# CovidTO

November 24, 2020

# Predicting Covid Cases in Toronto Neighbourhoods: An Analysis using Linear Regression

Alexei Marcilio

November 24, 2020

### Executive Summary

This study used multiple linear regression to predict covid rates in Toronto's 140 neighborhoods using demographic data from Stats Canada. The correlation of these features to the target variable were analyzed in order to limit the number of potential predictors. Features which were highly correlated to each other were removed and an analysis of the p-values were done recursively to eliminate further predictors. Using the IQR method a few neighborhoods were removed that were considered outliers. The final model used only three variables to predict covid rates in the neighborhoods with an  2 value of 0.605. Specifically these variables in the Stat Canada demographic database are know as Black (Visible Minority), Occupations in manufacturing and utilities, and Journey to work Between 12 p.m. and 4:59 a.m. A model such as this one (*Rate = 1038 + 157.7 Black% + 140.8 ManufactJob + 170.1 WorkNights*) could be useful in order to determine where to target prescriptive or preemptive action.

## Introduction

The purpose of this study is to build a linear regression model that will predict the number of covid-19 cases for any Toronto neighborhood based on its demographic data. Predicting covid-19 cases is useful in that an understanding of which socioeconomic factors that influence the growth of disease in a community will help with our understanding of the virus. This knowledge will enable resources to be better targeted in the future to help prevent transmission during this, or any future pandemic.

### Load Libraries

[150]:
```
# Import libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
import numpy as np
%matplotlib inline
import warnings
from scipy import stats
warnings.filterwarnings('ignore')
```

### Data

A population Census is held across Canada every 5 years. It collects data about age and sex, families and households, language, immigration and internal migration, racial diversity, Aboriginal peoples, housing, education, income, and labor. The City of Toronto Neighborhood Profiles use this Census data to provide a portrait of the demographic, social and economic characteristics of the people and households in each City of Toronto neighborhood[1].

The data is made available through Toronto's Open Data portal (https://open.toronto.ca/)[2].

Two datasets from this site were used for this study. One that shows the number of covid cases by Toronto neighborhood (https://open.toronto.ca/dataset/covid-19-cases-in-toronto/) and the other contains demographic features of each of these neighborhoods (https://open.toronto.ca/dataset/neighbourhood-profiles/). There are over 2,300 features show for each neighborhood including:

- Aboriginal peoples
- Age and sex
- Education
- Families, households and marital status
- Housing
- Immigration and ethnocultural diversity
- Income
- Journey to work
- Labour
- Language
- Language of work
- Mobility and migration
- Population and dwelling counts
- Type of dwelling

#### Load Files

[362]:
```
covid_to = pd.read_csv('data/CityofToronto_COVID-19_NeighbourhoodData.csv')
```

[363]:
```
neigh = pd.read_csv("data/neighbourhood-profiles-2016-csv_ADJ.csv")
```

#### Data Cleaning

The demographic data is a huge dataset containing over 2300 rows, each one representing a different demographic feature. In order to create one usable dataset the following tasks were performed:

1. The dataset was transposed so that the columns, which are the Toronto neighborhoods become row and each feature becomes a column.
2. Every column is given a unique identifying name.
3. A dataframe was created to capture the column descriptions which are quite lengthy.
4. Null rows were dropped - these were summary rows which did not relate to each neighborhood.
5. All the numeric columns were converted to floats - there were no categorical columns.
6. The two dataframes (demographic data and covid case data) were merged on the neighborhood id column to create one dataframe.

[364]:
```
# The Neighbourhood file has Neighbourhoods as columns so we must
# transpose it.
neighTO = neigh.transpose()
```

There are over 2300 columns. Let's keep track of the names meanings of these columns so we can later use this information to interpret the results. In the meantime we will refer to the columns by their numeric names.

[365]:
```
# Let's the column names to strings
neighTO.columns = ["Col_" + str(x) for x in neighTO.columns]
```

[366]:
```
colNames = neighTO.iloc[0:5,:].transpose()
```

[367]:
```
# Now we have a dataframe of all the column names
colNames.head(5).iloc[:,0:5]
```

[367]:
```
      _id                  Category                          Topic Special  \
Col_0   1  Neighbourhood Information  Neighbourhood Information     NaN
Col_1   1  Neighbourhood Information  Neighbourhood Information     NaN
Col_2   3                Population   Population and dwellings       X
Col_3   4                Population   Population and dwellings     NaN
Col_4   6                Population   Population and dwellings     NaN

                Characteristic
Col_0     Neighbourhood Number
Col_1     Neighbourhood Number
Col_2         Population, 2016
Col_3         Population, 2011
Col_4  Total private dwellings
```

[368]:
```
# Let's remove all the descriptive rows
neighTO = neighTO.iloc[5:,:]
```

[369]:
```
# We can remove the city of Toronto
neighTO = neighTO[neighTO.index != 'City of Toronto']
```

[370]:
```
# Let's ensure the neighborhoodID in both files is an int so we can
# join the files
neighTO['Col_0'] = neighTO['Col_0'].astype(int)
```

[371]:
```
# We drop the null row and convert ID to int
covid_to.dropna(inplace=True)
covid_to['Neighbourhood ID'] = covid_to['Neighbourhood ID'].astype(int)
```

[372]:
```
# Let's convert the demographic data to floats - select only string columns
for x in neighTO.select_dtypes(include='object').columns:
    if x != 'Col_0':
        # Remove % symbol if present
        neighTO[x] = neighTO[x].replace({'%':''}, regex = True)
        neighTO[x] = neighTO[x].astype(float)
```

[373]:
```
# We see they are all floats except for the int column
neighTO.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 140 entries, Agincourt North to Yorkdale-Glen Park
Columns: 2306 entries, Col_0 to Col_2305
dtypes: float64(2305), int64(1)
memory usage: 2.5+ MB
```

[374]:
```
# Now we can see that all our columns are numeric
neighTO.describe().iloc[:,1:5]
```

[374]:

| | Col_1 | Col_2 | Col_3 | Col_4 |
|---|---|---|---|---|
| count | 140.000000 | 140.000000 | 140.000000 | 140.000000 |
| mean | 0.434000 | 19511.221429 | 96.683000 | 42.818071 |
| std | 0.319835 | 10033.589222 | 6.728256 | 8.067717 |
| min | 0.000000 | 6577.000000 | 65.790000 | 29.450000 |
| 25% | 0.185000 | 12019.500000 | 95.925000 | 37.540000 |
| 50% | 0.360000 | 16749.500000 | 98.550000 | 40.930000 |
| 75% | 0.630000 | 23854.500000 | 100.347500 | 45.862500 |
| max | 1.700000 | 65913.000000 | 108.220000 | 71.620000 |

We need to join the datasets so we have one dataset with all the features and target.

[375]:
```
# Rename the column so they match in both files
neighTO.rename(columns={'Col_0': "Neighbourhood ID"}, inplace=True)
```

[376]:
```
# Join the files on Neighbourhood ID
NeighCases = pd.merge(neighTO, covid_to, how='left', on=['Neighbourhood ID'])
```

[377]:
```
# Let's write this to excel to back it up.
NeighCases.to_csv("data/NeigCases.csv")
```

### Data Exploration

There's a huge difference in the rate of covid-19 cases in each Toronto neighborhood.

The top 5 neighborhoods **dramatically higher case rates** than the bottom five neighborhoods (Figure 1).

[518]:
```
sns.reset_orig()
df = NeighCases[['Neighbourhood Name', "Rate per 100,000 people"]].
 ↪sort_values("Rate per 100,000 people")
df.columns = ['Neighbourhood', 'Rate per 100,000 people']

ax = sns.barplot(data=df.head(5).append(df.tail(5)), y='Neighbourhood',x='Rate␣
 ↪per 100,000 people',orient='h')
txt="Figure 1. The top 5 and bottom 5 Toronto neighborhoods by covid case rate."
```

```python
# plt.figtext(0.5, 0.01, txt, wrap=True, horizontalalignment='center',␣
↪fontsize=14)
plt.figtext(0.5, -0.1, txt, wrap=True, horizontalalignment='center',␣
↪fontsize=13)

plt.title('The Top 5 and Bottom 5 Toronto Neighbourhoods by Covid Case␣
↪Rate',fontsize= 15)
plt.xlabel("Covid Case Rate per 100,000 people", fontsize=12)
plt.ylabel("Toronto Neighbourhood", fontsize=12);
```

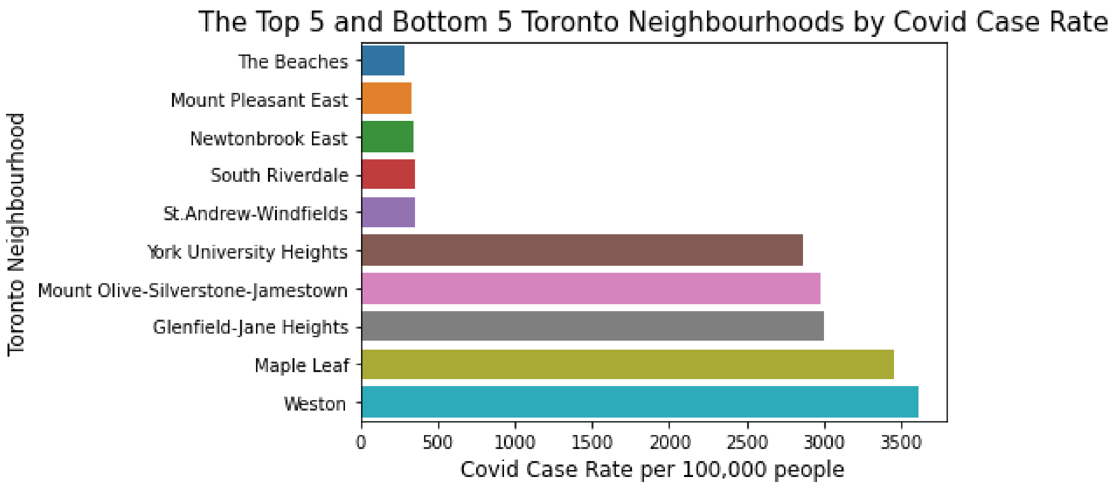



Figure 1. The top 5 and bottom 5 Toronto neighborhoods by covid case rate.

The rate in the neighborhood of Weston has a covid case rate of almost 13 times that of the Beaches neighborhood.

[379]:
```python
df.head(1).append(df.tail(1))
```

[379]:

| | Neighbourhood | Rate per 100,000 people |
|---|---|---|
| 116 | The Beaches | 282.839523 |
| 126 | Weston | 3612.716763 |

Various media reports have shown that certain demographic factors can influence covid rates[3]. Reports have shown that the virus has disproportionately affected communities based on the following factors:

- Racial Profile
- Density
- Employment
- Income

Let's examine some columns that correspond to these factors and see how they relate to covid rate.

[380]:
```python
colNames.Category.unique()
```

[380]:
```
array(['Neighbourhood Information', 'Population',
       'Families, households and marital status', 'Language', 'Income',
       'Immigration and citizenship', 'Visible minority', 'Ethnic origin',
       'Aboriginal peoples', 'Education', 'Housing', 'Language of work',
       'Labour', 'Journey to work', 'Mobility'], dtype=object)
```

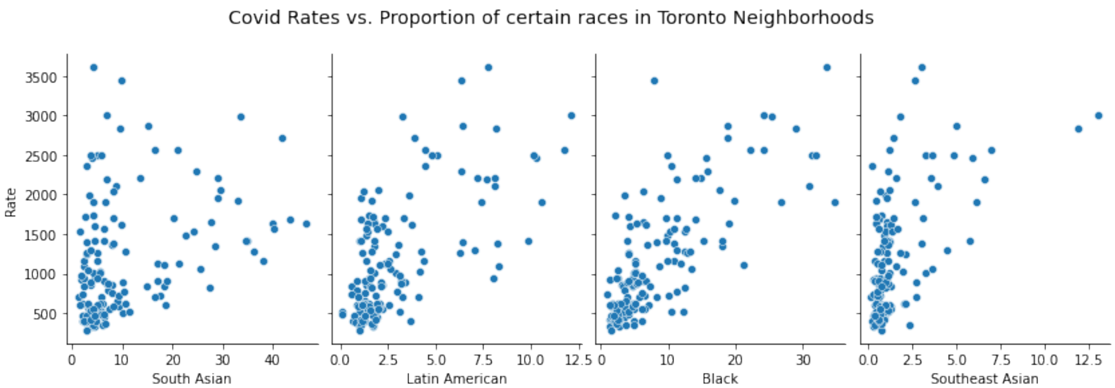
The following graph shows the how covid rates vary depending on the proportion of different racial groups in each Toronto neighborhood.

There seems to be a moderate positive relationship between covid rates and the percent of Blacks in each neighborhood and also Latin Americans.

[388]:
```python
colNames[colNames['Category'] == 'Visible minority'].index
```

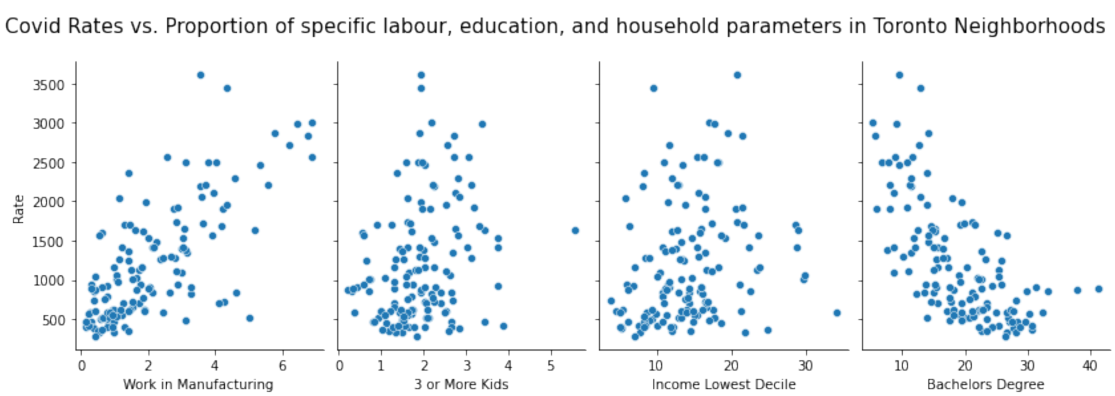
[388]:
```
Index(['Col_1077', 'Col_1265', 'Col_1266', 'Col_1267', 'Col_1268', 'Col_1269',
       'Col_1270', 'Col_1271', 'Col_1272', 'Col_1273', 'Col_1274', 'Col_1275',
       'Col_1276', 'Col_1277', 'Col_1278'],
      dtype='object')
```

[397]:
```python
# Let's look at the top 5 visible minority categories
df = NeighCases[colNames[colNames['Category'] == 'Visible minority'].index]
df = df.rename(columns={"Col_1266": "Visible Minority", "Col_1267": "South␣
↪Asian",
                  "Col_1077": "Latin American", "Col_1269": "Black", "Col_1272":
↪ "Southeast Asian"})
df['Rate'] = NeighCases['Rate per 100,000 people']
df = df[["South Asian", "Latin American", "Black", "Southeast Asian","Rate"]]
#Create the pairplot
ax = sns.pairplot(data=df,
            x_vars=["South Asian", "Latin American", "Black", "Southeast␣
↪Asian"],
            y_vars=['Rate'], diag_kind=None);
#Add titles
ax.fig.suptitle("Covid Rates vs. Proportion of certain races in Toronto␣
↪Neighborhoods", y=1.08, fontsize=14) # y= some height>1
ax.fig.set_size_inches(12,4)
```




Now lets check some other potential predictors, including income, density and employment.

[398]:
```python
# Let's look at the top 5 visible minority categories
df = NeighCases[["Col_1855", "Col_100", "Col_1047", "Col_1635"]]
df = df.rename(columns={"Col_1855": "Work in Manufacturing", "Col_100": "3 or More Kids",\
                        "Col_1047": "Income Lowest Decile", "Col_1635": "Bachelors Degree"})
df['Rate'] = NeighCases['Rate per 100,000 people']
# Create the pairplot
ax = sns.pairplot(data=df,
            x_vars=["Work in Manufacturing", "3 or More Kids",\
                    "Income Lowest Decile", "Bachelors Degree"],
            y_vars=['Rate'], diag_kind=None);
# Add titles
ax.fig.suptitle("Covid Rates vs. Proportion of specific labour, education, \
and household parameters in Toronto Neighborhoods", y=1.08, fontsize=15) # y= some height>1
# Set the size
ax.fig.set_size_inches(12,4)
# Add the figure text below
```



There seems to be a positive trend between the proportion of the population that work in manufacturing jobs in a neighborhood vs. covid rates, and a negative trend between the proportion who have Bachelor's Degrees (Figure 3).

### Feature Selection

There are so many features (over 2,300) so in order to narrow it down we will examine the correlation of every feature to the target (Covid rate per 100,000 people). One of the most common ways to identify potential features is to examine the correlation between columns of X and y (Corr)[4].

We will select the top 100 of based on this correlation and use this as our new pared down dataset.

[464]:
```
# Let's look at all the features and how they compare to
# the target - this will show the top 100 features
cor = NeighCases.corr()
threshold = 0.6
a=abs(cor['Rate per 100,000 people'])
result=pd.DataFrame(a[a>0.6])
feat_cor = result.sort_values('Rate per 100,000 people', ascending=False).
 ↪head(100)
```

[465]:
```
# Let's remove the Case Count column and examine these variables
feat_cor.drop('Case Count', inplace=True)
# feat_cor.drop('Rate per 100,000 people', inplace=True)
```

[466]:
```
try:
    feat_cor.reset_index(level=0, inplace=True)
except:
    None
```

[467]:
```
try:
    colNames.reset_index(level=0, inplace=True)
except:
    None
```

[468]:
```
dfCol = colNames[['index','Category', 'Characteristic']]
```

[404]:
```
feat_corName = pd.merge(feat_cor, dfCol, how='left', on=['index'])
```

The top 10 features in order of descending correlation to "Rate per 100,000" column. The top features are all seemingly related to Black, Caribbean and African origins except for a Labor related column which is Occupations in manufacturing and utilities.

[405]:
```
# Here are the top 10 features in order of desending correlation to "Rate per␣
 ↪100,000"
feat_corName.sort_values(by='Rate per 100,000 people', ascending=False).head(10)
```

[405]:
```
                     index  Rate per 100,000 people  \
0  Rate per 100,000 people                 1.000000
1                 Col_1269                 0.744447
2                 Col_1105                 0.741047
3                 Col_1099                 0.737303
4                 Col_1377                 0.731488
5                  Col_329                 0.727007
6                 Col_1855                 0.726108
7                 Col_1135                 0.724949
8                 Col_1414                 0.713611
9                 Col_1329                 0.712460

                     Category  \
0                         NaN
1            Visible minority
2  Immigration and citizenship
3  Immigration and citizenship
4               Ethnic origin
5                    Language
6                      Labour
7  Immigration and citizenship
8               Ethnic origin
9               Ethnic origin

                                       Characteristic
0                                                 NaN
1                                               Black
2                                             Jamaica
3                                            Americas
4                                            Jamaican
5                              Niger-Congo languages
6      9 Occupations in manufacturing and utilities
7                                             Nigeria
8                    Central and West African origins
9                                   Caribbean origins
```

Let's look at the top correlation in each Category of predictors. The category with the top correlated predictor is "Visible Minority" (**0.75**) and "Housing" is the category with the lowest correlated predictor (**0.63**).

[406]:
```python
# Return top from each category
groupMax = feat_corName.groupby(["Category"])
groupMax.max('Rate per 100,000 people')
```

[406]:
```
                                          Rate per 100,000 people
Category
Education                                                0.677412
```

```
Ethnic origin                                            0.731488
Families, households and marital status                  0.690762
Housing                                                  0.629968
Immigration and citizenship                              0.741047
Income                                                   0.684972
Journey to work                                          0.666669
Labour                                                   0.726108
Language                                                 0.727007
Visible minority                                         0.744447
```

It seems logical that predictors within the same category would be correlated, so let's find the top correlated predictor within each category.

[407]:
```
# *****
# Let's find the rows with the highest value for each category
idx = feat_corName.groupby(['Category'])['Rate per 100,000 people'].
 ↪transform(max) == \
feat_corName['Rate per 100,000 people']
feat_corName[idx]
```

[407]:
```
       index  Rate per 100,000 people  \
1   Col_1269                 0.744447
2   Col_1105                 0.741047
4   Col_1377                 0.731488
5    Col_329                 0.727007
6   Col_1855                 0.726108
14   Col_105                 0.690762
16  Col_1049                 0.684972
19  Col_1635                 0.677412
27  Col_1907                 0.666669
74  Col_1594                 0.629968

                                   Category  \
1                          Visible minority
2               Immigration and citizenship
4                             Ethnic origin
5                                  Language
6                                    Labour
14  Families, households and marital status
16                                   Income
19                                Education
27                          Journey to work
74                                   Housing

                             Characteristic
1                                     Black
2                                   Jamaica
```

```
4                                               Jamaican
5                                  Niger-Congo languages
6        9 Occupations in manufacturing and utilities
14                                    3 or more children
16                                   In the third decile
19                                     Bachelor's degree
27                         Between 12 p.m. and 4:59 a.m.
74                               2 household maintainers
```

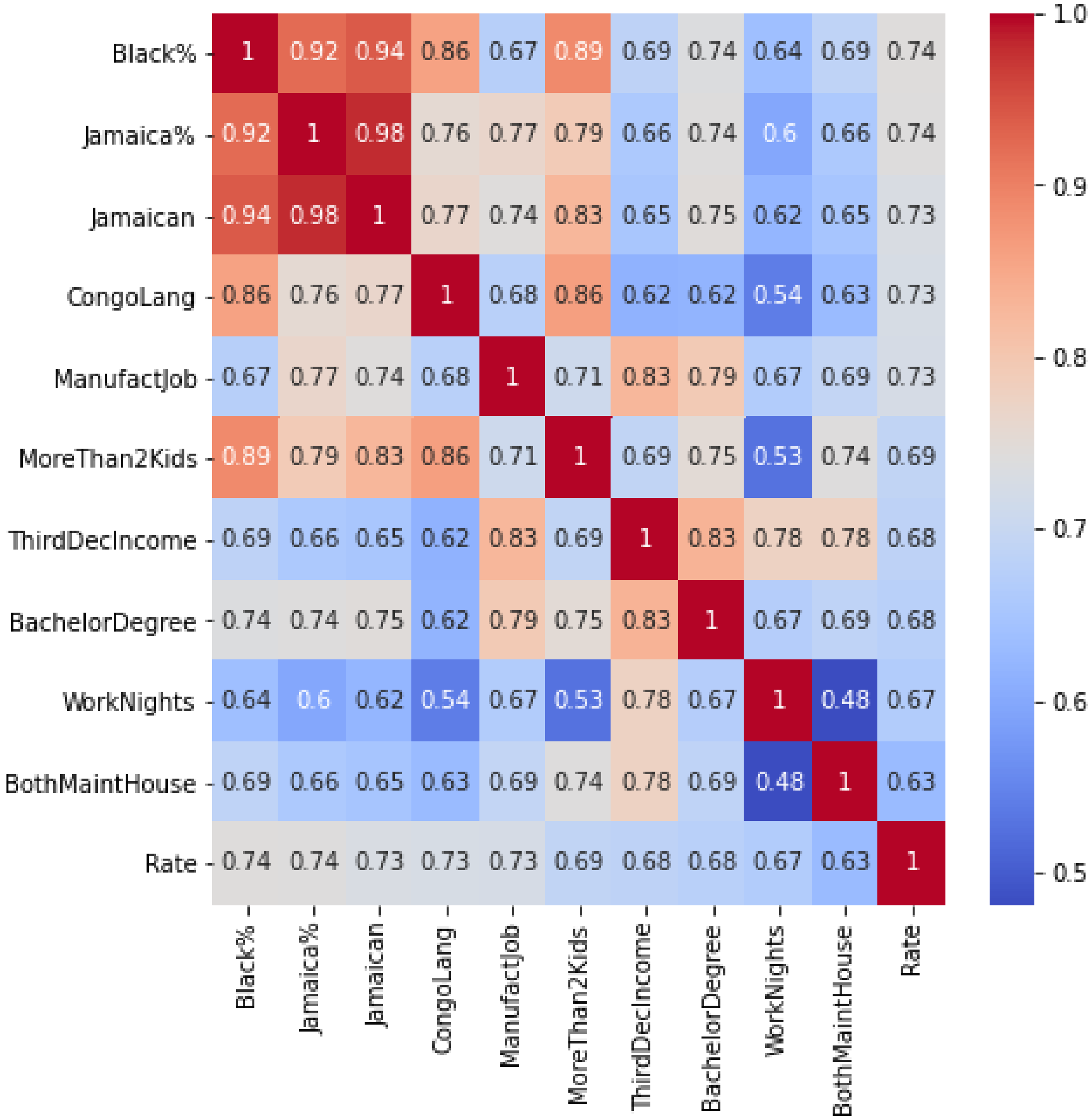
[408]:
```
# Let's create a dataframe using only these values and let's make the
# columns more readable.
NeighCovid = NeighCases[feat_corName[idx]['index']]
NeighCovid['Rate'] = NeighCases['Rate per 100,000 people']
```

[409]:
```
NeighCovid.columns = ['Black%', 'Jamaica%', 'Jamaican', 'CongoLang',␣
↪'ManufactJob', 'MoreThan2Kids',\
                       'ThirdDecIncome', 'BachelorDegree', 'WorkNights',␣
↪'BothMaintHouse', 'Rate']
```

### Remove Correlated Predictors

If certain features are highly correlated with other ones then there are redundant features and they can be removed to make the model more simple. The following heatmap shows that there are several highly correlated features, for example *Black* with *Jamaica%*, *Jamaican*, *CongoLang* and *MoreThan2Kids*.

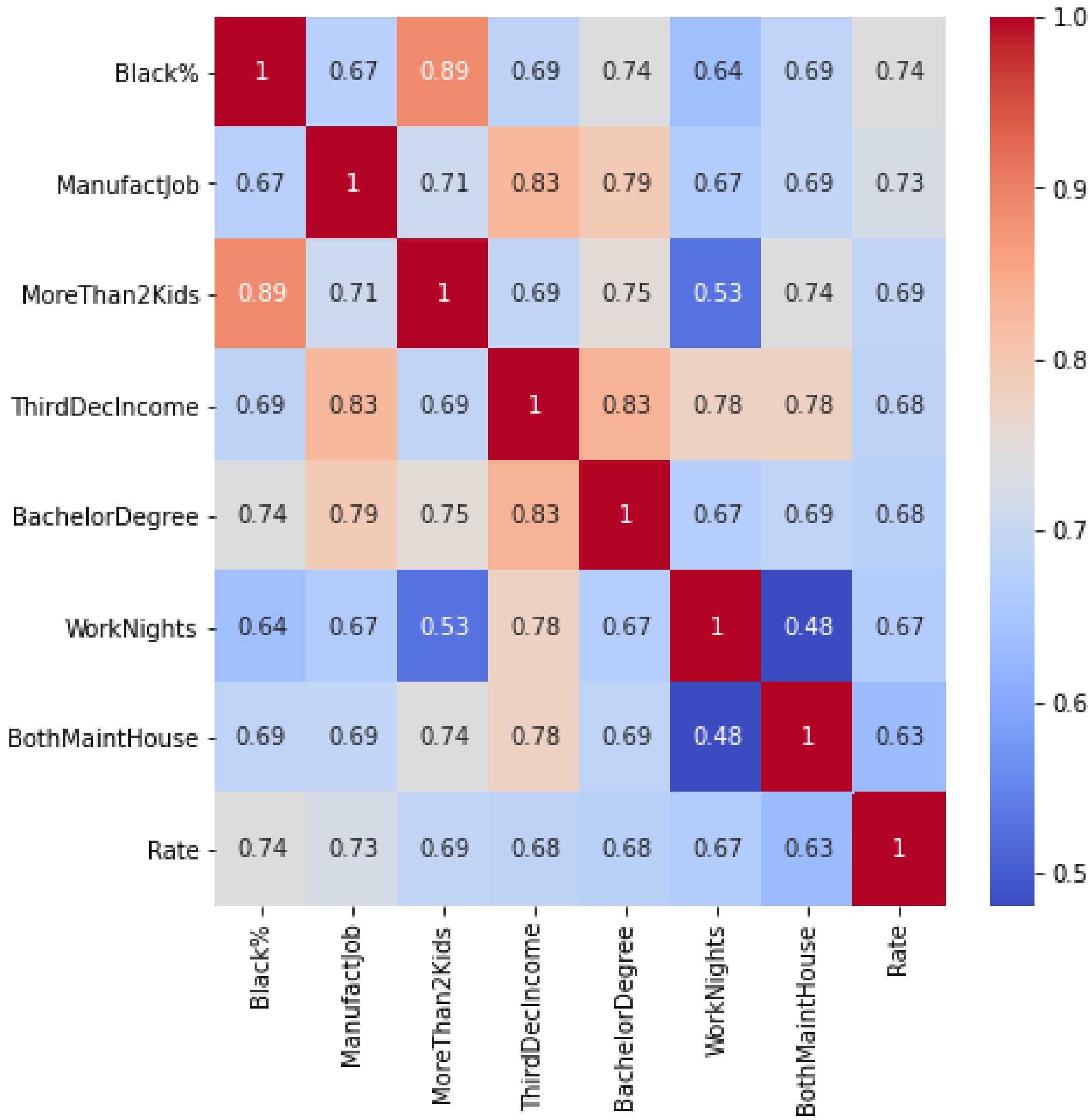
[410]:
```
# Correlation of variables
plt.figure(figsize=(7,7))
sns.heatmap(abs(NeighCovid.corr()), annot=True, cmap="coolwarm");
```




It makes sense than language, ethnic origin, language and visible minority categories of data are highly correlated. Let's remove the *Jamaica%*, *Jamaican*, and *CongoLang* predictors.

[411]:
```
# Remove columns from dataset
NeighCovid.drop(['Jamaica%', 'Jamaican', 'CongoLang'], axis=1, inplace=True)
```

[412]:
```
# Check the heatmap again
plt.figure(figsize=(7,7))
sns.heatmap(abs(NeighCovid.corr()), annot=True, cmap="coolwarm");
```

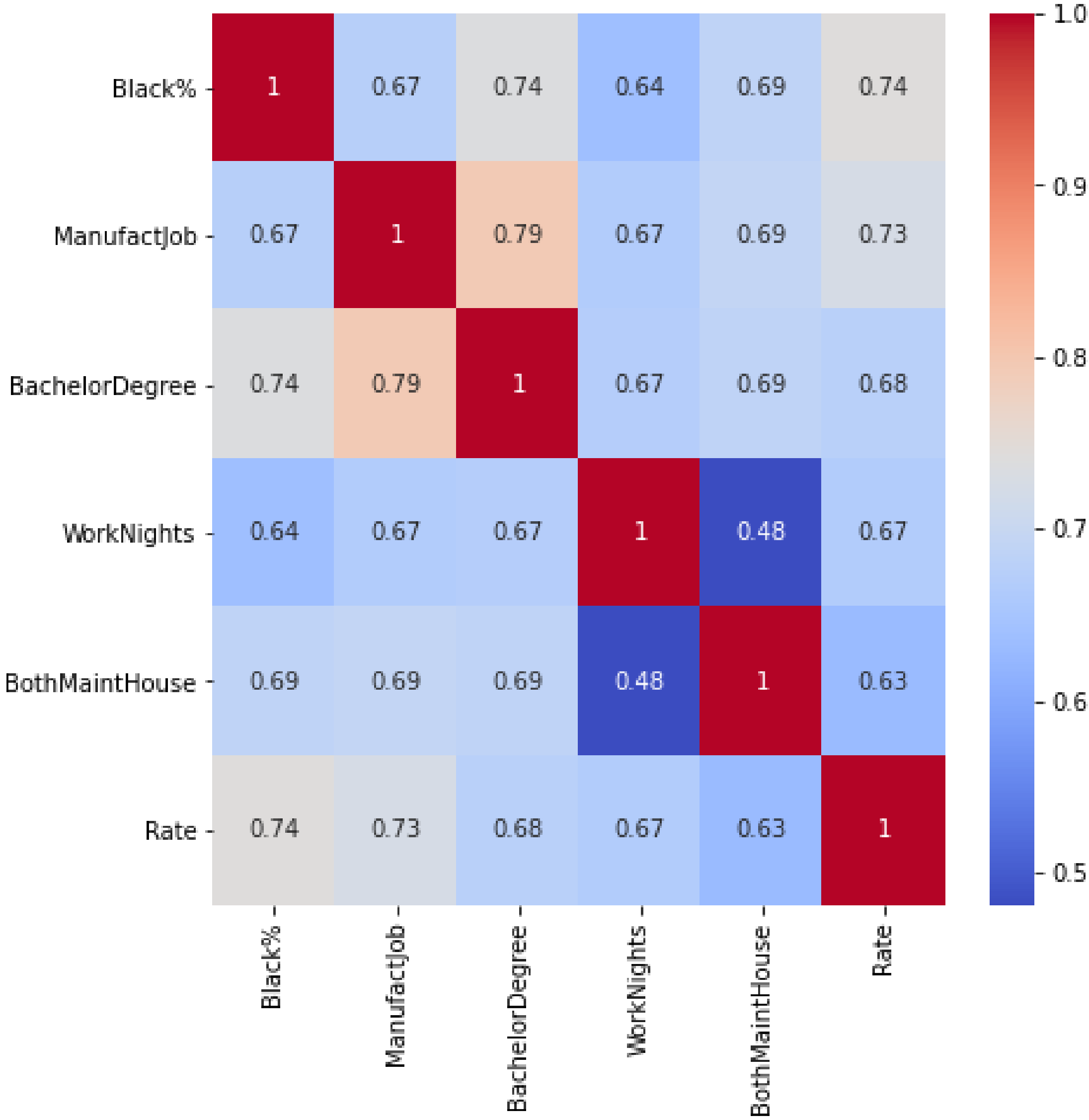



In the interest of simplicity we can reduce features further. Black and Morethan2Kids is highly correlated, as well as ManufactJob and ThirdDecIncome so we will reduce the features that are least correlated with Rate.

[413]:
```
NeighCovid.drop(['MoreThan2Kids', 'ThirdDecIncome'], axis=1, inplace=True)
```

Let's look at the heatmap of feature correlations. We can now see that we have all the correlations are under **0.80**. Let's use these remaining features to build a model.

[414]:
```
plt.figure(figsize=(7,7))
sns.heatmap(abs(NeighCovid.corr()), annot=True, cmap="coolwarm");
```




## Results

### Linear Regression

Linear regression is a linear way of modeling the relationship between a dependent variable (target) and one or more independent variables (features). If there one feature it is called simple linear regression, and if there is more than one it is called multiple linear regression.[5]

The equation for multiple linear regression is as follows:

$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$ where, for i=n observations: $y_i$ = dependent variable $x_i$ = expanatory variables $\beta_0$ = y-intercept (constant term) $\beta_p$ = slope coefficients for each explanatory variable $\epsilon$ = the model's error term (also known as the residuals

```
[415]: # Run the functions notebook
       %run CovidFunctions.ipynb
```

**Simple Linear Regression** Let's start with a simple linear regression model. We will use only one feature to predict the one target (Covid Rate). We will use the column which is most highly correlated to Covid Rate which is the percent of Black people in each neighborhood.

```
[416]: # Create the X and y datasets
       dfB = NeighCovid[['Black%','Rate']]
       X = NeighCovid[["Black%"]]
       y = NeighCovid["Rate"]
```

```
[417]: from sklearn.preprocessing import StandardScaler
       sc = StandardScaler()
       X = sc.fit_transform(X)
```

```
[418]: # Split data into train and test
       from sklearn.model_selection import train_test_split
       X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
        ↪random_state = 11)
```

```
[419]: # Create and train the model
       from sklearn.linear_model import LinearRegression
       #Create the model :
       regressor = LinearRegression()
       #Train the model :
       regressor.fit(X_train, y_train)
```

```
[419]: LinearRegression()
```

Here's the coefficient of the model:

```
[420]: list(regressor.coef_)
```

```
[420]: [535.7894650023125]
```

```
[421]: L = regressor.coef_
       L[0]
```

```
[421]: 535.7894650023125
```

```
[422]: # regressor.coef_
       # coeff_X = pd.DataFrame(regressor.coef_, index = NeighCovid.columns[:-1],␣
        ↪columns=['Coefficient'])
       # coeff_X
```

Here is the intercept:

[423]:
```
regressor.intercept_
```

[423]: 1217.1073039109747

[424]:
```
print("Rate = {:.2f} + {:.2f}*Black% ".format(regressor.intercept_,L[0]))
```

```
Rate = 1217.11 + 535.79*Black%
```

[425]:
```
y_pred = regressor.predict(X_test)
```

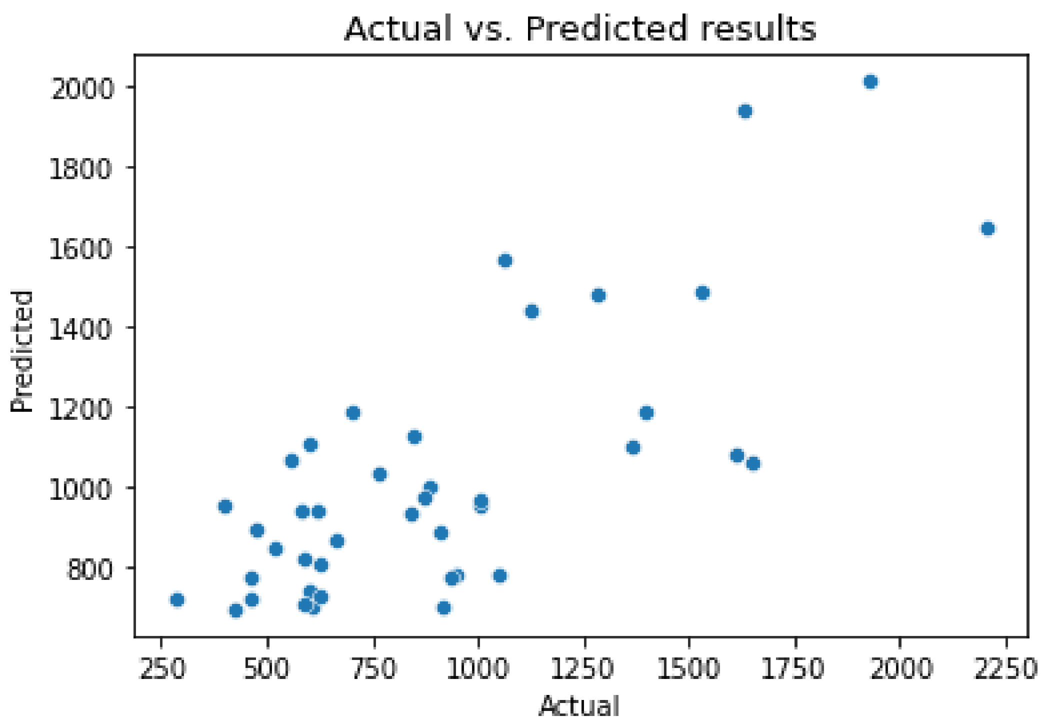
Let's compare the actual vs. predicted values. They don't appear to be that close.

[426]:
```
df_results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df_results.head(10)
```

[426]:

| | Actual | Predicted |
|---|---|---|
| 97 | 1050.743209 | 778.804244 |
| 116 | 282.839523 | 720.168045 |
| 46 | 554.900465 | 1067.746475 |
| 100 | 1064.519115 | 1565.094471 |
| 51 | 1280.614695 | 1483.851545 |
| 29 | 477.299185 | 894.663720 |
| 69 | 461.163450 | 718.755125 |
| 94 | 462.895993 | 775.978403 |
| 9 | 916.681012 | 703.213000 |
| 61 | 1530.105706 | 1485.970926 |

[427]:
```
# Let's examine a scatterplot of the actual vs. predicted values
sns.scatterplot(data=df_results, x = 'Actual', y = 'Predicted')
plt.title("Actual vs. Predicted results", fontsize=13);
```




Let's look at the results of the model. The $R^2$ **value is 0.49**. This is quite good for a social study such as this one.

[428]:
```
print(f"The R-squared value is: \
      {ReturnR2value(dfB,'R2')} \nThe Root MSE is:\t\
      {ReturnR2value(dfB,'MSE')} \nThe Intercept is:\t\
      {ReturnR2value(dfB,'Intercept')}")
```

```
The R-squared value is:       49.09
The Root MSE is:              313.20868156422546
The Intercept is:             1217.1073039109747
```

**Outliers**

In order to improve the model we could use more training data, however as we only have 140 neighborhoods we do not have that much data to begin with. We can also look at outliers.

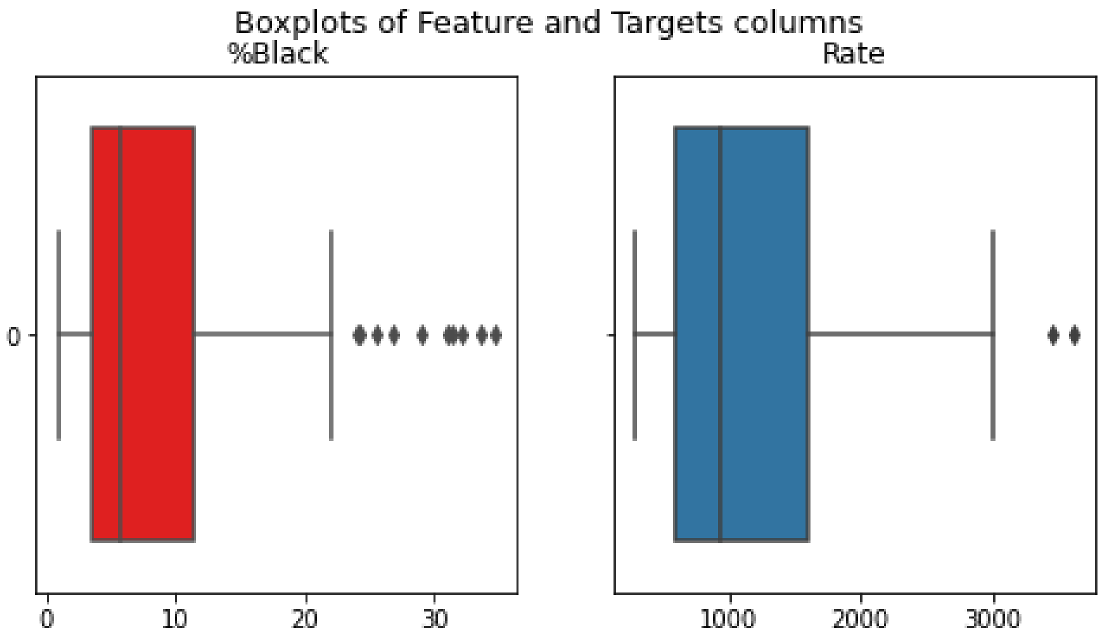
The following boxplots show the outliers for the X (%Black) and y (Rate) columns.

[430]:
```
# Show boxplots for predictor and target
df = NeighCovid[["Black%"]]
df['Rate'] = NeighCovid['Rate']

fig, (ax1, ax2) = plt.subplots(ncols=2, sharey=True, figsize=(8,4))
sns.boxplot(data=df['Black%'], orient="h", ax=ax1, color='r')
```

```
sns.boxplot(data=df['Rate'], orient="h", ax=ax2)

# Add the figure text below
ax1.set_title("%Black")
ax2.set_title("Rate")
plt.suptitle("Boxplots of Feature and Targets columns", fontsize=13);
```




The figure above shows us that there are outliers in both the feature and target columns. Let's remove the outliers and recreate the model and see if there's an improvement.

This code will remove all the rows that are outliers according to IQR. We call shape to determine that 11 rows have been removed.

[432]:
```
df_out = RemoveOutlierDF(df)
```

[433]:
```
df_out.shape
```

[433]:
```
(129, 2)
```

Let's run the model again to see if there's an improvement.

[434]:
```
print(f"The R-squared value is: \
      {ReturnR2value(df_out,'R2')} \nThe Root MSE is:\t\
      {ReturnR2value(df_out,'MSE')} \nThe Intercept is:\t\
      {ReturnR2value(df_out,'Intercept')}")
```

```
The R-squared value is:       36.37
```

```
The Root MSE is:              538.6686246770221
The Intercept is:             1008.2957708286475
```

Let's look at the results of the new model. The $R^2$ **value is now 0.36**.

So removing the outliers actually made things worse.

### Multiple Linear Regression

Now let's see if we can improve the model by using more than one feature. We will use the five features we discussed earlier: %Black, ManufactJob, Bachelor Degree, WorkNights and BothMaintHouse.

[435]:
```
# Create the X and y datasets
X = NeighCovid[['Black%', 'ManufactJob', 'BachelorDegree',␣
 ↪'WorkNights','BothMaintHouse']]
y = NeighCovid["Rate"]
```

Here's the coefficients of the model:

[436]:
```
cvC = ReturnR2value(NeighCovid,'Coefficients')
cvC
```

[436]:

| | Coefficient |
|---|---|
| Black% | 243.434794 |
| ManufactJob | 268.551318 |
| BachelorDegree | 19.125431 |
| WorkNights | 175.770549 |
| BothMaintHouse | -47.114529 |

Here is the intercept:

[437]:
```
ReturnR2value(NeighCovid, 'Intercept')
```

[437]: 1212.2741827886584

Here's our equation.

*Rate = 1212.23 + 243.43Black% + 268.55ManufactJob + 19.13BachelorDegree + 175.77WorkNights + -47.11BothMaintHouse*

Let's compare the actual vs. predicted values. They don't appear to be that bad.

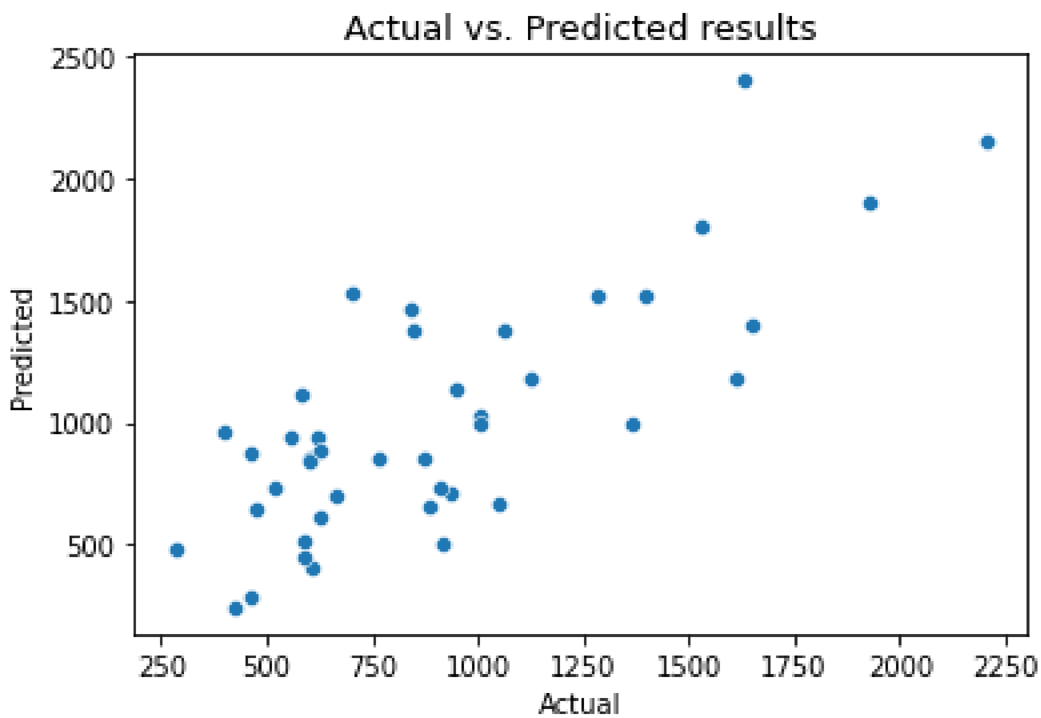
[469]:
```
ReturnR2value(NeighCovid,'ActualVsPred').head(7)
```

[469]:

| | Actual | Predicted |
|---|---|---|
| 97 | 1050.743209 | 669.303910 |
| 116 | 282.839523 | 481.683648 |
| 46 | 554.900465 | 936.515458 |
| 100 | 1064.519115 | 1381.808597 |
| 51 | 1280.614695 | 1516.826511 |

```
29     477.299185   642.361845
69     461.163450   283.858738
```

[470]:
```
# Let's examine a scatterplot of the actual vs. predicted values
sns.scatterplot(data=ReturnR2value(NeighCovid,'ActualVsPred'), \
                x = 'Actual', y = 'Predicted')
plt.title("Actual vs. Predicted results", fontsize=13);
```



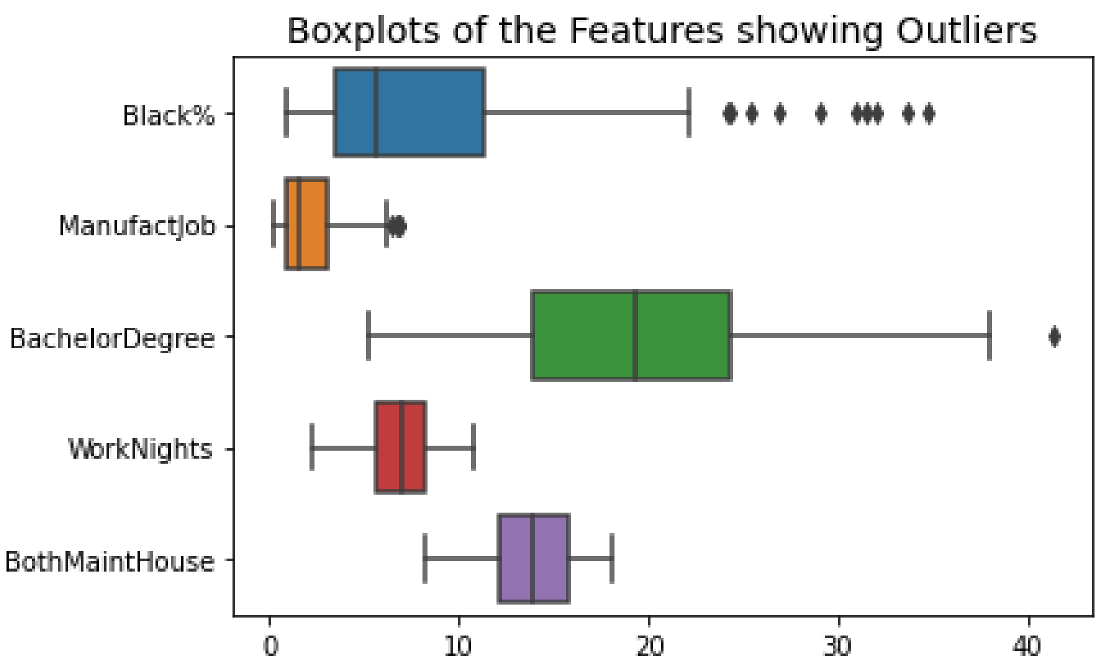

Let's look at the results of the model. The $R^2$ **value is 0.43**. It's lower than simply using one predictor.

[471]:
```
print(f"The R-squared value is: \
      {ReturnR2value(NeighCovid,'R2')} \nThe Root MSE is:\t\
      {ReturnR2value(NeighCovid,'MSE')} \nThe Intercept is:\t\
      {ReturnR2value(NeighCovid,'Intercept')}")
```

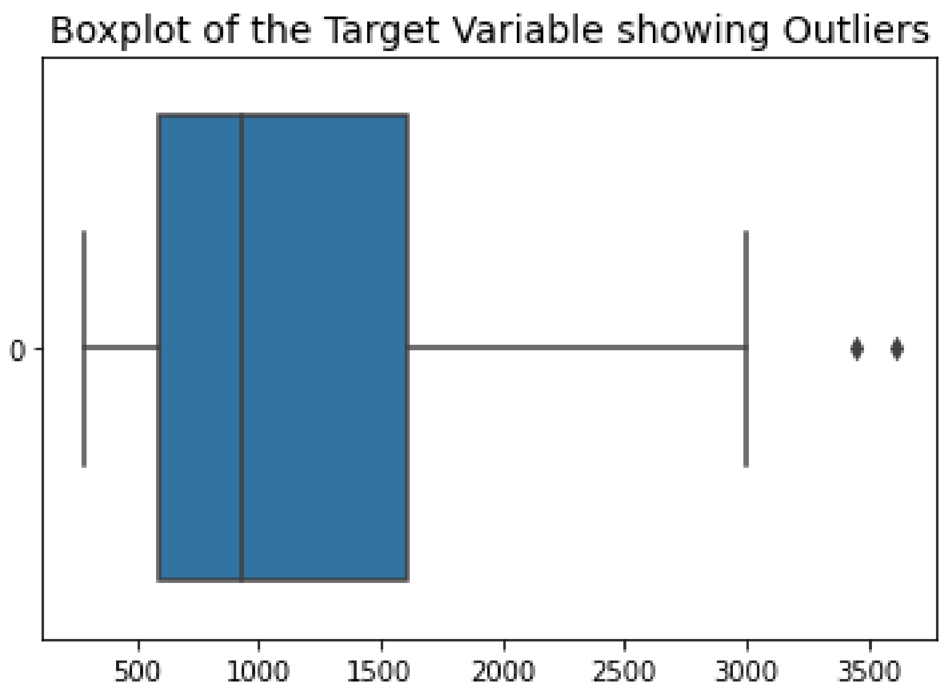
```
The R-squared value is:       43.47
The Root MSE is:              330.03675517892543
The Intercept is:             1212.2741827886584
```

We can try to remove outliers in each column but I suspect it won't help the accuracy of the model. We have already seen that Rate has outliers, the figure below also shows there are a couple outliers in BachelorDegree and ManufactJob.

[472]:
```
# Outliers
sns.boxplot(data=NeighCovid.drop('Rate',axis=1), orient="h");
plt.title("Boxplots of the Features showing Outliers",fontsize=14);
```




[473]:
```
sns.boxplot(data=NeighCovid['Rate'], orient="h")
plt.title("Boxplot of the Target Variable showing Outliers",fontsize=14);
```




[474]:
```
# Let's call the function to remove outliers
newNeighCovid = RemoveOutlierDF(NeighCovid)
```

[476]:
```
print(f"The R-squared value is: \
      {ReturnR2value(newNeighCovid,'R2')} \nThe Root MSE is:\t\
      {ReturnR2value(newNeighCovid,'MSE')} \nThe Intercept is:\t\
      {ReturnR2value(newNeighCovid,'Intercept')}")
```

```
The R-squared value is:       56.82
The Root MSE is:              432.34644674183556
The Intercept is:             1031.6761147565676
```

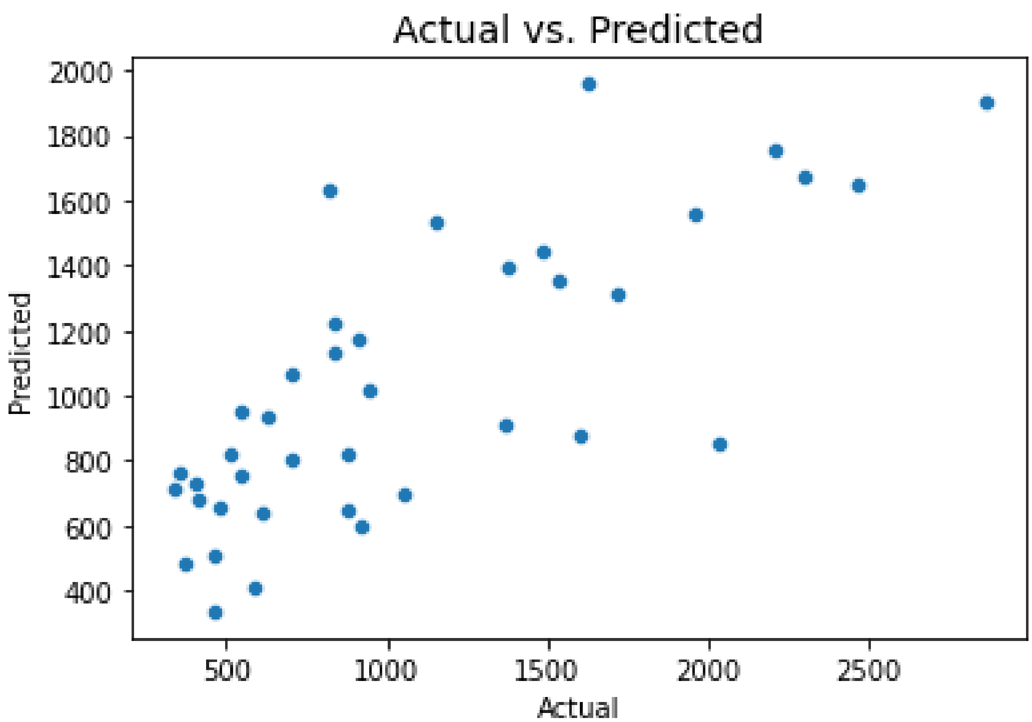
These are very good results as the $R^2$ value has improved to almost **57%**. This is considered very good for a social science study.

[477]:
```
print(f"We removed {NeighCovid.shape[0] - newNeighCovid.shape[0]} outliers from␣
 ↪the dataframe")
```

```
We removed 13 outliers from the dataframe
```

Let's look at the new Predicted vs Actual plot. It's not bad, although it seems to be less accurate for the mid level values.

[478]:
```
sns.scatterplot(data = ReturnR2value(newNeighCovid, 'ActualVsPred'),
                x='Actual', y='Predicted')
plt.title("Actual vs. Predicted", fontsize=14);
```




**Recursive Feature Elimination (RFE)**

Let's try a technique called recursive feature elimination (RFE) and see how it compares to our current model. RFE is part of the sklearn package.

RFE is described as follows:

*Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), the goal of recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features.*

[447]:
```
# Let's perform RFE on our dataset of the top 100 features. This should
# find only the necessary number of features.

# Our feat_cor contain our top 100 features
top100cols = [x for x in feat_cor.iloc[1:,:]['index']]
X = NeighCases[top100cols]
y = NeighCases['Rate per 100,000 people']

from sklearn.linear_model import LinearRegression
from sklearn.feature_selection import RFE
model = LinearRegression()
#Initializing RFE model
rfe = RFE(model, 7)
#Transforming data using RFE
```

```
X_rfe = rfe.fit_transform(X,y)
#Fitting the data to model
model.fit(X_rfe,y)
print(rfe.support_)
print(rfe.ranking_)
```

```
[False False False False False False False 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  True 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 False False False False False False
  True False False False False False False False False False False False
 False False  True False False  True False False False False False False
 False False]
[59 58 61 40 23 78 57 71 41 54 33 17  1 76 25  9  1 69 35 60 85 31 53 84
 56 48 15  5 34 91 26 72 28 37 67 73 39  1 18 52 45 51 32 80  2 10 12 88
 21 36  1 66 16 75 89 87 64 81 24 68 55 50 14 74 62 13 92 79 77  6 44  8
  1 65 43  7 83 38  4 11 63 30 47 22 42 27  1  3 49  1 82 46 19 70 90 20
 86 29]
```

Now let's check what the RFE recommended columns are.

[448]:
```
rfe_cols = [x for x in X.columns[rfe.support_]]
```

[449]:
```
colNames[colNames['index'].isin(rfe_cols)]
```

[449]:
```
         index   _id                      Category  \
330    Col_330   337                      Language
332    Col_332   339                      Language
958    Col_958   984                        Income
964    Col_964   991                        Income
1170  Col_1170  1241  Immigration and citizenship
1190  Col_1190  1261  Immigration and citizenship
1631  Col_1631  1707                     Education

                                            Topic Special  \
330                                 Mother tongue     NaN
332                                 Mother tongue     NaN
958                  Income of individuals in 2015     NaN
964                  Income of individuals in 2015     NaN
1170  Recent immigrants by selected place of birth     NaN
1190  Recent immigrants by selected place of birth     NaN
1631        Highest certificate, diploma or degree     NaN

                                   Characteristic
330                                    Akan (Twi)
332                                           Edo
```

```
958                                     $20,000 to $29,999
964                                     $20,000 to $29,999
1170                     Other places of birth in Americas
1190                                               Nigeria
1631        Certificate of Apprenticeship or Certifi…
```

This is interesting. Three have to do with Education, two with languages, two with income and two with immigration.

Let's check the $R^2$ when we use these columns.

[457]:
```
rfe_df = NeighCases[rfe_cols].join(NeighCases['Rate per 100,000 people'])
```

[458]:
```
ReturnR2value(rfe_df)
```

[458]:
```
44.81
```

The $R^2$ value of this model is almost **45%**. That's pretty good.

A couple features are **highly correlated** so lets remove those, and remove outliers and try again.

[459]:
```
rfe_df.corr()
```

[459]:

| | Col_1170 | Col_1190 | Col_332 | Col_330 | Col_1631 | Col_958 | Col_964 | Rate per 100,000 people |
|---|---|---|---|---|---|---|---|---|
| Col_1170 | 1.000000 | 0.662406 | 0.784227 | 0.683240 | 0.611573 | 0.541691 | 0.541693 | 0.704420 |
| Col_1190 | 0.662406 | 1.000000 | 0.761809 | 0.657477 | 0.334436 | 0.342120 | 0.343919 | 0.684025 |
| Col_332 | 0.784227 | 0.761809 | 1.000000 | 0.870917 | 0.446778 | 0.419054 | 0.413849 | 0.657404 |
| Col_330 | 0.683240 | 0.657477 | 0.870917 | 1.000000 | 0.455120 | 0.380258 | 0.374636 | 0.648345 |
| Col_1631 | 0.611573 | 0.334436 | 0.446778 | 0.455120 | 1.000000 | 0.644684 | 0.662032 | 0.630197 |
| Col_958 | 0.541691 | 0.342120 | 0.419054 | 0.380258 | 0.644684 | 1.000000 | 0.996086 | 0.618513 |
| Col_964 | 0.541693 | 0.343919 | 0.413849 | 0.374636 | 0.662032 | 0.996086 | 1.000000 | 0.617837 |
| Rate per 100,000 people | 0.704420 | 0.684025 | 0.657404 | 0.648345 | 0.630197 | 0.618513 | 0.617837 | 1.000000 |

[460]:
```
# Remove column 964 and 330
rfe_df.drop(["Col_964","Col_330"], axis=1, inplace=True)
```

[461]:
```
# Remove outliers
NewRfeDf = RemoveOutlierDF(rfe_df)
```

[462]:
```
print(f"{rfe_df.shape[0] - NewRfeDf.shape[0]} columns with outliers were␣
↪removed.")
```

```
28 columns with outliers were removed.
```

[463]:
```
ReturnR2value(NewRfeDf)
```

[463]:
```
2.93
```

The $R^2$ has now dropped significantly to 2.93.

#### P-value for Feature Reduction

One way to select predictor variables is using the p-value.

[491]:
```
X = newNeighCovid[['Black%', 'ManufactJob', 'BachelorDegree', 'WorkNights',␣
↪'BothMaintHouse']]
y = newNeighCovid['Rate']
```

[492]:
```
import statsmodels.api as sm
OLS_regressor = sm.OLS(y,X)
OLS_regressor.fit().summary()
```

[492]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                   Rate   R-squared (uncentered):
0.892
Model:                            OLS   Adj. R-squared (uncentered):
0.888
Method:                 Least Squares   F-statistic:
202.6
Date:                Mon, 23 Nov 2020   Prob (F-statistic):
2.59e-57
Time:                        15:20:31   Log-Likelihood:
-938.92
No. Observations:                 127   AIC:
1888.
Df Residuals:                     122   BIC:
1902.
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025
0.975]
```

```
--------------------------------------------------------------------------------
--
Black%             34.9662     11.486      3.044      0.003      12.228
57.704
ManufactJob       132.2317     38.480      3.436      0.001      56.057
208.406
BachelorDegree      7.6066      8.356      0.910      0.364      -8.934
24.147
WorkNights         95.0617     28.633      3.320      0.001      38.379
151.744
BothMaintHouse   -16.7414     16.746     -1.000      0.319     -49.892
16.409
==============================================================================
Omnibus:                       20.608   Durbin-Watson:                   1.861
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               27.621
Skew:                           0.874   Prob(JB):                     1.01e-06
Kurtosis:                       4.472   Cond. No.                         31.2
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

The p-values indicate that there are potentially two features that do not contribute to the model. Let's remove one at a time and check the results.

[493]:
```python
X = newNeighCovid[['Black%', 'ManufactJob', 'WorkNights', 'BothMaintHouse']]
y = newNeighCovid['Rate']
import statsmodels.api as sm
OLS_regressor = sm.OLS(y,X)
OLS_regressor.fit().summary()
```

[493]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                                 OLS Regression Results
================================================================================
=======
Dep. Variable:                   Rate   R-squared (uncentered):
0.892
Model:                            OLS   Adj. R-squared (uncentered):
0.888
Method:                 Least Squares   F-statistic:
253.3
Date:                Mon, 23 Nov 2020   Prob (F-statistic):
```

```
2.30e-58
Time:                        15:21:27   Log-Likelihood:
-939.35
No. Observations:                 127   AIC:
1887.
Df Residuals:                     123   BIC:
1898.
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
==
                     coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
--
Black%            33.9600     11.425      2.972      0.004      11.345
56.575
ManufactJob      122.1344     36.821      3.317      0.001      49.249
195.019
WorkNights        94.1754     28.597      3.293      0.001      37.570
150.781
BothMaintHouse    -3.8738      8.974     -0.432      0.667     -21.637
13.890
==============================================================================
Omnibus:                       16.975   Durbin-Watson:                   1.847
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               20.380
Skew:                           0.800   Prob(JB):                     3.75e-05
Kurtosis:                       4.137   Cond. No.                         19.8
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Now there's one feature left with a high p-value, let's remove it.

[494]:
```python
X = newNeighCovid[['Black%', 'ManufactJob', 'WorkNights']]
y = newNeighCovid['Rate']
import statsmodels.api as sm
OLS_regressor = sm.OLS(y,X)
OLS_regressor.fit().summary()
```

[494]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
```

```
                            OLS Regression Results
================================================================================
=======
Dep. Variable:                   Rate   R-squared (uncentered):
0.892
Model:                            OLS   Adj. R-squared (uncentered):
0.889
Method:                 Least Squares   F-statistic:
340.0
Date:                Mon, 23 Nov 2020   Prob (F-statistic):
1.26e-59
Time:                        15:21:38   Log-Likelihood:
-939.44
No. Observations:                 127   AIC:
1885.
Df Residuals:                     124   BIC:
1893.
Df Model:                           3
Covariance Type:            nonrobust
===============================================================================
                  coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------
Black%         36.0324     10.333      3.487      0.001      15.580      56.484
ManufactJob   126.8286     35.063      3.617      0.000      57.429     196.228
WorkNights     83.0461     12.330      6.735      0.000      58.641     107.452
==============================================================================
Omnibus:                       16.559   Durbin-Watson:                   1.856
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               19.492
Skew:                           0.798   Prob(JB):                     5.85e-05
Kurtosis:                       4.067   Cond. No.                         11.0
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not
contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Now all our features are significant. Let's check the $R^2$ value and the MSE of the new model.

[495]:
```
newestNeighCovid = newNeighCovid[['Black%', 'ManufactJob', 'WorkNights',␣
↪'Rate']]

print(f"The R-squared value is: \
      {ReturnR2value(newestNeighCovid,'R2')} \nThe Root MSE is:\t\
      {ReturnR2value(newestNeighCovid,'MSE')} \nThe Intercept is:\t\
```

```
    {ReturnR2value(newestNeighCovid,'Intercept')}")
```

```
The R-squared value is:       60.52
The Root MSE is:              413.42407505558594
The Intercept is:             1038.1824194302878
```

The new coefficients are:

[498]:
```
ReturnR2value(newestNeighCovid, 'Coefficients')
```

[498]:

| | Coefficient |
|---|---|
| Black% | 157.692253 |
| ManufactJob | 140.833170 |
| WorkNights | 170.074059 |

Great news, we have simplified the model and increased the $R^2$ value to **over 60%**.

This could be considered a very good $R^2$ value for a social sciences type of study as this one.[6]

Our final model is:

*Rate = 1038 + 157.7 Black% + 140.8 ManufactJob + 170.1 WorkNights*

## Conclusions

This study used multiple linear regression to predict covid rates in Toronto's 140 neighborhoods using demographic data from Stats Canada. The data initially contained over 2,300 potential predictors. The correlation of these features to the target variable were analyzed in order to limit the number of potential predictors. Features which were highly correlated to each other were removed and an analysis of the p-values were done recursively to eliminate further predictors. Using the IQR method a few neighborhoods were removed that were considered outliers. The final model used only three variables to predict covid rates in the neighborhoods with an $R^2$ value of 0.605.

Specifically these variables in the Stat Canada demographic database are know as *Black (Visible Minority)*, *Occupations in manufacturing and utilities*, and *Journey to work Between 12 p.m. and 4:59 a.m.*. It has been suggested that certain racialized communities are affected by covid for various reasons including a higher proportion with underlying health conditions such as hypertension, the increased likelihood of living in more densely populated neighborhoods, and the lower proportion that can work from home.[3]

Further research is warranted and could include a larger analysis of neighborhoods across Canada, or in other countries. A model such as the one built for this study could help to target resources where they are most needed, or to preemptively determine where the most severe outbreaks will occur in order to take preventative action.

## References


1. Www12.statcan.gc.ca. 2020. 2016 Census Of Population – Data Products. [online] Available at: <https:www12.statcan.gc.cacensus-recensement2016dp-pdindex-eng.cfm> [Accessed 22 November 2020].

2. Open.toronto.ca. 2020. Homepage. [online] Available at: <https://open.toronto.ca/> [Accessed 22 November 2020].

3. Shah, M., Sachdeva, M. and Dodiuk-Gad, R.P., 2020. COVID-19 and racial disparities. Journal of the American Academy of Dermatology, 83(1), p.e35.

4. Teofilo, R.F., Martins, J.P.A. and Ferreira, M.M., 2009. Sorting variables by using informative vectors as a strategy for feature selection in multivariate regression. Journal of Chemometrics: A Journal of the Chemometrics Society, 23(1), pp.32-48.

5. Wikipedia. (n.d.). Linear Regression. Retrieved November 23, 2020, from https://en.wikipedia.org/wiki/Linear_regression

6. Grace-Martin, K., 2012. Can a Regression Model with a Small R-squared Be Useful?. The Analysis Factor.[siteerattu 29.10. 2016]. Saatavana World Wide Webistä:< URL: http://www.theanalysisfactor. com/small-rsquared.

# Appendix A

## Functions

[145]:
```
def ReturnR2value(df, choice_str='R2'):
    '''Accepts a dataframe returns R2 value, MSE, Intercept or Coefficients
    Depending on Choice'''
    # Author: Alexei Marcilio
    # Date: Nov 20, 2020
    # Ver 1.0
    # We assume the last column is the target
    X = df.iloc[:,0:-1]
    y = df.iloc[:,-1]
    # Scale and fit
    sc = StandardScaler()
    X = sc.fit_transform(X)
    # Split data into train and test
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3,␣
↪random_state = 11)
    # Create and train the model
    from sklearn.linear_model import LinearRegression
    #Create the model :
    regressor = LinearRegression()
    #Train the model :
    regressor.fit(X_train, y_train)
    # Predict
    y_pred = regressor.predict(X_test)
    from sklearn.metrics import mean_squared_error , r2_score
    mse = mean_squared_error(y_test, y_pred)

    # Root Mean Squared Error:
    root_mse = np.sqrt(mse)

    coeff_X = pd.DataFrame(regressor.coef_, index=df.columns[:-1],␣
↪columns=['Coefficient'])

    df_ActPred = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

    #R_squared :
    if choice_str == 'R2':
        return round(r2_score(y_test, y_pred)*100,2)
    elif choice_str == 'MSE':
        return root_mse
    elif choice_str == 'Intercept':
        return regressor.intercept_
    elif choice_str == 'Coefficients':
        return coeff_X
```

```python
    elif choice_str == 'ActualVsPred':
        return df_ActPred
```

[146]:
```python
def RemoveOutlierDF(df):
    '''Accepts a dataframe returns a dataframe with all outliers based
    on IQR removed'''
    # Author: Alexei Marcilio
    # Date: Nov 20, 2020
    # Ver 1.0
    # Function takes a dataframe and removes all outliers
    # based in IQR
    # returns a new df
    from scipy import stats
    # IQR
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    return df[~((df < (Q1 - 1.5 * IQR)) |(df > (Q3 + 1.5 * IQR))).any(axis=1)]
```

[ ]: