	57525000	27950000	
$70,000 to $79,999	24000000	59625000	58500000	29250000	
$80,000 to $89,999	23375000	60775000	55675000	26350000	
$90,000 to $99,999	23750000	57475000	57475000	28500000	
$100,000 and over	191500000	589500000	461500000	202500000	

	Bay Street Corridor	Bayview Village	\
Characteristic			
Under $5,000	6275000	1462500	
$5,000 to $9,999	5512500	1950000	
$10,000 to $14,999	9250000	3625000	
$15,000 to $19,999	12075000	5775000	
$20,000 to $24,999	13050000	8212500	
$25,000 to $29,999	15400000	10175000	
$30,000 to $34,999	15925000	11700000	
$35,000 to $39,999	19312500	12375000	
$40,000 to $44,999	18700000	14875000	
$45,000 to $49,999	19475000	15912500	
$50,000 to $59,999	46750000	35750000	
$60,000 to $69,999	54600000	42575000	
$70,000 to $79,999	61125000	46125000	
$80,000 to $89,999	59925000	44200000	
$90,000 to $99,999	54150000	47025000	
$100,000 and over	360000000	301000000	

Characteristic	Bayview Woods-Steeles	Bedford Park-Nortown	...
Under $5,000	225000	350000	...
$5,000 to $9,999	525000	637500	...
$10,000 to $14,999	1000000	1875000	...
$15,000 to $19,999	2275000	4637500	...
$20,000 to $24,999	4500000	5175000	...
$25,000 to $29,999	4675000	5912500	...
$30,000 to $34,999	6662500	7475000	...
$35,000 to $39,999	6750000	8062500	...
$40,000 to $44,999	8500000	9562500	...
$45,000 to $49,999	7837500	11400000	...
$50,000 to $59,999	18150000	24200000	...
$60,000 to $69,999	18850000	25675000	...
$70,000 to $79,999	19875000	29250000	...
$80,000 to $89,999	21675000	30600000	...
$90,000 to $99,999	23750000	26125000	...
$100,000 and over	183000000	474500000	...

Characteristic	Willowridge-Martingrove-Richview	Woburn
Under $5,000	225000	1087500
$5,000 to $9,999	637500	3412500
$10,000 to $14,999	1937500	8562500
$15,000 to $19,999	5162500	20475000
$20,000 to $24,999	8212500	18562500
$25,000 to $29,999	10862500	26400000
$30,000 to $34,999	12187500	29575000
$35,000 to $39,999	14062500	35625000
$40,000 to $44,999	16150000	40587500
$45,000 to $49,999	17575000	38712500
$50,000 to $59,999	34650000	94875000
$60,000 to $69,999	39650000	91325000
$70,000 to $79,999	41625000	93000000
$80,000 to $89,999	43350000	90950000
$90,000 to $99,999	41325000	82175000
$100,000 and over	289500000	398000000

Characteristic	Woodbine Corridor	Woodbine-Lumsden	Wychwood
Under $5,000	162500	137500	300000
$5,000 to $9,999	937500	487500	900000
$10,000 to $14,999	3312500	1312500	2312500
$15,000 to $19,999	5512500	3412500	6912500
$20,000 to $24,999	6187500	3375000	7312500
$25,000 to $29,999	5225000	3712500	7425000
$30,000 to $34,999	7312500	3575000	7312500

$35,000 to $39,999	6000000	4875000	8437500
$40,000 to $44,999	8075000	4887500	9562500
$45,000 to $49,999	8550000	5462500	10687500
$50,000 to $59,999	16225000	13475000	22000000
$60,000 to $69,999	18850000	13325000	25350000
$70,000 to $79,999	17625000	15000000	23625000
$80,000 to $89,999	19975000	15725000	25500000
$90,000 to $99,999	20900000	18525000	26125000
$100,000 and over	219000000	124500000	189500000

Characteristic	Yonge-Eglinton	Yonge-St.Clair	York University Heights \
Under $5,000	512500	537500	862500
$5,000 to $9,999	787500	900000	1725000
$10,000 to $14,999	1812500	2312500	4250000
$15,000 to $19,999	3062500	3850000	9100000
$20,000 to $24,999	3937500	5062500	11812500
$25,000 to $29,999	4675000	6050000	14437500
$30,000 to $34,999	6175000	6987500	19175000
$35,000 to $39,999	7875000	8812500	21375000
$40,000 to $44,999	8287500	11050000	22525000
$45,000 to $49,999	8550000	12350000	25175000
$50,000 to $59,999	20350000	27775000	52250000
$60,000 to $69,999	24375000	28925000	54925000
$70,000 to $79,999	24375000	31125000	46125000
$80,000 to $89,999	23375000	28900000	51425000
$90,000 to $99,999	23275000	30875000	48925000
$100,000 and over	234000000	286500000	195500000

Characteristic	Yorkdale-Glen Park	Average
Under $5,000	250000	2500
$5,000 to $9,999	487500	7500
$10,000 to $14,999	1500000	12500
$15,000 to $19,999	3850000	17500
$20,000 to $24,999	5512500	22500
$25,000 to $29,999	6600000	27500
$30,000 to $34,999	9425000	32500
$35,000 to $39,999	10312500	37500
$40,000 to $44,999	11900000	42500
$45,000 to $49,999	10925000	47500
$50,000 to $59,999	25850000	55000
$60,000 to $69,999	24050000	65000
$70,000 to $79,999	27000000	75000
$80,000 to $89,999	25075000	85000
$90,000 to $99,999	22325000	95000
$100,000 and over	155000000	100000

```
[16 rows x 141 columns]
```

[40]:
```
FinalIncomeProfile = pd.DataFrame(index = ['FinalAvg'], columns = IncomeProfile.
↪columns)
FinalIncomeProfile = FinalIncomeProfile.drop(['City of Toronto'], axis = 1)

#Creating new dataframe to calculate average income of each neighbourhood
#Using columns from IncomeProfile DF to copy into new dataframe
#Dropping columm "City of Toronto", as it is not needed
```

[41]:
```
FinalIncomeProfile

#Printing dataframe to visualize
```

[41]:
```
         Agincourt North Agincourt South-Malvern West Alderwood Annex  \
FinalAvg             NaN                          NaN       NaN   NaN

         Banbury-Don Mills Bathurst Manor Bay Street Corridor Bayview Village  \
FinalAvg               NaN            NaN                 NaN             NaN

         Bayview Woods-Steeles Bedford Park-Nortown  … Willowdale West  \
FinalAvg                   NaN                  NaN  …             NaN

         Willowridge-Martingrove-Richview Woburn Woodbine Corridor  \
FinalAvg                              NaN    NaN               NaN

         Woodbine-Lumsden Wychwood Yonge-Eglinton Yonge-St.Clair  \
FinalAvg              NaN      NaN            NaN            NaN

         York University Heights Yorkdale-Glen Park
FinalAvg                     NaN                NaN

[1 rows x 140 columns]
```

[42]:
```
for col in FinalIncomeProfile.columns:
    FinalIncomeProfile[col] = round((AvgIncomeProfile[col].sum()/
↪IncomeProfile[col].sum()),2)

#Creating for loop to calculate last step to finding average income for each␣
↪neighbourhood
#Weighted average
```

[43]:
```
FinalIncomeProfile

#Printing dataframe with newly calculated averages
```

[43]:
```
             Agincourt North  Agincourt South-Malvern West  Alderwood     Annex  \
FinalAvg            63840.26                      61861.54   73110.09  64160.38

          Banbury-Don Mills  Bathurst Manor  Bay Street Corridor  \
FinalAvg            69756.2        65366.26             51264.12

          Bayview Village  Bayview Woods-Steeles  Bedford Park-Nortown  …  \
FinalAvg         63312.76               69692.14              77376.45  …

          Willowdale West  Willowridge-Martingrove-Richview    Woburn  \
FinalAvg         59240.07                          67736.21  58190.57

          Woodbine Corridor  Woodbine-Lumsden  Wychwood  Yonge-Eglinton  \
FinalAvg           66700.27          67184.78  63372.24        69617.08

          Yonge-St.Clair  York University Heights  Yorkdale-Glen Park
FinalAvg        69789.01                 56878.07            63622.54

[1 rows x 140 columns]
```

[44]:
```python
#Want to merge FinalAvg values from FinalIncomeProfile into COVIDdf for further
 ↪analysis
```

[45]:
```python
Neighbourhoodlist = FinalIncomeProfile.T

#Creating new variable 'Neighbourhoodlist'
#Transoposing FinalIncomeProfile in order to get neighbourhood names as rows,
 ↪with FinalAvg as column
```

[46]:
```python
Neighbourhoodlist
```

[46]:
```
                              FinalAvg
Agincourt North               63840.26
Agincourt South-Malvern West  61861.54
Alderwood                     73110.09
Annex                         64160.38
Banbury-Don Mills             69756.20
…                                …
Wychwood                      63372.24
Yonge-Eglinton                69617.08
Yonge-St.Clair                69789.01
York University Heights       56878.07
Yorkdale-Glen Park            63622.54

[140 rows x 1 columns]
```

[47]:
```
Neighbourhoodlist = Neighbourhoodlist.to_dict()

#Changing Neighbourhoodlist from dataframe into dictionary
#Making this change will make it possible to add FinalAvg onto COVIDdf
```

[48]:
```
Neighbourhoodlist
```

[48]:
```
{'FinalAvg': {'Agincourt North': 63840.26,
  'Agincourt South-Malvern West': 61861.54,
  'Alderwood': 73110.09,
  'Annex': 64160.38,
  'Banbury-Don Mills': 69756.2,
  'Bathurst Manor': 65366.26,
  'Bay Street Corridor': 51264.12,
  'Bayview Village': 63312.76,
  'Bayview Woods-Steeles': 69692.14,
  'Bedford Park-Nortown': 77376.45,
  'Beechborough-Greenbrook': 51937.15,
  'Bendale': 60398.65,
  'Birchcliffe-Cliffside': 66908.65,
  'Black Creek': 51197.06,
  'Blake-Jones': 62588.57,
  'Briar Hill-Belgravia': 61088.0,
  'Bridle Path-Sunnybrook-York Mills': 86982.23,
  'Broadview North': 56875.55,
  'Brookhaven-Amesbury': 56687.84,
  'Cabbagetown-South St. James Town': 60437.4,
  'Caledonia-Fairbank': 64444.82,
  'Casa Loma': 72599.08,
  'Centennial Scarborough': 83986.87,
  'Church-Yonge Corridor': 56852.79,
  'Clairlea-Birchmount': 65765.8,
  'Clanton Park': 65532.44,
  'Cliffcrest': 67150.13,
  'Corso Italia-Davenport': 66377.27,
  'Danforth': 66859.06,
  'Danforth East York': 69283.17,
  'Don Valley Village': 62859.47,
  'Dorset Park': 59826.11,
  'Dovercourt-Wallace Emerson-Junction': 62881.79,
  'Downsview-Roding-CFB': 60070.48,
  'Dufferin Grove': 60343.63,
  'East End-Danforth': 63933.42,
  'Edenbridge-Humber Valley': 71983.17,
  'Eglinton East': 57114.9,
  'Elms-Old Rexdale': 61281.06,
  'Englemount-Lawrence': 59524.53,
```

```
'Eringate-Centennial-West Deane': 75143.74,
'Etobicoke West Mall': 63584.24,
'Flemingdon Park': 53544.73,
'Forest Hill North': 66120.29,
'Forest Hill South': 72800.2,
'Glenfield-Jane Heights': 56592.74,
'Greenwood-Coxwell': 64479.77,
'Guildwood': 74806.49,
'Henry Farm': 57246.26,
'High Park North': 64157.21,
'High Park-Swansea': 70928.9,
'Highland Creek': 78415.54,
'Hillcrest Village': 64366.2,
'Humber Heights-Westmount': 66677.75,
'Humber Summit': 62971.15,
'Humbermede': 60380.6,
'Humewood-Cedarvale': 61914.45,
'Ionview': 56666.67,
'Islington-City Centre West': 67355.18,
'Junction Area': 67348.67,
'Keelesdale-Eglinton West': 60000.0,
'Kennedy Park': 54866.82,
'Kensington-Chinatown': 50219.07,
'Kingsview Village-The Westway': 62062.7,
'Kingsway South': 84394.99,
'Lambton Baby Point': 68666.13,
"L'Amoreaux": 60667.94,
'Lansing-Westgate': 68218.61,
'Lawrence Park North': 81940.85,
'Lawrence Park South': 81204.58,
'Leaside-Bennington': 79745.33,
'Little Portugal': 63858.29,
'Long Branch': 61572.92,
'Malvern': 63783.58,
'Maple Leaf': 64082.28,
'Markland Wood': 74740.98,
'Milliken': 64242.82,
'Mimico (includes Humber Bay Shores)': 63773.89,
'Morningside': 61257.47,
'Moss Park': 54874.4,
'Mount Dennis': 55231.44,
'Mount Olive-Silverstone-Jamestown': 57949.14,
'Mount Pleasant East': 73075.12,
'Mount Pleasant West': 60586.64,
'New Toronto': 55824.07,
'Newtonbrook East': 60137.3,
'Newtonbrook West': 57446.93,
```

```
'Niagara': 70100.51,
'North Riverdale': 71775.4,
'North St. James Town': 46486.95,
'Oakridge': 46394.14,
'Oakwood Village': 60173.95,
"O'Connor-Parkview": 61003.34,
'Old East York': 68843.14,
'Palmerston-Little Italy': 65441.87,
'Parkwoods-Donalda': 63747.65,
'Pelmo Park-Humberlea': 71184.39,
'Playter Estates-Danforth': 66084.38,
'Pleasant View': 66392.92,
'Princess-Rosethorn': 84366.06,
'Regent Park': 48756.31,
'Rexdale-Kipling': 62328.15,
'Rockcliffe-Smythe': 57530.44,
'Roncesvalles': 61225.6,
'Rosedale-Moore Park': 76377.12,
'Rouge': 77508.41,
'Runnymede-Bloor West Village': 79447.37,
'Rustic': 52506.84,
'Scarborough Village': 54521.5,
'South Parkdale': 47883.52,
'South Riverdale': 66958.82,
'St.Andrew-Windfields': 75214.68,
'Steeles': 63872.52,
'Stonegate-Queensway': 71302.41,
"Tam O'Shanter-Sullivan": 60075.64,
'Taylor-Massey': 51297.69,
'The Beaches': 74619.25,
'Thistletown-Beaumond Heights': 64890.32,
'Thorncliffe Park': 51365.07,
'Trinity-Bellwoods': 66623.82,
'University': 58000.74,
'Victoria Village': 55650.78,
'Waterfront Communities-The Island': 68722.88,
'West Hill': 58003.75,
'West Humber-Clairville': 67019.7,
'Westminster-Branson': 56768.44,
'Weston': 52083.88,
'Weston-Pelham Park': 59748.52,
'Wexford/Maryvale': 62770.91,
'Willowdale East': 59734.63,
'Willowdale West': 59240.07,
'Willowridge-Martingrove-Richview': 67736.21,
'Woburn': 58190.57,
'Woodbine Corridor': 66700.27,
```

```
'Woodbine-Lumsden': 67184.78,
'Wychwood': 63372.24,
'Yonge-Eglinton': 69617.08,
'Yonge-St.Clair': 69789.01,
'York University Heights': 56878.07,
'Yorkdale-Glen Park': 63622.54}}
```

[49]:

```
COVIDdf['NeighbourhoodAvgIncome'] = 0

#Creating new column in COVIDdf to add neighbourhood average income
```

[50]:

```
COVIDdf['NeighbourhoodAvgIncome'] = COVIDdf['Neighbourhood Name'].
↪map(Neighbourhoodlist['FinalAvg'])

#Using Neighbourhoodlist to average income, by matching with neighbourhood names
```

[51]:

```
OutcomeNames = COVIDdf[COVIDdf['Outcome'] == 'ACTIVE'].index

COVIDdf.drop(OutcomeNames, inplace = True)
COVIDdf

#Data prep for analysis
#Dropping 'ACTIVE' in COVIDdf.Outcomes because not needed for analysis
```

[51]:

```
              Age Group     Neighbourhood Name Client Gender   Outcome  \
0        50 to 59 Years        Willowdale East        FEMALE  RESOLVED
1        50 to 59 Years        Willowdale East          MALE  RESOLVED
2        20 to 29 Years      Parkwoods-Donalda        FEMALE  RESOLVED
3        60 to 69 Years   Church-Yonge Corridor       FEMALE  RESOLVED
4        60 to 69 Years   Church-Yonge Corridor         MALE  RESOLVED
…                   …                      …           …         …
111484   19 and younger   Kensington-Chinatown         MALE  RESOLVED
111552   19 and younger             L'Amoreaux         MALE  RESOLVED
111642   30 to 39 Years             Cliffcrest        FEMALE  RESOLVED
112005   19 and younger                 Weston        FEMALE  RESOLVED
112009   19 and younger    Lawrence Park North       UNKNOWN  RESOLVED

        NeighbourhoodAvgIncome
0                     59734.63
1                     59734.63
2                     63747.65
3                     56852.79
4                     56852.79
…                          …
111484                50219.07
111552                60667.94
111642                67150.13
```

```
112005                  52083.88
112009                  81940.85

[103105 rows x 5 columns]
```

[52]:
```
OutcomeRandomization = COVIDdf

#Creating identical COVIDdf dataframe to be used for randomized dropping of␣
↪values
```

[53]:
```
OutcomeRandomization
```

[53]:
```
             Age Group     Neighbourhood Name Client Gender   Outcome  \
0       50 to 59 Years        Willowdale East        FEMALE  RESOLVED
1       50 to 59 Years        Willowdale East          MALE  RESOLVED
2       20 to 29 Years      Parkwoods-Donalda        FEMALE  RESOLVED
3       60 to 69 Years  Church-Yonge Corridor        FEMALE  RESOLVED
4       60 to 69 Years  Church-Yonge Corridor          MALE  RESOLVED
…                  …                    …           …         …
111484  19 and younger   Kensington-Chinatown          MALE  RESOLVED
111552  19 and younger             L'Amoreaux          MALE  RESOLVED
111642  30 to 39 Years             Cliffcrest        FEMALE  RESOLVED
112005  19 and younger                 Weston        FEMALE  RESOLVED
112009  19 and younger    Lawrence Park North       UNKNOWN  RESOLVED

        NeighbourhoodAvgIncome
0                     59734.63
1                     59734.63
2                     63747.65
3                     56852.79
4                     56852.79
…                          …
111484                50219.07
111552                60667.94
111642                67150.13
112005                52083.88
112009                81940.85

[103105 rows x 5 columns]
```

[54]:
```
COVIDdf = COVIDdf.loc[COVIDdf['Outcome'] == 'FATAL']
COVIDdf

#Choosing to only keep 'Outcome' values with only 'FATAL' in COVIDdf
```

[54]:
```
             Age Group       Neighbourhood Name Client Gender Outcome  \
76      70 to 79 Years         Victoria Village          MALE   FATAL
```

```
263      60 to 69 Years                 Niagara          MALE    FATAL
266        90 and older             Morningside          MALE    FATAL
274        90 and older        O'Connor-Parkview         MALE    FATAL
290      70 to 79 Years       Don Valley Village         MALE    FATAL
…                     …                        …          …       …
105961   70 to 79 Years           Flemingdon Park      FEMALE    FATAL
107283   60 to 69 Years  York University Heights       FEMALE    FATAL
108207   70 to 79 Years   Glenfield-Jane Heights       FEMALE    FATAL
109017   50 to 59 Years        O'Connor-Parkview       FEMALE    FATAL
109734   80 to 89 Years        Rockcliffe-Smythe       FEMALE    FATAL

        NeighbourhoodAvgIncome
76                    55650.78
263                   70100.51
266                   61257.47
274                   61003.34
290                   62859.47
…                            …
105961                53544.73
107283                56878.07
108207                56592.74
109017                61003.34
109734                57530.44

[2759 rows x 5 columns]
```

[55]:
```
OutcomeRandomization = OutcomeRandomization.loc[OutcomeRandomization['Outcome']
 == 'RESOLVED']
OutcomeRandomization

#Choosing to only keep 'Outcome' values with only 'RESOLVED' in
 OutcomeRandomization
```

[55]:
```
             Age Group     Neighbourhood Name Client Gender   Outcome  \
0       50 to 59 Years         Willowdale East        FEMALE  RESOLVED
1       50 to 59 Years         Willowdale East          MALE  RESOLVED
2       20 to 29 Years       Parkwoods-Donalda        FEMALE  RESOLVED
3       60 to 69 Years   Church-Yonge Corridor        FEMALE  RESOLVED
4       60 to 69 Years   Church-Yonge Corridor          MALE  RESOLVED
…                    …                       …           …         …
111484  19 and younger    Kensington-Chinatown          MALE  RESOLVED
111552  19 and younger              L'Amoreaux          MALE  RESOLVED
111642  30 to 39 Years              Cliffcrest        FEMALE  RESOLVED
112005  19 and younger                  Weston        FEMALE  RESOLVED
112009  19 and younger      Lawrence Park North      UNKNOWN  RESOLVED

        NeighbourhoodAvgIncome
```

```
0                      59734.63
1                      59734.63
2                      63747.65
3                      56852.79
4                      56852.79
…                         …
111484                 50219.07
111552                 60667.94
111642                 67150.13
112005                 52083.88
112009                 81940.85

[100346 rows x 5 columns]
```

[56]:

```python
np.random.seed(10)
Nremove = 97587

drop_indices = np.random.choice(OutcomeRandomization.index, Nremove,␣
 ↪replace=False)
OutcomeRandomization = OutcomeRandomization.drop(drop_indices)
OutcomeRandomization

#Need to randomly select to keep 2759/100346 rows in OutcomeRandomization
#Need to have same number of 'RESOLVED' as 'FATAL' for analysis
```

[56]:

```
              Age Group                     Neighbourhood Name Client Gender  \
51       50 to 59 Years                    Leaside-Bennington          MALE
88       30 to 39 Years                           The Beaches          MALE
139      30 to 39 Years                            University        FEMALE
258      30 to 39 Years                           Blake-Jones        FEMALE
289      20 to 29 Years                     Woodbine Corridor          MALE
…                  …                                   …           …
107850   30 to 39 Years                                Rustic        FEMALE
107871   19 and younger                          Mount Dennis        FEMALE
108017   19 and younger             Islington-City Centre West        FEMALE
108078   19 and younger   Mimico (includes Humber Bay Shores)        FEMALE
108222   60 to 69 Years                   Downsview-Roding-CFB        FEMALE

          Outcome  NeighbourhoodAvgIncome
51       RESOLVED                79745.33
88       RESOLVED                74619.25
139      RESOLVED                58000.74
258      RESOLVED                62588.57
289      RESOLVED                66700.27
…           …                     …
107850   RESOLVED                52506.84
107871   RESOLVED                55231.44
```

```
108017   RESOLVED                 67355.18
108078   RESOLVED                 63773.89
108222   RESOLVED                 60070.48

[2759 rows x 5 columns]
```

[57]:
```python
frames = [COVIDdf, OutcomeRandomization]
COVIDdf = pd.concat(frames)

#Adding OutcomeRandomization into COVIDdf
```

[58]:
```python
COVIDdf['Outcome'] = COVIDdf['Outcome'].replace(['FATAL', 'RESOLVED'], [0,1])
COVIDdf

#Changing 'Outcome' values from 'FATAL','RESOLVED', into '0','1'
```

[58]:
```
              Age Group                   Neighbourhood Name Client Gender  \
76      70 to 79 Years                      Victoria Village          MALE
263     60 to 69 Years                               Niagara          MALE
266       90 and older                           Morningside          MALE
274       90 and older                      O'Connor-Parkview          MALE
290     70 to 79 Years                    Don Valley Village          MALE
…                  …                                     …           …
107850  30 to 39 Years                                Rustic        FEMALE
107871  19 and younger                          Mount Dennis        FEMALE
108017  19 and younger           Islington-City Centre West        FEMALE
108078  19 and younger  Mimico (includes Humber Bay Shores)        FEMALE
108222  60 to 69 Years                   Downsview-Roding-CFB        FEMALE

        Outcome  NeighbourhoodAvgIncome
76            0                55650.78
263           0                70100.51
266           0                61257.47
274           0                61003.34
290           0                62859.47
…           …                     …
107850        1                52506.84
107871        1                55231.44
108017        1                67355.18
108078        1                63773.89
108222        1                60070.48

[5518 rows x 5 columns]
```

[59]:
```python
COVIDdf
```

[59]:
```
                Age Group                      Neighbourhood Name Client Gender  \
76      70 to 79 Years                       Victoria Village          MALE
263     60 to 69 Years                                Niagara          MALE
266       90 and older                            Morningside          MALE
274       90 and older                       O'Connor-Parkview          MALE
290     70 to 79 Years                     Don Valley Village          MALE
…                  …                                      …           …
107850  30 to 39 Years                                 Rustic        FEMALE
107871  19 and younger                          Mount Dennis        FEMALE
108017  19 and younger            Islington-City Centre West        FEMALE
108078  19 and younger  Mimico (includes Humber Bay Shores)        FEMALE
108222  60 to 69 Years                   Downsview-Roding-CFB        FEMALE

        Outcome  NeighbourhoodAvgIncome
76            0                55650.78
263           0                70100.51
266           0                61257.47
274           0                61003.34
290           0                62859.47
…           …                   …
107850        1                52506.84
107871        1                55231.44
108017        1                67355.18
108078        1                63773.89
108222        1                60070.48

[5518 rows x 5 columns]
```

[60]:
```python
COVIDdf = COVIDdf.fillna(0)

#Filling NA values with value of 0
```

[61]:
```python
print(COVIDdf['NeighbourhoodAvgIncome'])
```

```
76        55650.78
263       70100.51
266       61257.47
274       61003.34
290       62859.47
            …
107850    52506.84
107871    55231.44
108017    67355.18
108078    63773.89
108222    60070.48
Name: NeighbourhoodAvgIncome, Length: 5518, dtype: float64
```

[62]:
```
COVIDdf.describe()

#Descriptive stats for COVIDdf
```

[62]:
```
           Outcome  NeighbourhoodAvgIncome
count  5518.000000             5518.000000
mean      0.500000            62015.445239
std       0.500045             8133.764355
min       0.000000                0.000000
25%       0.000000            57114.900000
50%       0.500000            61861.540000
75%       1.000000            66392.920000
max       1.000000            86982.230000
```

[63]:
```
COVIDdf = pd.get_dummies(COVIDdf)

#One-hot encoding the categorical variables ("Age Group" and "Client Gender")
```

[64]:
```
COVIDdf
```

[64]:
```
        Outcome  NeighbourhoodAvgIncome  Age Group_19 and younger  \
76            0                55650.78                         0
263           0                70100.51                         0
266           0                61257.47                         0
274           0                61003.34                         0
290           0                62859.47                         0
…           …                     …                       …
107850        1                52506.84                         0
107871        1                55231.44                         1
108017        1                67355.18                         1
108078        1                63773.89                         1
108222        1                60070.48                         0

        Age Group_20 to 29 Years  Age Group_30 to 39 Years  \
76                             0                         0
263                            0                         0
266                            0                         0
274                            0                         0
290                            0                         0
…                            …                         …
107850                         0                         1
107871                         0                         0
108017                         0                         0
108078                         0                         0
108222                         0                         0

        Age Group_40 to 49 Years  Age Group_50 to 59 Years  \
```

76	0	0
263	0	0
266	0	0
274	0	0
290	0	0
…	…	…
107850	0	0
107871	0	0
108017	0	0
108078	0	0
108222	0	0

	Age Group_60 to 69 Years	Age Group_70 to 79 Years	\
76	0	1	
263	1	0	
266	0	0	
274	0	0	
290	0	1	
…	…	…	
107850	0	0	
107871	0	0	
108017	0	0	
108078	0	0	
108222	1	0	

	Age Group_80 to 89 Years	…	Neighbourhood Name_Woodbine Corridor	\
76	0	…	0	
263	0	…	0	
266	0	…	0	
274	0	…	0	
290	0	…	0	
…	…	…	…	
107850	0	…	0	
107871	0	…	0	
108017	0	…	0	
108078	0	…	0	
108222	0	…	0	

	Neighbourhood Name_Woodbine-Lumsden	Neighbourhood Name_Wychwood	\
76	0	0	
263	0	0	
266	0	0	
274	0	0	
290	0	0	
…	…	…	
107850	0	0	
107871	0	0	

```
108017                                 0                            0
108078                                 0                            0
108222                                 0                            0

        Neighbourhood Name_Yonge-Eglinton  Neighbourhood Name_Yonge-St.Clair  \
76                                      0                                  0
263                                     0                                  0
266                                     0                                  0
274                                     0                                  0
290                                     0                                  0
…                                     …                                …
107850                                  0                                  0
107871                                  0                                  0
108017                                  0                                  0
108078                                  0                                  0
108222                                  0                                  0

        Neighbourhood Name_York University Heights  \
76                                               0
263                                              0
266                                              0
274                                              0
290                                              0
…                                              …
107850                                           0
107871                                           0
108017                                           0
108078                                           0
108222                                           0

        Neighbourhood Name_Yorkdale-Glen Park  Client Gender_FEMALE  \
76                                          0                     0
263                                         0                     0
266                                         0                     0
274                                         0                     0
290                                         0                     0
…                                         …                  …
107850                                      0                     1
107871                                      0                     1
108017                                      0                     1
108078                                      0                     1
108222                                      0                     1

        Client Gender_MALE  Client Gender_UNKNOWN
76                       1                      0
263                      1                      0
266                      1                      0
```

```
274                       1                     0
290                       1                     0
…                     …                     …
107850                    0                     0
107871                    0                     0
108017                    0                     0
108078                    0                     0
108222                    0                     0

[5518 rows x 154 columns]
```

[65]:
```
COVIDdf.groupby('Outcome').mean()

#Breaking down and analysing dataset by 'Outcome'
#Data exploration
```

[65]:
```
         NeighbourhoodAvgIncome  Age Group_19 and younger  \
Outcome
0                  62276.856173                  0.000362
1                  61754.034306                  0.147155

         Age Group_20 to 29 Years  Age Group_30 to 39 Years  \
Outcome
0                        0.001087                  0.002537
1                        0.206234                  0.174339

         Age Group_40 to 49 Years  Age Group_50 to 59 Years  \
Outcome
0                        0.010149                  0.035158
1                        0.151142                  0.136644

         Age Group_60 to 69 Years  Age Group_70 to 79 Years  \
Outcome
0                        0.094962                  0.186662
1                        0.090250                  0.044944

         Age Group_80 to 89 Years  Age Group_90 and older  …  \
Outcome                                                    …
0                        0.358101                0.310982  …
1                        0.031171                0.018123  …

         Neighbourhood Name_Woodbine Corridor  \
Outcome
0                                    0.001450
1                                    0.001812

         Neighbourhood Name_Woodbine-Lumsden  Neighbourhood Name_Wychwood  \
```

```
Outcome
0                                    0.000362                     0.007249
1                                    0.000000                     0.002175

         Neighbourhood Name_Yonge-Eglinton  Neighbourhood Name_Yonge-St.Clair  \
Outcome
0                                  0.000362                           0.003262
1                                  0.000362                           0.001450

         Neighbourhood Name_York University Heights  \
Outcome
0                                           0.027546
1                                           0.020660

         Neighbourhood Name_Yorkdale-Glen Park  Client Gender_FEMALE  \
Outcome
0                                      0.018847              0.488583
1                                      0.006524              0.491482

         Client Gender_MALE  Client Gender_UNKNOWN
Outcome
0                  0.500181               0.011236
1                  0.503806               0.004712

[2 rows x 153 columns]
```

[66]:
```
LogCOVIDdf = COVIDdf
```

[67]:
```
for col in COVIDdf.columns:
    print(col)
```

```
Outcome
NeighbourhoodAvgIncome
Age Group_19 and younger
Age Group_20 to 29 Years
Age Group_30 to 39 Years
Age Group_40 to 49 Years
Age Group_50 to 59 Years
Age Group_60 to 69 Years
Age Group_70 to 79 Years
Age Group_80 to 89 Years
Age Group_90 and older
Neighbourhood Name_Agincourt North
Neighbourhood Name_Agincourt South-Malvern West
Neighbourhood Name_Alderwood
Neighbourhood Name_Annex
Neighbourhood Name_Banbury-Don Mills
```

```
Neighbourhood Name_Bathurst Manor
Neighbourhood Name_Bay Street Corridor
Neighbourhood Name_Bayview Village
Neighbourhood Name_Bayview Woods-Steeles
Neighbourhood Name_Bedford Park-Nortown
Neighbourhood Name_Beechborough-Greenbrook
Neighbourhood Name_Bendale
Neighbourhood Name_Birchcliffe-Cliffside
Neighbourhood Name_Black Creek
Neighbourhood Name_Blake-Jones
Neighbourhood Name_Briar Hill-Belgravia
Neighbourhood Name_Bridle Path-Sunnybrook-York Mills
Neighbourhood Name_Broadview North
Neighbourhood Name_Brookhaven-Amesbury
Neighbourhood Name_Cabbagetown-South St. James Town
Neighbourhood Name_Caledonia-Fairbank
Neighbourhood Name_Casa Loma
Neighbourhood Name_Centennial Scarborough
Neighbourhood Name_Church-Yonge Corridor
Neighbourhood Name_Clairlea-Birchmount
Neighbourhood Name_Clanton Park
Neighbourhood Name_Cliffcrest
Neighbourhood Name_Corso Italia-Davenport
Neighbourhood Name_Danforth
Neighbourhood Name_Danforth East York
Neighbourhood Name_Don Valley Village
Neighbourhood Name_Dorset Park
Neighbourhood Name_Dovercourt-Wallace Emerson-Junction
Neighbourhood Name_Downsview-Roding-CFB
Neighbourhood Name_Dufferin Grove
Neighbourhood Name_East End-Danforth
Neighbourhood Name_Edenbridge-Humber Valley
Neighbourhood Name_Eglinton East
Neighbourhood Name_Elms-Old Rexdale
Neighbourhood Name_Englemount-Lawrence
Neighbourhood Name_Eringate-Centennial-West Deane
Neighbourhood Name_Etobicoke West Mall
Neighbourhood Name_Flemingdon Park
Neighbourhood Name_Forest Hill North
Neighbourhood Name_Forest Hill South
Neighbourhood Name_Glenfield-Jane Heights
Neighbourhood Name_Greenwood-Coxwell
Neighbourhood Name_Guildwood
Neighbourhood Name_Henry Farm
Neighbourhood Name_High Park North
Neighbourhood Name_High Park-Swansea
Neighbourhood Name_Highland Creek
Neighbourhood Name_Hillcrest Village
```

```
Neighbourhood Name_Humber Heights-Westmount
Neighbourhood Name_Humber Summit
Neighbourhood Name_Humbermede
Neighbourhood Name_Humewood-Cedarvale
Neighbourhood Name_Ionview
Neighbourhood Name_Islington-City Centre West
Neighbourhood Name_Junction Area
Neighbourhood Name_Keelesdale-Eglinton West
Neighbourhood Name_Kennedy Park
Neighbourhood Name_Kensington-Chinatown
Neighbourhood Name_Kingsview Village-The Westway
Neighbourhood Name_Kingsway South
Neighbourhood Name_L'Amoreaux
Neighbourhood Name_Lambton Baby Point
Neighbourhood Name_Lansing-Westgate
Neighbourhood Name_Lawrence Park North
Neighbourhood Name_Lawrence Park South
Neighbourhood Name_Leaside-Bennington
Neighbourhood Name_Little Portugal
Neighbourhood Name_Long Branch
Neighbourhood Name_Malvern
Neighbourhood Name_Maple Leaf
Neighbourhood Name_Markland Wood
Neighbourhood Name_Milliken
Neighbourhood Name_Mimico (includes Humber Bay Shores)
Neighbourhood Name_Morningside
Neighbourhood Name_Moss Park
Neighbourhood Name_Mount Dennis
Neighbourhood Name_Mount Olive-Silverstone-Jamestown
Neighbourhood Name_Mount Pleasant East
Neighbourhood Name_Mount Pleasant West
Neighbourhood Name_New Toronto
Neighbourhood Name_Newtonbrook East
Neighbourhood Name_Newtonbrook West
Neighbourhood Name_Niagara
Neighbourhood Name_North Riverdale
Neighbourhood Name_North St. James Town
Neighbourhood Name_O'Connor-Parkview
Neighbourhood Name_Oakridge
Neighbourhood Name_Oakwood Village
Neighbourhood Name_Old East York
Neighbourhood Name_Palmerston-Little Italy
Neighbourhood Name_Parkwoods-Donalda
Neighbourhood Name_Pelmo Park-Humberlea
Neighbourhood Name_Playter Estates-Danforth
Neighbourhood Name_Pleasant View
Neighbourhood Name_Princess-Rosethorn
Neighbourhood Name_Regent Park
```

```
Neighbourhood Name_Rexdale-Kipling
Neighbourhood Name_Rockcliffe-Smythe
Neighbourhood Name_Roncesvalles
Neighbourhood Name_Rosedale-Moore Park
Neighbourhood Name_Rouge
Neighbourhood Name_Runnymede-Bloor West Village
Neighbourhood Name_Rustic
Neighbourhood Name_Scarborough Village
Neighbourhood Name_South Parkdale
Neighbourhood Name_South Riverdale
Neighbourhood Name_St.Andrew-Windfields
Neighbourhood Name_Steeles
Neighbourhood Name_Stonegate-Queensway
Neighbourhood Name_Tam O'Shanter-Sullivan
Neighbourhood Name_Taylor-Massey
Neighbourhood Name_The Beaches
Neighbourhood Name_Thistletown-Beaumond Heights
Neighbourhood Name_Thorncliffe Park
Neighbourhood Name_Trinity-Bellwoods
Neighbourhood Name_University
Neighbourhood Name_Victoria Village
Neighbourhood Name_Waterfront Communities-The Island
Neighbourhood Name_West Hill
Neighbourhood Name_West Humber-Clairville
Neighbourhood Name_Westminster-Branson
Neighbourhood Name_Weston
Neighbourhood Name_Weston-Pellam Park
Neighbourhood Name_Wexford/Maryvale
Neighbourhood Name_Willowdale East
Neighbourhood Name_Willowdale West
Neighbourhood Name_Willowridge-Martingrove-Richview
Neighbourhood Name_Woburn
Neighbourhood Name_Woodbine Corridor
Neighbourhood Name_Woodbine-Lumsden
Neighbourhood Name_Wychwood
Neighbourhood Name_Yonge-Eglinton
Neighbourhood Name_Yonge-St.Clair
Neighbourhood Name_York University Heights
Neighbourhood Name_Yorkdale-Glen Park
Client Gender_FEMALE
Client Gender_MALE
Client Gender_UNKNOWN
```

[68]:
```
#LOGISTIC REGRESSION
```

[69]:
```
LogCOVIDdf = COVIDdf
```

```
#Creating new dataframe for logistic regression
```

[70]:
```
X = LogCOVIDdf.loc[:, LogCOVIDdf.columns != 'Outcome']
y = LogCOVIDdf.loc[:, LogCOVIDdf.columns == 'Outcome']
y = y.astype('int')

os = SMOTE(random_state=0)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=0)
columns = X_train.columns


os_data_X,os_data_y=os.fit_resample(X_train, y_train)
os_data_X = pd.DataFrame(data=os_data_X,columns=columns )
os_data_y= pd.DataFrame(data=os_data_y,columns=['Outcome'])

print("Length of oversampled data is ",len(os_data_X))
print("Number of no subscription in oversampled␣
↪data",len(os_data_y[os_data_y['Outcome']==0]))
print("Number of subscription",len(os_data_y[os_data_y['Outcome']==1]))
print("Proportion of no subscription data in oversampled data is␣
↪",len(os_data_y[os_data_y['Outcome']==0])/len(os_data_X))
print("Proportion of subscription data in oversampled data is␣
↪",len(os_data_y[os_data_y['Outcome']==1])/len(os_data_X))

#Implementing SMOTE
#Creating perfectly balanced data
```

```
Length of oversampled data is  3864
Number of no subscription in oversampled data 1932
Number of subscription 1932
Proportion of no subscription data in oversampled data is  0.5
Proportion of subscription data in oversampled data is  0.5
```

[71]:
```
Cdf_final_vars = LogCOVIDdf.columns.values.tolist()
y = ['Outcome']
X =[i for i in Cdf_final_vars if i not in y ]
logreg = LogisticRegression()
rfe = RFE(logreg, 20)
rfe = rfe.fit(os_data_X, os_data_y.values.ravel())
print(rfe.support_)
print(rfe.ranking_)

#RecursiveFeatureElimination
#Help select best and worst features
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:72:
```

```
FutureWarning: Pass n_features_to_select=20 as keyword args. From version 1.0
(renaming of 0.25) passing these as positional arguments will result in an error
  "will result in an error", FutureWarning)
```

```
[False  True  True  True  True False  True  True  True  True False False
 False False False False False False False False False False  True False
 False False False False False False False  True False False False False
 False False False False False False False  True False False False False
 False  True False False False  True False False False False False False
 False False False False False False False False False False False False
 False False False False False False  True False False False False False
  True False False False  True False False False False False False False
 False False False False False False False  True False False False False
 False False False False False False False False False False False  True
 False False False False False False False False False  True False False
 False False False False False False False False False False False False
 False False False False False  True False False False]
[134   1   1   1   1  52   1   1   1   1 110  14  16 130  64  92  61  50
 108   9 107 100   1 114  46 120 118 116  60  88  79   1  58  78  57  21
  43  33  22  90  20  81 101   1  48  13  42  82  73   1  87  44  85   1
 119  32  71   4 132   6  34 125  47   7   3  84  49  93 111  23  65  19
  54  75 122 129 131  89   1  17 123  76  98  91   1  68  25  62   1  72
  95  35 126  12  66  36  28 105 112  45   8  56  26   1  40  41  80  29
  70   5  11  74  96  37  39 133  63  53 121   1  86  67 104  18 106 109
  38  94  55   1  59  24  27 117  77 103  51 102  31 124  99   2 128  97
  83  10 115  69 127   1  15 113  30]
```

[72]:
```python
cols = ['Age Group_19 and younger', 'Age Group_20 to 29 Years', 'Age Group_30␣
↪to 39 Years', 'Age Group_40 to 49 Years',
        'Age Group_60 to 69 Years', 'Age Group_70 to 79 Years', 'Age Group_80␣
↪to 89 Years', 'Age Group_90 and older',
        'Neighbourhood Name_Birchcliffe-Cliffside', 'Neighbourhood Name_Casa␣
↪Loma','Neighbourhood Name_Downsview-Roding-CFB',
        'Neighbourhood Name_Englemount-Lawrence', 'Neighbourhood Name_Forest␣
↪Hill North', 'Neighbourhood Name_Lawrence Park North',
        'Neighbourhood Name_Maple Leaf', 'Neighbourhood Name_Morningside',␣
↪'Neighbourhood Name_Old East York',
        'Neighbourhood Name_South Parkdale', 'Neighbourhood␣
↪Name_Trinity-Bellwoods', 'Neighbourhood Name_Yorkdale-Glen Park']

X=os_data_X[cols]
y=os_data_y['Outcome']

import statsmodels.api as sm
logit_model=sm.Logit(y,X)

result=logit_model.fit()
```

```
print(result.summary2())

#Selecting best features
#Implementing the model
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.283759
         Iterations: 35
                                 Results: Logit
================================================================================
==================
Model:                         Logit                      Pseudo R-squared:
0.591
Dependent Variable:            Outcome                    AIC:
2232.8857
Date:                          2021-04-05 15:04           BIC:
2358.0749
No. Observations:              3864                       Log-Likelihood:
-1096.4
Df Model:                      19                         LL-Null:
-2678.3
Df Residuals:                  3844                       LLR p-value:
0.0000
Converged:                     0.0000                     Scale:
1.0000
No. Iterations:                35.0000
--------------------------------------------------------------------------------
------------------
                                         Coef.   Std.Err.   z    P>|z|
[0.025    0.975]
--------------------------------------------------------------------------------
------------------
Age Group_19 and younger                  5.6577   1.0026   5.6428 0.0000
3.6925    7.6228
Age Group_20 to 29 Years                  6.0865   1.0056   6.0526 0.0000
4.1155    8.0575
Age Group_30 to 39 Years                  4.5073   0.5082   8.8687 0.0000
3.5112    5.5034
Age Group_40 to 49 Years                  2.8678   0.2588  11.0830 0.0000
2.3607    3.3750
Age Group_60 to 69 Years                 -0.0966   0.1077  -0.8973 0.3696
-0.3076    0.1144
Age Group_70 to 79 Years                 -1.3689   0.1192 -11.4824 0.0000
-1.6025   -1.1352
Age Group_80 to 89 Years                 -2.4543   0.1386 -17.7078 0.0000
-2.7260   -2.1827
Age Group_90 and older                   -3.2613   0.2135 -15.2740 0.0000
```

```
-3.6798    -2.8428
Neighbourhood Name_Birchcliffe-Cliffside  -0.9006    0.4918  -1.8312 0.0671
-1.8644     0.0633
Neighbourhood Name_Casa Loma               1.4757    0.6812   2.1662 0.0303
0.1405     2.8109
Neighbourhood Name_Downsview-Roding-CFB    1.4959    0.4207   3.5560 0.0004
0.6714     2.3204
Neighbourhood Name_Englemount-Lawrence     1.3519    0.5335   2.5340 0.0113
0.3062     2.3976
Neighbourhood Name_Forest Hill North       2.8481    0.8731   3.2620 0.0011
1.1368     4.5593
Neighbourhood Name_Lawrence Park North    16.4367  697.5992   0.0236 0.9812
-1350.8325  1383.7059
Neighbourhood Name_Maple Leaf             -1.0256    0.6674  -1.5368 0.1244
-2.3337     0.2824
Neighbourhood Name_Morningside            -1.0358    0.5793  -1.7879 0.0738
-2.1713     0.0997
Neighbourhood Name_Old East York         -20.4627 6569.4532  -0.0031 0.9975
-12896.3543 12855.4289
Neighbourhood Name_South Parkdale         -0.7540    0.4681  -1.6110 0.1072
-1.6714     0.1634
Neighbourhood Name_Trinity-Bellwoods       2.7186    1.2807   2.1227 0.0338
0.2084     5.2288
Neighbourhood Name_Yorkdale-Glen Park     -1.8988    0.9292  -2.0433 0.0410
-3.7201    -0.0775
================================================================================
==================


/opt/conda/lib/python3.7/site-packages/statsmodels/base/model.py:568:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

[73]:

```
cols = ['Age Group_19 and younger', 'Age Group_20 to 29 Years', 'Age Group_30␣
 ↪to 39 Years', 'Age Group_40 to 49 Years',
        'Age Group_70 to 79 Years', 'Age Group_80 to 89 Years', 'Age Group_90␣
 ↪and older',
        'Neighbourhood Name_Casa Loma','Neighbourhood␣
 ↪Name_Downsview-Roding-CFB','Neighbourhood Name_Englemount-Lawrence',
        'Neighbourhood Name_Forest Hill North','Neighbourhood␣
 ↪Name_Trinity-Bellwoods', 'Neighbourhood Name_Yorkdale-Glen Park']

X=os_data_X[cols]
y=os_data_y['Outcome']

import statsmodels.api as sm
```

```
logit_model=sm.Logit(y,X)

result=logit_model.fit()
print(result.summary2())

#Remove features with p-value higher than 0.05
#Implementing model with fewer features
```

```
Optimization terminated successfully.
         Current function value: 0.287772
         Iterations 10
                                    Results: Logit
================================================================================
========
Model:                      Logit                    Pseudo R-squared:
0.585
Dependent Variable:         Outcome                  AIC:
2249.9023
Date:                       2021-04-05 15:04         BIC:
2331.2752
No. Observations:           3864                     Log-Likelihood:
-1112.0
Df Model:                   12                       LL-Null:
-2678.3
Df Residuals:               3851                     LLR p-value:
0.0000
Converged:                  1.0000                   Scale:
1.0000
No. Iterations:             10.0000
--------------------------------------------------------------------------------
--------
                                         Coef.  Std.Err.    z     P>|z|   [0.025
0.975]
--------------------------------------------------------------------------------
--------
Age Group_19 and younger                 5.6195   1.0022   5.6069 0.0000  3.6551
7.5839
Age Group_20 to 29 Years                 6.0367   1.0052   6.0053 0.0000  4.0665
8.0068
Age Group_30 to 39 Years                 4.4588   0.5075   8.7851 0.0000  3.4641
5.4536
Age Group_40 to 49 Years                 2.7850   0.2507  11.1109 0.0000  2.2938
3.2763
Age Group_70 to 79 Years                -1.4112   0.1185 -11.9082 0.0000 -1.6435
-1.1790
Age Group_80 to 89 Years                -2.5049   0.1380 -18.1539 0.0000 -2.7753
-2.2345
```

```
Age Group_90 and older                  -3.2702   0.2089 -15.6546 0.0000 -3.6796
-2.8608
Neighbourhood Name_Casa Loma             1.4879   0.6856   2.1702 0.0300  0.1441
2.8317
Neighbourhood Name_Downsview-Roding-CFB  1.4893   0.4231   3.5201 0.0004  0.6601
2.3186
Neighbourhood Name_Englemount-Lawrence   1.3792   0.5340   2.5827 0.0098  0.3326
2.4259
Neighbourhood Name_Forest Hill North     2.8683   0.8706   3.2945 0.0010  1.1619
4.5747
Neighbourhood Name_Trinity-Bellwoods     2.7500   1.2881   2.1350 0.0328  0.2254
5.2746
Neighbourhood Name_Yorkdale-Glen Park   -1.8671   0.9207  -2.0280 0.0426 -3.6716
-0.0627
================================================================================
========
```

[74]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

[74]:
```
LogisticRegression()
```

[75]:
```
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.
 ↪format(logreg.score(X_test, y_test)))

#Predicting the test set results and calculating the accuracy
```

```
Accuracy of logistic regression classifier on test set: 0.87
```

[76]:
```
confusion_matrix = confusion_matrix(y_test, y_pred)
print(confusion_matrix)

#result is telling us that we have 488+89 correct predictions and 61+522␣
 ↪incorrect predictions.
```

```
[[488  89]
 [ 61 522]]
```

[77]:
```
print(classification_report(y_test, y_pred))

#Compute precision, recall, F-measure and support
```

```
              precision    recall  f1-score   support
```

	precision	recall	f1-score	support
0	0.89	0.85	0.87	577
1	0.85	0.90	0.87	583
accuracy			0.87	1160
macro avg	0.87	0.87	0.87	1160
weighted avg	0.87	0.87	0.87	1160

[78]:

```
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()

#ROC curve
```

[79]:
```
#RANDOM FOREST REGRESSION
```

[80]:
```
RfCOVIDdf = COVIDdf
```

[81]:
```
labels = np.array(RfCOVIDdf['Outcome'])
RfCOVIDdf = RfCOVIDdf.drop('Outcome', axis =1)
RfCOVIDdf_list = list(RfCOVIDdf.columns)
RfCOVIDdf = np.array(RfCOVIDdf)

#Creating labels as values to be predicted for model
#Removing labels from COVIDdf
#Storing COVIDdf names for later use
#Converting COVIDdf dataframe into numpy array
```

[82]:
```
train_COVIDdf, test_COVIDdf, train_labels, test_labels =␣
↪train_test_split(RfCOVIDdf, labels, test_size = 0.25, random_state = 42)

# Using Skicit-learn to split data into training and testing sets
# Split the data into training and testing sets
```

[83]:
```
print(train_labels)
```

```
[1 1 0 … 1 1 0]
```

[84]:
```
print('Training COVIDdf Shape:', train_COVIDdf.shape)
print('Training Labels Shape:', train_labels.shape)
print('Testing COVIDdfs Shape:', test_COVIDdf.shape)
print('Testing Labels Shape:', test_labels.shape)
```

```
Training COVIDdf Shape: (4138, 153)
Training Labels Shape: (4138,)
Testing COVIDdfs Shape: (1380, 153)
Testing Labels Shape: (1380,)
```

[85]:
```
rf = RandomForestRegressor(n_estimators = 1, random_state = 1)
rf.fit(train_COVIDdf, train_labels);

# Import the model we are using
# Instantiate model with 1 decision trees
# Train the model on training data
```

[86]:
```
predictions = rf.predict(test_COVIDdf)
errors = abs(predictions - test_labels)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

# Use the forest's predict method on the test data
# Calculate the absolute errors
```

```
# Print out the mean absolute error (mae)
```

```
Mean Absolute Error: 0.16 degrees.
```

[87]:
```
rf = RandomForestRegressor(n_estimators = 10, random_state = 10)
rf.fit(train_COVIDdf, train_labels);



predictions = rf.predict(test_COVIDdf)
errors = abs(predictions - test_labels)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

Accuracycount = 0

for i in range(0, len(predictions)):
    if predictions[i] == test_labels[i]:
          Accuracycount +=1

Accuracy = 100* Accuracycount/ len(predictions)
Accuracy

# Instantiate model with 10 decision trees
# Train the model on training data
# Use the forest's predict method on the test data
# Calculate the absolute errors
# Print out the mean absolute error (mae)
# Print out accuracy
```

```
Mean Absolute Error: 0.15 degrees.
```

[87]:
```
61.73913043478261
```

[88]:
```
rf = RandomForestRegressor(n_estimators = 15, random_state = 1)
rf.fit(train_COVIDdf, train_labels);

predictions = rf.predict(test_COVIDdf)
errors = abs(predictions - test_labels)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

Accuracycount = 0

for i in range(0, len(predictions)):
    if predictions[i] == test_labels[i]:
          Accuracycount +=1

Accuracy = 100* Accuracycount/ len(predictions)
Accuracy
```

```
# Instantiate model with 15 decision trees
# Train the model on training data
# Use the forest's predict method on the test data
# Calculate the absolute errors
# Print out the mean absolute error (mae)
# Print out accuracy
```

```
Mean Absolute Error: 0.15 degrees.
```

[88]: 60.43478260869565

[89]:
```
rf = RandomForestRegressor(n_estimators = 100, random_state = 1)
rf.fit(train_COVIDdf, train_labels);

predictions = rf.predict(test_COVIDdf)
errors = abs(predictions - test_labels)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

Accuracycount = 0

for i in range(0, len(predictions)):
    if predictions[i] == test_labels[i]:
          Accuracycount +=1

Accuracy = 100* Accuracycount/ len(predictions)
Accuracy

# Instantiate model with 100 decision trees
# Train the model on training data
# Use the forest's predict method on the test data
# Calculate the absolute errors
# Print out the mean absolute error (mae)
# Print out accuracy
```

```
Mean Absolute Error: 0.15 degrees.
```

[89]: 53.26086956521739

[90]:
```
rf = RandomForestRegressor(n_estimators = 150, random_state = 10)
rf.fit(train_COVIDdf, train_labels);

predictions = rf.predict(test_COVIDdf)
errors = abs(predictions - test_labels)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

Accuracycount = 0
```

```python
for i in range(0, len(predictions)):
    if predictions[i] == test_labels[i]:
          Accuracycount +=1

Accuracy = 100* Accuracycount/ len(predictions)
Accuracy

# Instantiate model with 150 decision trees
# Train the model on training data
# Use the forest's predict method on the test data
# Calculate the absolute errors
# Print out the mean absolute error (mae)
# Print out accuracy
```

```
Mean Absolute Error: 0.15 degrees.
```

[90]: 52.68115942028985

[91]:
```python
rf = RandomForestRegressor(n_estimators = 1000, random_state = 1)
rf.fit(train_COVIDdf, train_labels);

predictions = rf.predict(test_COVIDdf)
errors = abs(predictions - test_labels)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')

Accuracycount = 0

for i in range(0, len(predictions)):
    if predictions[i] == test_labels[i]:
          Accuracycount +=1

Accuracy = 100* Accuracycount/ len(predictions)
Accuracy

# Instantiate model with 1000 decision trees
# Train the model on training data
# Use the forest's predict method on the test data
# Calculate the absolute errors
# Print out the mean absolute error (mae)
# Print out accuracy
```

```
Mean Absolute Error: 0.15 degrees.
```

[91]: 46.666666666666664

[92]:
```
rf = RandomForestRegressor(n_estimators = 1, random_state = 1)
rf.fit(train_COVIDdf, train_labels);


predictions = rf.predict(test_COVIDdf)
errors = abs(predictions - test_labels)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
Accuracycount = 0

for i in range(0, len(predictions)):
    if predictions[i] == test_labels[i]:
          Accuracycount +=1

Accuracy = 100* Accuracycount/ len(predictions)
Accuracy

# Instantiate model with 1 decision trees
# Train the model on training data
# Use the forest's predict method on the test data
# Calculate the absolute errors
# Print out the mean absolute error (mae)
# Print out accuracy
# This model has highest accuracy out of all models
```

```
Mean Absolute Error: 0.16 degrees.
```

[92]:
```
73.55072463768116
```

[93]:
```
tree = rf.estimators_[0]
export_graphviz(tree, out_file = 'tree.dot', feature_names = RfCOVIDdf_list,␣
 ↪rounded = True, precision = 1)
(graph, ) = pydotplus.graph_from_dot_file('tree.dot')
graph.write_png('tree.png')

# Import tools needed for visualization
# Pull out one tree from the forest
# Export the image to a dot file
# Use dot file to create a graph
# Write graph to a png file
```

```
    ␣
↪---------------------------------------------------------------------------

      TypeError                                 Traceback (most recent call␣
↪last)
```

```
<ipython-input-93-d79fc8d0441e> in <module>
      1 tree = rf.estimators_[0]
      2 export_graphviz(tree, out_file = 'tree.dot', feature_names =␣
↪RfCOVIDdf_list, rounded = True, precision = 1)
----> 3 (graph, ) = pydotplus.graph_from_dot_file('tree.dot')
      4 graph.write_png('tree.png')
      5


TypeError: cannot unpack non-iterable Dot object
```

[94]:
```
# Get numerical feature importances
importances = list(rf.feature_importances_)
# List of tuples with variable and importance
RfCOVIDdf_importances = [(RfCOVIDdf, round(importance, 2)) for RfCOVIDdf,␣
↪importance in zip(RfCOVIDdf_list, importances)]
# Sort the feature importances by most important first
RfCOVIDdf_importances = sorted(RfCOVIDdf_importances, key = lambda x: x[1],␣
↪reverse = True)
# Print out the feature and importances
[print('Variable: {:20} Importance: {}'.format(*pair)) for pair in␣
↪RfCOVIDdf_importances];
```

```
Variable: Age Group_90 and older Importance: 0.29
Variable: Age Group_80 to 89 Years Importance: 0.22
Variable: Age Group_70 to 79 Years Importance: 0.19
Variable: Age Group_60 to 69 Years Importance: 0.07
Variable: NeighbourhoodAvgIncome Importance: 0.04
Variable: Client Gender_FEMALE Importance: 0.02
Variable: Age Group_50 to 59 Years Importance: 0.01
Variable: Client Gender_MALE   Importance: 0.01
Variable: Age Group_19 and younger Importance: 0.0
Variable: Age Group_20 to 29 Years Importance: 0.0
Variable: Age Group_30 to 39 Years Importance: 0.0
Variable: Age Group_40 to 49 Years Importance: 0.0
Variable: Neighbourhood Name_Agincourt North Importance: 0.0
Variable: Neighbourhood Name_Agincourt South-Malvern West Importance: 0.0
Variable: Neighbourhood Name_Alderwood Importance: 0.0
Variable: Neighbourhood Name_Annex Importance: 0.0
Variable: Neighbourhood Name_Banbury-Don Mills Importance: 0.0
Variable: Neighbourhood Name_Bathurst Manor Importance: 0.0
Variable: Neighbourhood Name_Bay Street Corridor Importance: 0.0
Variable: Neighbourhood Name_Bayview Village Importance: 0.0
Variable: Neighbourhood Name_Bayview Woods-Steeles Importance: 0.0
Variable: Neighbourhood Name_Bedford Park-Nortown Importance: 0.0
Variable: Neighbourhood Name_Beechborough-Greenbrook Importance: 0.0
```

```
Variable: Neighbourhood Name_Bendale Importance: 0.0
Variable: Neighbourhood Name_Birchcliffe-Cliffside Importance: 0.0
Variable: Neighbourhood Name_Black Creek Importance: 0.0
Variable: Neighbourhood Name_Blake-Jones Importance: 0.0
Variable: Neighbourhood Name_Briar Hill-Belgravia Importance: 0.0
Variable: Neighbourhood Name_Bridle Path-Sunnybrook-York Mills Importance: 0.0
Variable: Neighbourhood Name_Broadview North Importance: 0.0
Variable: Neighbourhood Name_Brookhaven-Amesbury Importance: 0.0
Variable: Neighbourhood Name_Cabbagetown-South St. James Town Importance: 0.0
Variable: Neighbourhood Name_Caledonia-Fairbank Importance: 0.0
Variable: Neighbourhood Name_Casa Loma Importance: 0.0
Variable: Neighbourhood Name_Centennial Scarborough Importance: 0.0
Variable: Neighbourhood Name_Church-Yonge Corridor Importance: 0.0
Variable: Neighbourhood Name_Clairlea-Birchmount Importance: 0.0
Variable: Neighbourhood Name_Clanton Park Importance: 0.0
Variable: Neighbourhood Name_Cliffcrest Importance: 0.0
Variable: Neighbourhood Name_Corso Italia-Davenport Importance: 0.0
Variable: Neighbourhood Name_Danforth Importance: 0.0
Variable: Neighbourhood Name_Danforth East York Importance: 0.0
Variable: Neighbourhood Name_Don Valley Village Importance: 0.0
Variable: Neighbourhood Name_Dorset Park Importance: 0.0
Variable: Neighbourhood Name_Dovercourt-Wallace Emerson-Junction Importance: 0.0
Variable: Neighbourhood Name_Downsview-Roding-CFB Importance: 0.0
Variable: Neighbourhood Name_Dufferin Grove Importance: 0.0
Variable: Neighbourhood Name_East End-Danforth Importance: 0.0
Variable: Neighbourhood Name_Edenbridge-Humber Valley Importance: 0.0
Variable: Neighbourhood Name_Eglinton East Importance: 0.0
Variable: Neighbourhood Name_Elms-Old Rexdale Importance: 0.0
Variable: Neighbourhood Name_Englemount-Lawrence Importance: 0.0
Variable: Neighbourhood Name_Eringate-Centennial-West Deane Importance: 0.0
Variable: Neighbourhood Name_Etobicoke West Mall Importance: 0.0
Variable: Neighbourhood Name_Flemingdon Park Importance: 0.0
Variable: Neighbourhood Name_Forest Hill North Importance: 0.0
Variable: Neighbourhood Name_Forest Hill South Importance: 0.0
Variable: Neighbourhood Name_Glenfield-Jane Heights Importance: 0.0
Variable: Neighbourhood Name_Greenwood-Coxwell Importance: 0.0
Variable: Neighbourhood Name_Guildwood Importance: 0.0
Variable: Neighbourhood Name_Henry Farm Importance: 0.0
Variable: Neighbourhood Name_High Park North Importance: 0.0
Variable: Neighbourhood Name_High Park-Swansea Importance: 0.0
Variable: Neighbourhood Name_Highland Creek Importance: 0.0
Variable: Neighbourhood Name_Hillcrest Village Importance: 0.0
Variable: Neighbourhood Name_Humber Heights-Westmount Importance: 0.0
Variable: Neighbourhood Name_Humber Summit Importance: 0.0
Variable: Neighbourhood Name_Humbermede Importance: 0.0
Variable: Neighbourhood Name_Humewood-Cedarvale Importance: 0.0
Variable: Neighbourhood Name_Ionview Importance: 0.0
Variable: Neighbourhood Name_Islington-City Centre West Importance: 0.0
```

```
Variable: Neighbourhood Name_Junction Area Importance: 0.0
Variable: Neighbourhood Name_Keelesdale-Eglinton West Importance: 0.0
Variable: Neighbourhood Name_Kennedy Park Importance: 0.0
Variable: Neighbourhood Name_Kensington-Chinatown Importance: 0.0
Variable: Neighbourhood Name_Kingsview Village-The Westway Importance: 0.0
Variable: Neighbourhood Name_Kingsway South Importance: 0.0
Variable: Neighbourhood Name_L'Amoreaux Importance: 0.0
Variable: Neighbourhood Name_Lambton Baby Point Importance: 0.0
Variable: Neighbourhood Name_Lansing-Westgate Importance: 0.0
Variable: Neighbourhood Name_Lawrence Park North Importance: 0.0
Variable: Neighbourhood Name_Lawrence Park South Importance: 0.0
Variable: Neighbourhood Name_Leaside-Bennington Importance: 0.0
Variable: Neighbourhood Name_Little Portugal Importance: 0.0
Variable: Neighbourhood Name_Long Branch Importance: 0.0
Variable: Neighbourhood Name_Malvern Importance: 0.0
Variable: Neighbourhood Name_Maple Leaf Importance: 0.0
Variable: Neighbourhood Name_Markland Wood Importance: 0.0
Variable: Neighbourhood Name_Milliken Importance: 0.0
Variable: Neighbourhood Name_Mimico (includes Humber Bay Shores) Importance: 0.0
Variable: Neighbourhood Name_Morningside Importance: 0.0
Variable: Neighbourhood Name_Moss Park Importance: 0.0
Variable: Neighbourhood Name_Mount Dennis Importance: 0.0
Variable: Neighbourhood Name_Mount Olive-Silverstone-Jamestown Importance: 0.0
Variable: Neighbourhood Name_Mount Pleasant East Importance: 0.0
Variable: Neighbourhood Name_Mount Pleasant West Importance: 0.0
Variable: Neighbourhood Name_New Toronto Importance: 0.0
Variable: Neighbourhood Name_Newtonbrook East Importance: 0.0
Variable: Neighbourhood Name_Newtonbrook West Importance: 0.0
Variable: Neighbourhood Name_Niagara Importance: 0.0
Variable: Neighbourhood Name_North Riverdale Importance: 0.0
Variable: Neighbourhood Name_North St. James Town Importance: 0.0
Variable: Neighbourhood Name_O'Connor-Parkview Importance: 0.0
Variable: Neighbourhood Name_Oakridge Importance: 0.0
Variable: Neighbourhood Name_Oakwood Village Importance: 0.0
Variable: Neighbourhood Name_Old East York Importance: 0.0
Variable: Neighbourhood Name_Palmerston-Little Italy Importance: 0.0
Variable: Neighbourhood Name_Parkwoods-Donalda Importance: 0.0
Variable: Neighbourhood Name_Pelmo Park-Humberlea Importance: 0.0
Variable: Neighbourhood Name_Playter Estates-Danforth Importance: 0.0
Variable: Neighbourhood Name_Pleasant View Importance: 0.0
Variable: Neighbourhood Name_Princess-Rosethorn Importance: 0.0
Variable: Neighbourhood Name_Regent Park Importance: 0.0
Variable: Neighbourhood Name_Rexdale-Kipling Importance: 0.0
Variable: Neighbourhood Name_Rockcliffe-Smythe Importance: 0.0
Variable: Neighbourhood Name_Roncesvalles Importance: 0.0
Variable: Neighbourhood Name_Rosedale-Moore Park Importance: 0.0
Variable: Neighbourhood Name_Rouge Importance: 0.0
Variable: Neighbourhood Name_Runnymede-Bloor West Village Importance: 0.0
```

```
Variable: Neighbourhood Name_Rustic Importance: 0.0
Variable: Neighbourhood Name_Scarborough Village Importance: 0.0
Variable: Neighbourhood Name_South Parkdale Importance: 0.0
Variable: Neighbourhood Name_South Riverdale Importance: 0.0
Variable: Neighbourhood Name_St.Andrew-Windfields Importance: 0.0
Variable: Neighbourhood Name_Steeles Importance: 0.0
Variable: Neighbourhood Name_Stonegate-Queensway Importance: 0.0
Variable: Neighbourhood Name_Tam O'Shanter-Sullivan Importance: 0.0
Variable: Neighbourhood Name_Taylor-Massey Importance: 0.0
Variable: Neighbourhood Name_The Beaches Importance: 0.0
Variable: Neighbourhood Name_Thistletown-Beaumond Heights Importance: 0.0
Variable: Neighbourhood Name_Thorncliffe Park Importance: 0.0
Variable: Neighbourhood Name_Trinity-Bellwoods Importance: 0.0
Variable: Neighbourhood Name_University Importance: 0.0
Variable: Neighbourhood Name_Victoria Village Importance: 0.0
Variable: Neighbourhood Name_Waterfront Communities-The Island Importance: 0.0
Variable: Neighbourhood Name_West Hill Importance: 0.0
Variable: Neighbourhood Name_West Humber-Clairville Importance: 0.0
Variable: Neighbourhood Name_Westminster-Branson Importance: 0.0
Variable: Neighbourhood Name_Weston Importance: 0.0
Variable: Neighbourhood Name_Weston-Pellam Park Importance: 0.0
Variable: Neighbourhood Name_Wexford/Maryvale Importance: 0.0
Variable: Neighbourhood Name_Willowdale East Importance: 0.0
Variable: Neighbourhood Name_Willowdale West Importance: 0.0
Variable: Neighbourhood Name_Willowridge-Martingrove-Richview Importance: 0.0
Variable: Neighbourhood Name_Woburn Importance: 0.0
Variable: Neighbourhood Name_Woodbine Corridor Importance: 0.0
Variable: Neighbourhood Name_Woodbine-Lumsden Importance: 0.0
Variable: Neighbourhood Name_Wychwood Importance: 0.0
Variable: Neighbourhood Name_Yonge-Eglinton Importance: 0.0
Variable: Neighbourhood Name_Yonge-St.Clair Importance: 0.0
Variable: Neighbourhood Name_York University Heights Importance: 0.0
Variable: Neighbourhood Name_Yorkdale-Glen Park Importance: 0.0
Variable: Client Gender_UNKNOWN Importance: 0.0
```

[95]:

```
# New random forest with only the two most important variables
rf_most_important = RandomForestRegressor(n_estimators= 1, random_state=1)
# Extract the two most important features
important_indices = [RfCOVIDdf_list.index('Age Group_90 and older'),␣
 ↪RfCOVIDdf_list.index('Age Group_80 to 89 Years')]
train_important = train_COVIDdf[:, important_indices]
test_important = test_COVIDdf[:, important_indices]
# Train the random forest
rf_most_important.fit(train_important, train_labels)
# Make predictions and determine the error
predictions = rf_most_important.predict(test_important)
errors = abs(predictions - test_labels)
```

```python
# Display the performance metrics
print('Mean Absolute Error:', round(np.mean(errors), 2), 'degrees.')
mape = np.mean(100 * (errors / test_labels))

Accuracycount = 0

for i in range(0, len(predictions)):
    if predictions[i] == test_labels[i]:
          Accuracycount +=1

Accuracy = 100* Accuracycount/ len(predictions)
Accuracy
```

```
Mean Absolute Error: 0.29 degrees.

/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: RuntimeWarning:
divide by zero encountered in true_divide
```

[95]: 0.0

[96]:
```python
#K-FOLD CROSS VALIDATION
```

[97]:
```python
kfCOVIDdf = COVIDdf
```

[98]:
```python
X = kfCOVIDdf.drop('Outcome', axis =1).values
Y = kfCOVIDdf['Outcome'].values

#Creating X and Y values for K-fold
```

[99]:
```python
scaler = MinMaxScaler(feature_range=(0, 1))
X = scaler.fit_transform(X)
```

[100]:
```python
kfold = model_selection.KFold(n_splits=10, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))

# evaluate a logistic regression model using repeated k-fold cross-validation,␣
 ↪10-fold
```

```
Accuracy: 88.93%
```

[101]:
```python
kfold = model_selection.KFold(n_splits=9, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

Accuracy: 88.64%

[102]:
```
kfold = model_selection.KFold(n_splits=8, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

Accuracy: 88.80%

[103]:
```
kfold = model_selection.KFold(n_splits=7, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

Accuracy: 88.80%

[104]:
```
kfold = model_selection.KFold(n_splits=6, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

Accuracy: 88.76%

[105]:
```
kfold = model_selection.KFold(n_splits=5, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

Accuracy: 88.76%

[106]:
```
kfold = model_selection.KFold(n_splits=4, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

Accuracy: 88.64%

[107]:
```
kfold = model_selection.KFold(n_splits=3, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

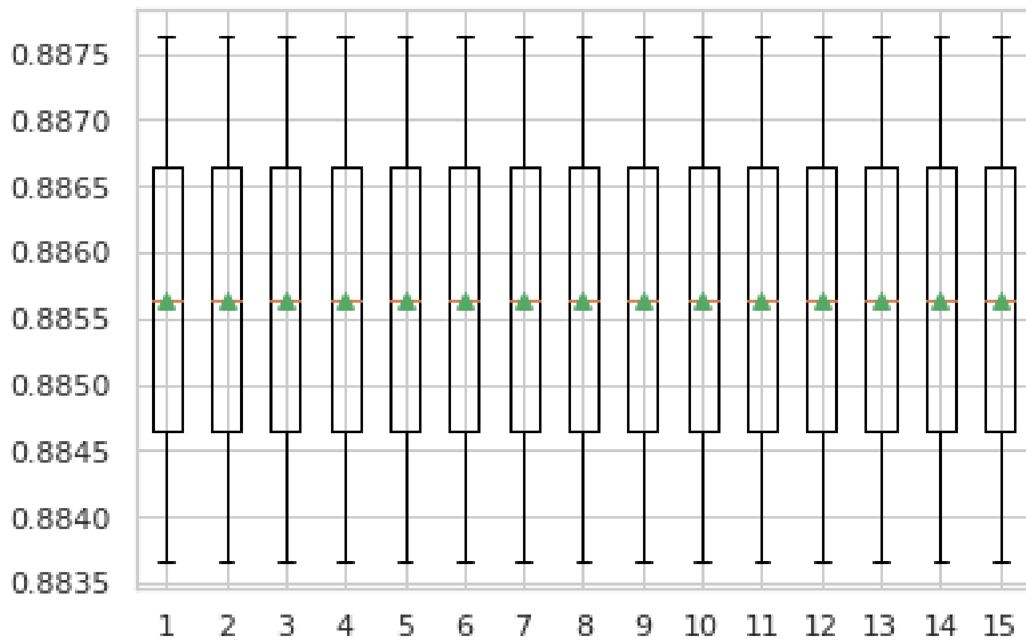
Accuracy: 88.69%

[108]:
```
kfold = model_selection.KFold(n_splits=2, random_state=100, shuffle = True)
model_kfold = LogisticRegression()
results_kfold = model_selection.cross_val_score(model_kfold, X, Y, cv=kfold)
print("Accuracy: %.2f%%" % (results_kfold.mean()*100.0))
```

Accuracy: 88.56%

[109]:
```
from scipy.stats import sem
from matplotlib import pyplot
def evaluate_model(X, Y, repeats):

    cv = RepeatedKFold(n_splits=10, n_repeats=repeats, random_state=1)
    model = LogisticRegression()
    scores = cross_val_score(model, X, Y, scoring='accuracy', cv=kfold,␣
↪n_jobs=-1)
    return scores

repeats = range(1,16)
results = list()

for r in repeats:
    scores = evaluate_model(X, Y, r)
    print('>%d mean=%.4f se=%.3f' % (r, mean(scores), sem(scores)))
    results.append(scores)
# plot the results
pyplot.boxplot(results, labels=[str(r) for r in repeats], showmeans=True)
pyplot.show()

#Created function to find mean, se for K-folds of 1-15
#Created box-and-whisker plot for visualization
```

```
>1 mean=0.8856 se=0.002
>2 mean=0.8856 se=0.002
>3 mean=0.8856 se=0.002
>4 mean=0.8856 se=0.002
>5 mean=0.8856 se=0.002
>6 mean=0.8856 se=0.002
>7 mean=0.8856 se=0.002
>8 mean=0.8856 se=0.002
>9 mean=0.8856 se=0.002
>10 mean=0.8856 se=0.002
>11 mean=0.8856 se=0.002
>12 mean=0.8856 se=0.002
>13 mean=0.8856 se=0.002
>14 mean=0.8856 se=0.002
>15 mean=0.8856 se=0.002
```


0.8875
0.8870
0.8865
0.8860
0.8855
0.8850
0.8845
0.8840
0.8835
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

[ ]: