

# infectious\_disease

April 2, 2019

Infectious Diseases Data Analysis

Cleaning Process

Following are the steps we followed for data analysis

## 1. Import the libraries

```
In [1]: #Import the Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

%matplotlib inline
# Algorithms
from sklearn import linear_model
from sklearn.linear_model import LinearRegression

import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

## 2. Load the dataset.

```
In [2]: dataset=pd.read_csv('data/data.csv')
```

```
In [3]: dataset.head(3)
```

```
Out[3]:
```

	Indicator Category \					
0	Behavioral Health/Substance Abuse					
1	Behavioral Health/Substance Abuse					
2	Behavioral Health/Substance Abuse					

	Indicator	Year	Sex \			
0	Opioid-Related Unintentional Drug Overdose Mor...	2010	Both			
1	Opioid-Related Unintentional Drug Overdose Mor...	2010	Both			
2	Opioid-Related Unintentional Drug Overdose Mor...	2010	Both			

	Race/Ethnicity	Value	Place \			
0	All	1.7	Washington, DC			

1	All	2.2	Fort Worth (Tarrant County), TX
2	All	2.3	Oakland (Alameda County), CA

	BCHC Requested Methodology \
0	Age-Adjusted rate of opioid-related mortality ...
1	Age-adjusted rate of opioid-related mortality ...
2	Age-adjusted rate of opioid-related mortality ...

	Source \
0	D.C. Department of Health, Center for Policy, ...
1	National Center for Health Statistics
2	CDC Wonder

	Methods \
0	NaN
1	NaN
2	Age-adjusted rate of opioid-related mortality ...

	Notes \
0	This indicator is not exclusive of other drugs...
1	This indicator is not exclusive of other drugs...
2	Data is for Alameda County. This indicator is ...

	90% Confidence Level - Low	90% Confidence Level - High \
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN

	95% Confidence Level - Low	95% Confidence Level - High
0	NaN	NaN
1	1.5	3.0
2	1.6	3.2

Above we saw the column names and we might need to fix the spaces in the column names. In order to change that we need to first know what are the actual names of the columns.

We do that using the pandas function `columns` to list all the columns

```
In [4]: dataset.columns
```

```
Out[4]: Index(['Indicator Category', 'Indicator', 'Year', 'Sex', 'Race/Ethnicity',
              'Value', 'Place', 'BCHC Requested Methodology', 'Source', 'Methods',
              'Notes', '90% Confidence Level - Low', '90% Confidence Level - High',
              '95% Confidence Level - Low', '95% Confidence Level - High'],
              dtype='object')
```

Now we rename the columns

```
In [5]: dataset.rename(columns={'Indicator Category': 'indicator_category', 'Indicator': 'indicator',
                              'Value': 'value', 'Place': 'place', 'BCHC Requested Methodology': 'bchc_req_meth',
```

```
'Notes':'notes', '90% Confidence Level - Low':'90pc_con_lvl-low', '90% Confiden
'95% Confidence Level - Low':'95pc_con_lvl-low', '95% Confidence Level - High':'9
```

3.Now we need to filter the data according to the indicator category. We use one of the values "Cancer".

```
In [6]: infectious_ds = dataset.loc[dataset["indicator_category"] == "Infectious Disease"]
```

4.And then we remove empty columns and unnecessary columns

```
In [7]: infectious_ds.drop(['indicator_category', 'bchc_req_meth', 'source', 'methods', 'notes', '9
axis = 1, inplace= True)
```

5. Now we remove all the rows which has NaN or NA values

```
In [8]: infectious_ds.dropna(axis=0, how='any',inplace= True)
```

```
In [9]: infectious_ds.to_csv("data/infectious_diseases.csv")
```

```
In [10]: infectious_ds.head(3)
```

```
Out[10]:
```

			indicator	year	sex	\
18078	Percent of Adults 65 and Over Who Received Pne...			2010	Both	
18081	Percent of Adults 65 and Over Who Received Pne...			2010	Both	
18084	Percent of Adults 65 and Over Who Received Pne...			2010	Both	

	race_ethnicity	value	place	95pc_con_lvl-low	\
18078	All	59.9	Charlotte, NC	51.3	
18081	All	73.0	Seattle, WA	65.0	
18084	Asian/PI	72.0	Seattle, WA	37.0	

	95pc_con_lvl-high
18078	68.5
18081	80.0
18084	91.0

Analysis

First we'll see how many patients have been reported for cancer in respective years from 2010 to 2016.

Following is the process to do the same

```
In [11]: c_year_2010=infectious_ds[infectious_ds['year']==2010]
c_year_2010_count=c_year_2010['year'].count()
```

```
In [12]: c_year_2010.shape
```

```
Out[12]: (92, 8)
```

```

In [13]: c_year_2011=infectious_ds[infectious_ds['year']==2011]
         c_year_2011_count=c_year_2011['year'].count()

         c_year_2012=infectious_ds[infectious_ds['year']==2012]
         c_year_2012_count=c_year_2012['year'].count()

         c_year_2013=infectious_ds[infectious_ds['year']==2013]
         c_year_2013_count=c_year_2013['year'].count()

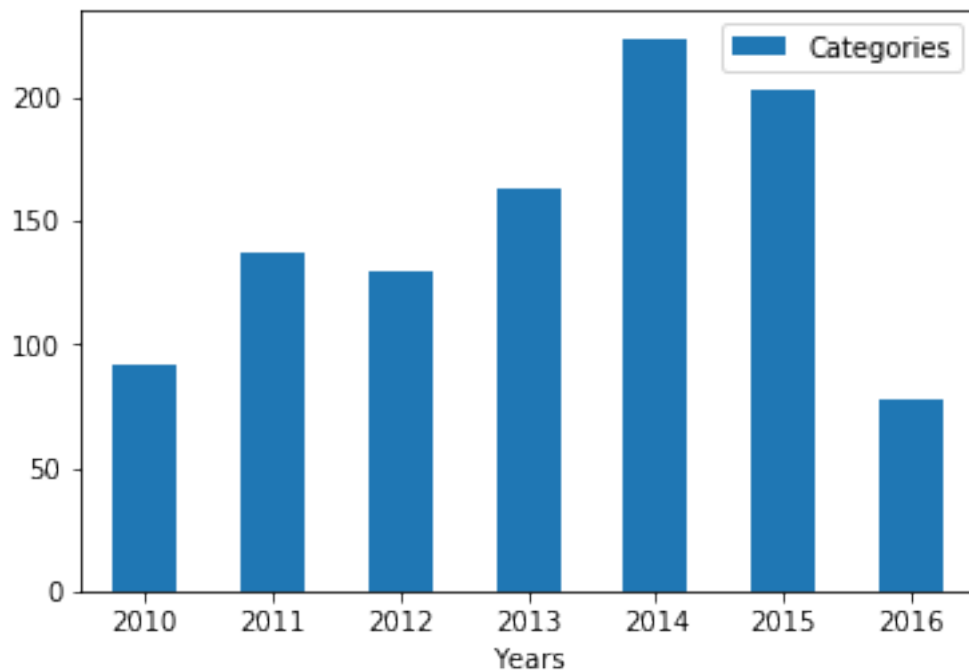
         c_year_2014=infectious_ds[infectious_ds['year']==2014]
         c_year_2014_count=c_year_2014['year'].count()

         c_year_2015=infectious_ds[infectious_ds['year']==2015]
         c_year_2015_count=c_year_2015['year'].count()

         c_year_2016=infectious_ds[infectious_ds['year']==2016]
         c_year_2016_count=c_year_2016['year'].count()

In [14]: fig1 = pd.DataFrame({'Years':['2010', '2011', '2012','2013','2014','2015','2016'], 'C
         ax = fig1.plot.bar(x='Years', rot=0)

```



Now we calculate the number of cases for each type of cancer. In order to that we will group according to the indicator and take the count.

```

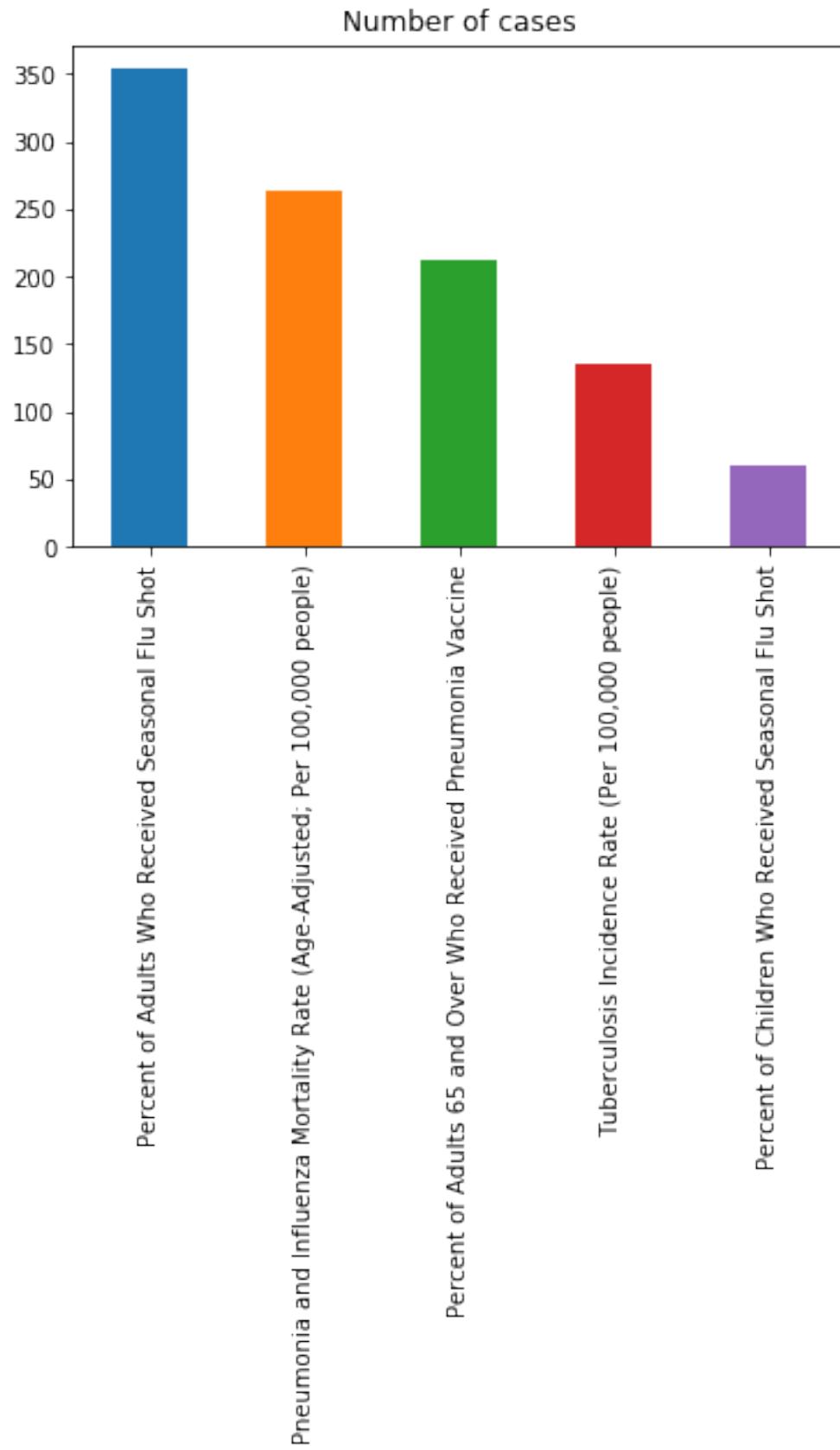
In [15]: sorted_infection = infectious_ds['indicator'].value_counts()
         sorted_infection

```

```
Out[15]: Percent of Adults Who Received Seasonal Flu Shot          354
        Pneumonia and Influenza Mortality Rate (Age-Adjusted; Per 100,000 people) 264
        Percent of Adults 65 and Over Who Received Pneumonia Vaccine          213
        Tuberculosis Incidence Rate (Per 100,000 people)          136
        Percent of Children Who Received Seasonal Flu Shot          60
        Name: indicator, dtype: int64
```

And we plot a histogram to see.

```
In [16]: labels=list(infectious_ds.columns)
        sorted_infection = infectious_ds['indicator'].value_counts().plot(title='Number of cases by indicator')
        plt.show()
        #label=list(group.columns)
```



Now we find out the distribution of cancer patients with respect to the race and ethnicity.

```
In [17]: all=infectious_ds[infectious_ds['race_ethnicity']=="All"]
all_count=all.race_ethnicity.count()

asian=infectious_ds[infectious_ds['race_ethnicity']=="Asian/PI"]
asian_count=asian.race_ethnicity.count()

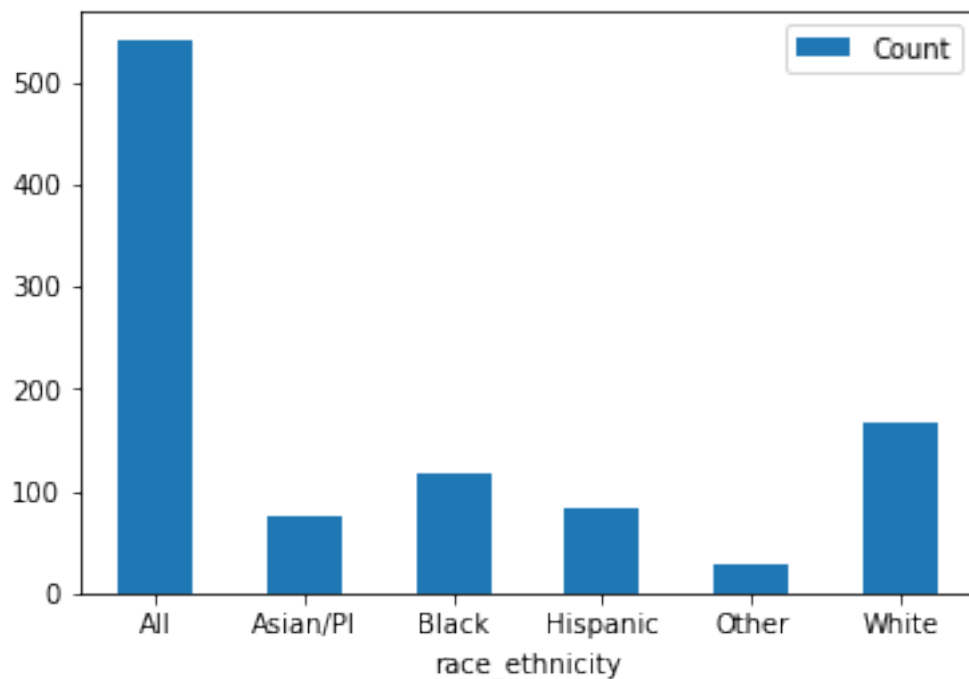
black=infectious_ds[infectious_ds['race_ethnicity']=="Black"]
black_count=black.race_ethnicity.count()

hispanic=infectious_ds[infectious_ds['race_ethnicity']=="Hispanic"]
hispanic_count=hispanic.race_ethnicity.count()

other=infectious_ds[infectious_ds['race_ethnicity']=="Other"]
other_count=other.race_ethnicity.count()

white=infectious_ds[infectious_ds['race_ethnicity']=="White"]
white_count=white.race_ethnicity.count()

In [18]: fig2 = pd.DataFrame({'race_ethnicity':['All', 'Asian/PI', 'Black', 'Hispanic', 'Other', 'White'],
                             'Count':[all_count, asian_count, black_count, hispanic_count, other_count, white_count]})
ax = fig2.plot.bar(x='race_ethnicity', rot=0)
```



```
In [19]: infectious_ds=infectious_ds.rename(columns={'95pc_con_lvl-low':'low','95pc_con_lvl-high':'high'})

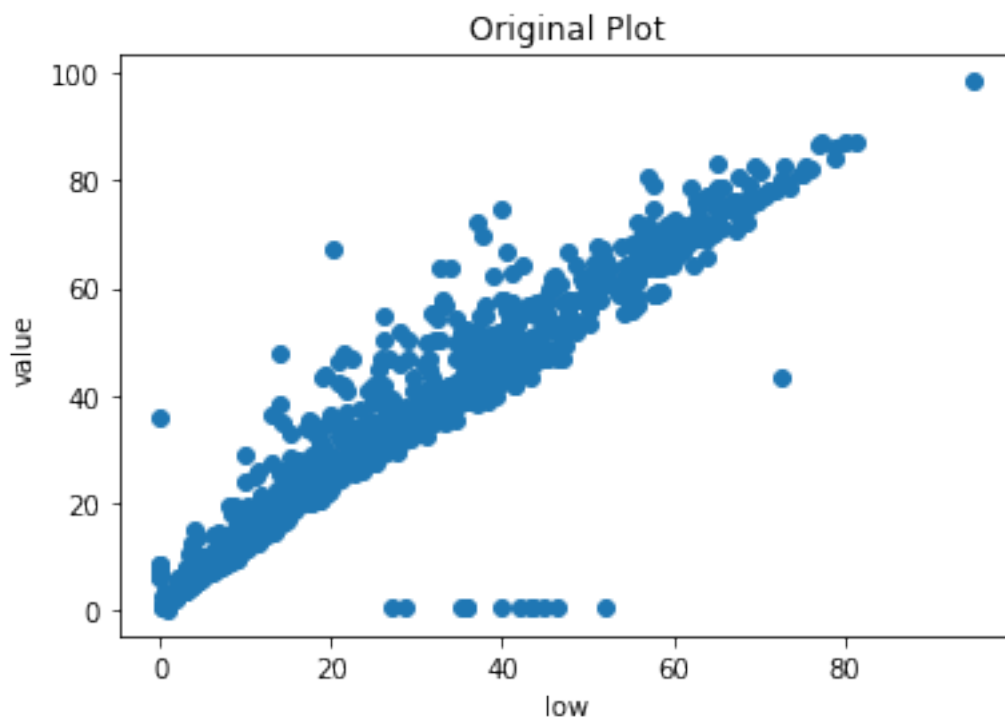
In [20]: ds=infectious_ds.drop(['indicator','year','sex','race_ethnicity',
                                'place'],
                                axis = 1)
```

```
In [21]: ds.head()
```

```
Out[21]:
```

	value	low	high
18078	59.9	51.3	68.5
18081	73.0	65.0	80.0
18084	72.0	37.0	91.0
18087	54.2	34.5	74.4
18088	55.0	26.0	81.0

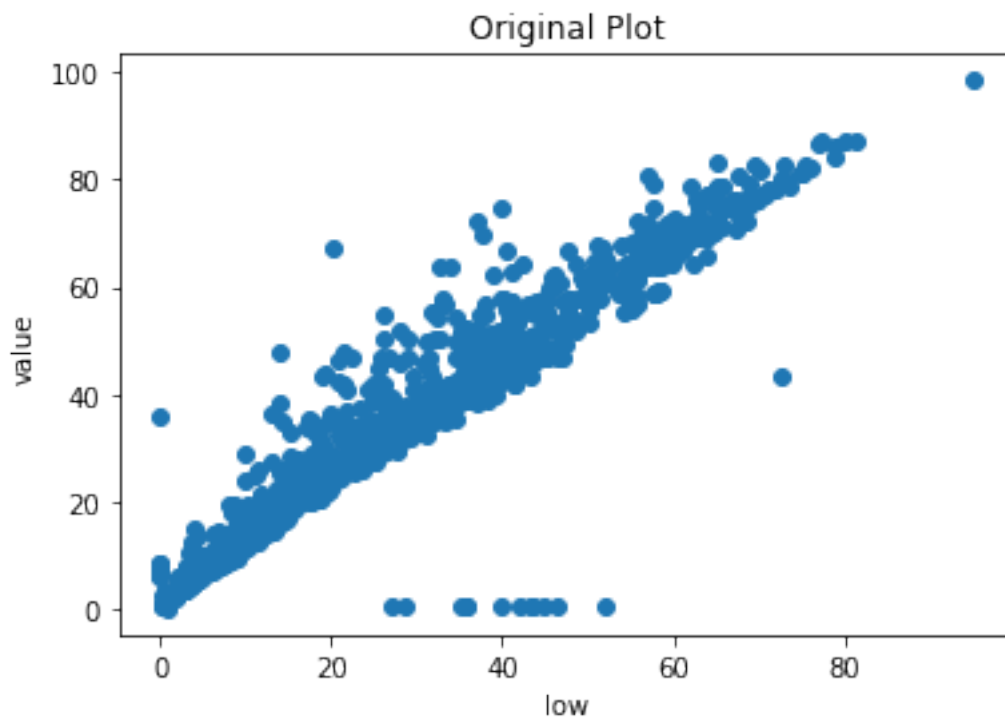
```
In [22]: #very simple plotting
fig = plt.figure(1)
ax1 = fig.add_subplot(111)
ax1.set_xlabel('low')
ax1.set_ylabel('value')
ax1.set_title('Original Plot')
ax1.scatter('low', 'value', data = ds);
```



Linear Regression with high values



```
In [23]: #very simple plotting
fig = plt.figure(1)
ax1 = fig.add_subplot(111)
ax1.set_xlabel('low')
ax1.set_ylabel('value')
ax1.set_title('Original Plot')
ax1.scatter('low', 'value', data = ds);
```



```
In [24]: x_y = np.array(ds)
x, y = x_y[:,0], x_y[:,1]

# Reshaping
x, y = x.reshape(-1,1), y.reshape(-1, 1)

# Linear Regression Object
lin_regression = LinearRegression()

# Fitting linear model to the data
lin_regression.fit(x,y)

# Get slope of fitted line
m = lin_regression.coef_

# Get y-Intercept of the Line
```

```

b = lin_regression.intercept_

# Get Predictions for original x values
# you can also get predictions for new data
predictions = lin_regression.predict(x)

# following slope intercept form
print ("formula: y = {0}x + {1}".format(m, b) )

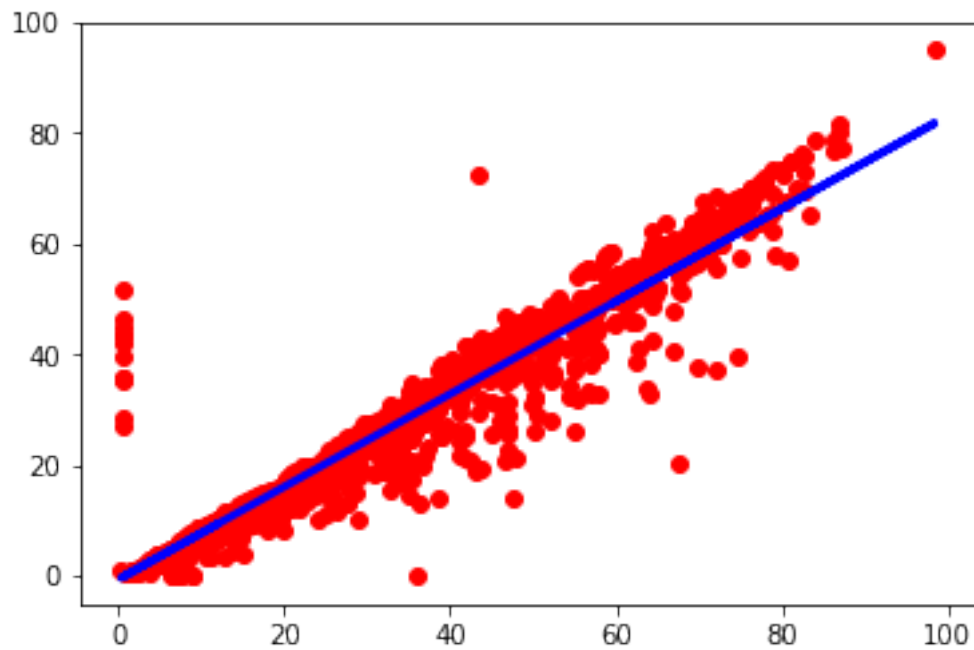
```

formula: y = [[0.83687858]]x + [-0.64247836]

```

In [25]: plt.scatter(x, y, color='red')
plt.plot(x, predictions, color='blue',linewidth=3)
plt.show()

```



In [ ]: