injury_violence

April 2, 2019

Injury and Violence 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':'indicat
               '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':'
```

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

```
In [6]: iv_ds = dataset.loc[dataset["indicator_category"] == "Injury/Violence"]
```

4.And then we remove empty columns and unnecessary columns

```
In [7]: iv_ds.drop(['indicator_category','bchc_req_meth','source','methods','notes','90pc_con_
                             axis = 1, inplace= True)
```

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

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

```
In [9]: iv_ds.to_csv("data/injury_violence.csv")
```

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

```
Out[10]:                                               indicator  year   sex  \
         21158  Firearm-Related Emergency Department Visit Rat...  2010  Both
         21161  Firearm-Related Emergency Department Visit Rat...  2010  Both
         21163  Firearm-Related Emergency Department Visit Rat...  2010  Both

               race_ethnicity  value                          place  95pc_con_lvl-low  \
         21158            All    1.2  Las Vegas (Clark County), NV               1.0
         21161            All    4.7                    Chicago, Il               4.5
         21163       Asian/PI    0.2  Las Vegas (Clark County), NV               0.0

                95pc_con_lvl-high
         21158                1.3
         21161                5.0
         21163                0.3
```

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

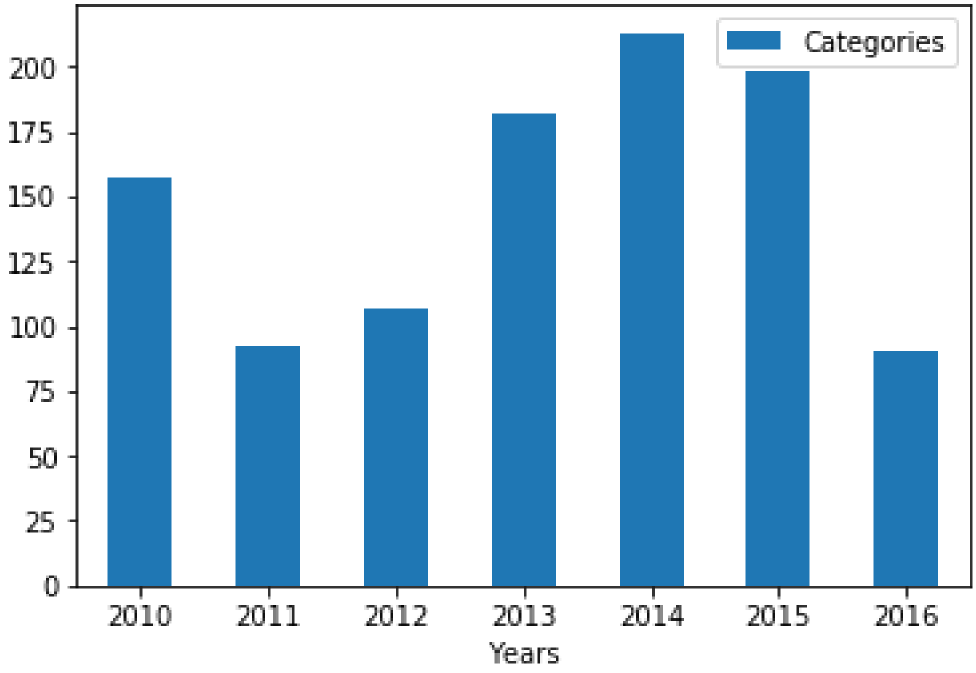
```
In [11]: c_year_2010=iv_ds[iv_ds['year']==2010]
         c_year_2010_count=c_year_2010['year'].count()
```

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

```
Out[12]: (157, 8)
```

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

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

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

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

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

         c_year_2016=iv_ds[iv_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'], 'Ca
         ax = fig1.plot.bar(x='Years', rot=0)
```



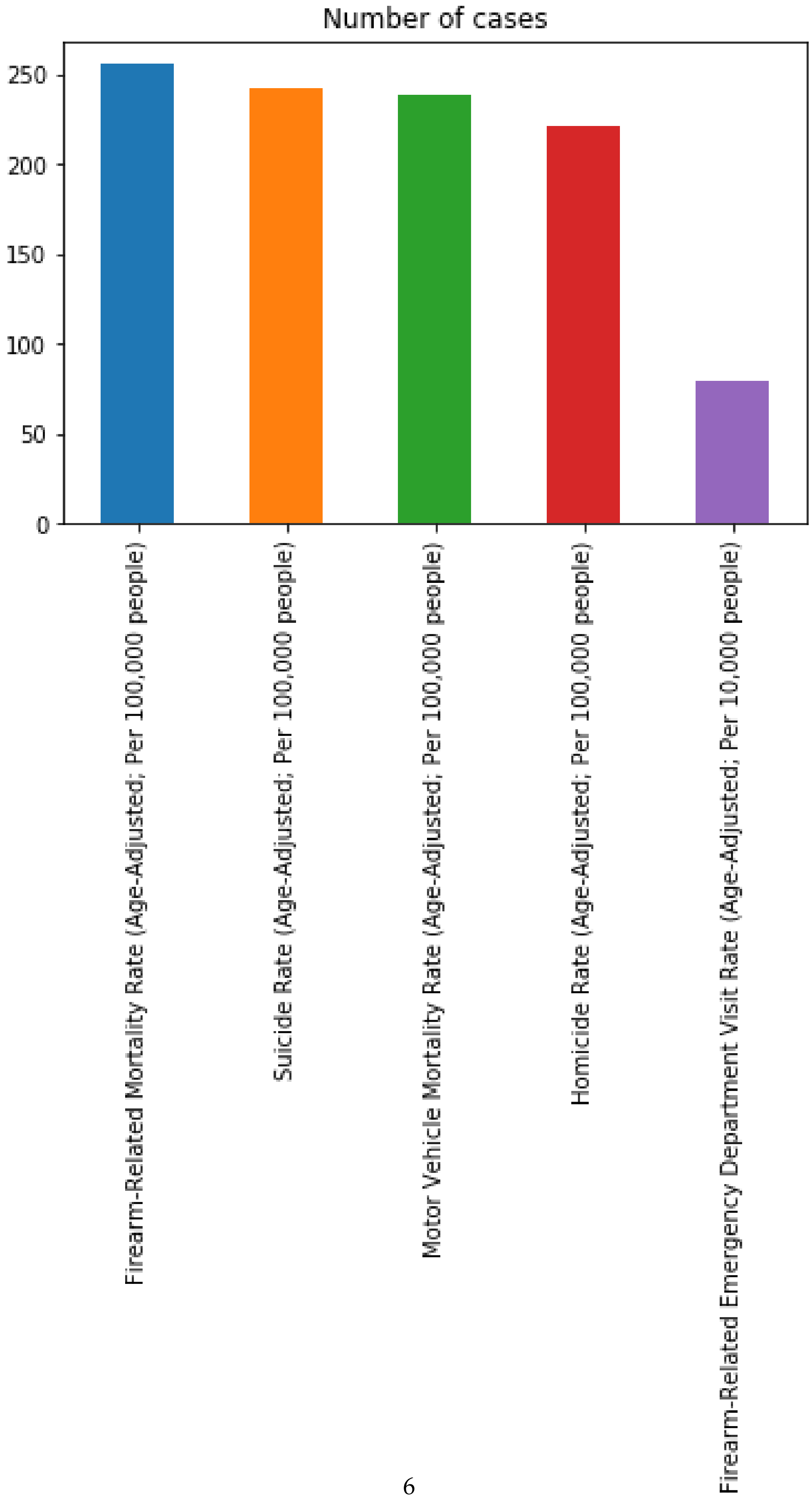
Now we calculate the number of cases for each type of cancer. In order to that we will group according to the indicator and take the count.

```
In [15]: sorted_injury = iv_ds['indicator'].value_counts()
         sorted_injury
```

```
Out[15]: Firearm-Related Mortality Rate (Age-Adjusted; Per 100,000 people)                    2
         Suicide Rate (Age-Adjusted; Per 100,000 people)                                     2
         Motor Vehicle Mortality Rate (Age-Adjusted; Per 100,000 people)                      2
         Homicide Rate (Age-Adjusted; Per 100,000 people)                                    2
         Firearm-Related Emergency Department Visit Rate (Age-Adjusted; Per 10,000 people)
         Name: indicator, dtype: int64
```

And we plot a histogram to see.

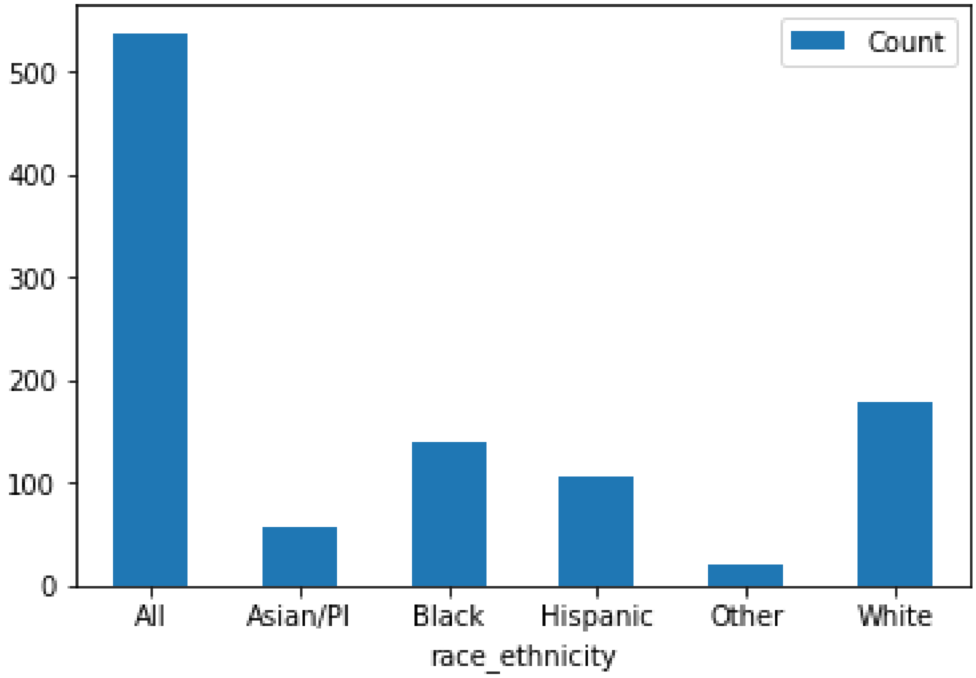
```
In [16]: labels=list(iv_ds.columns)
         sorted_injury = iv_ds['indicator'].value_counts().plot(title='Number of cases', kind=
         plt.show()
         #label=list(group.columns)
```


Number of cases
250
200
150
100
50
0
Firearm-Related Mortality Rate (Age-Adjusted; Per 100,000 people)
Suicide Rate (Age-Adjusted; Per 100,000 people)
Motor Vehicle Mortality Rate (Age-Adjusted; Per 100,000 people)
Homicide Rate (Age-Adjusted; Per 100,000 people)
Firearm-Related Emergency Department Visit Rate (Age-Adjusted; Per 10,000 people)

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

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

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

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

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

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

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

```
In [18]: fig2 = pd.DataFrame({'race_ethnicity':['All', 'Asian/PI', 'Black','Hispanic','Other',
         ax = fig2.plot.bar(x='race_ethnicity', rot=0)
```



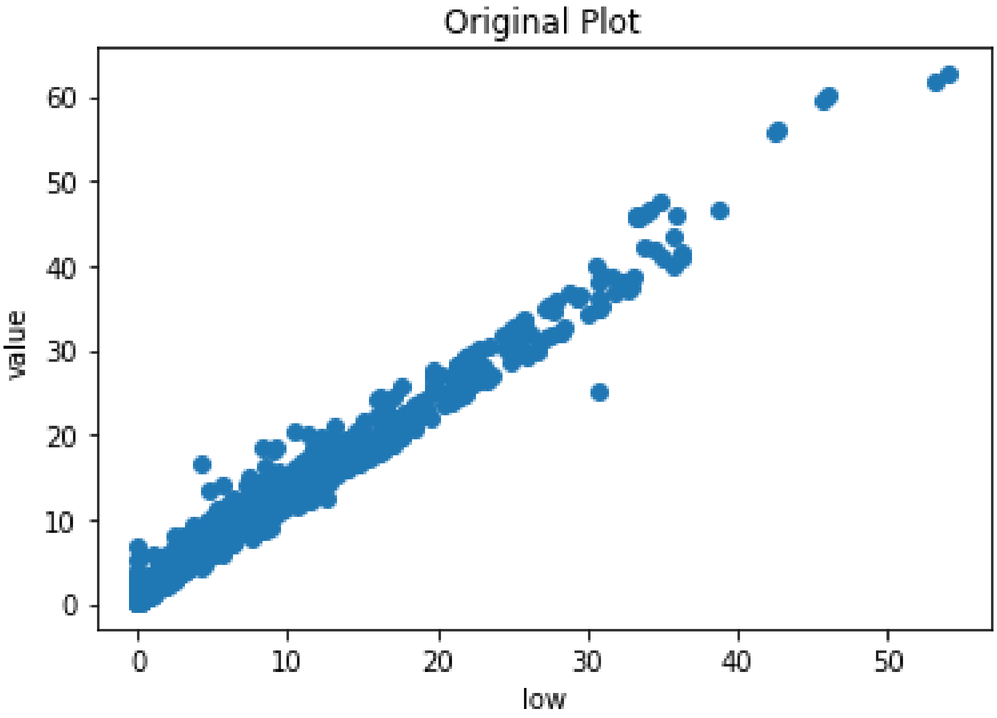
```
In [19]: iv_ds=iv_ds.rename(columns={'95pc_con_lvl-low':'low','95pc_con_lvl-high':'high'})
```

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

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

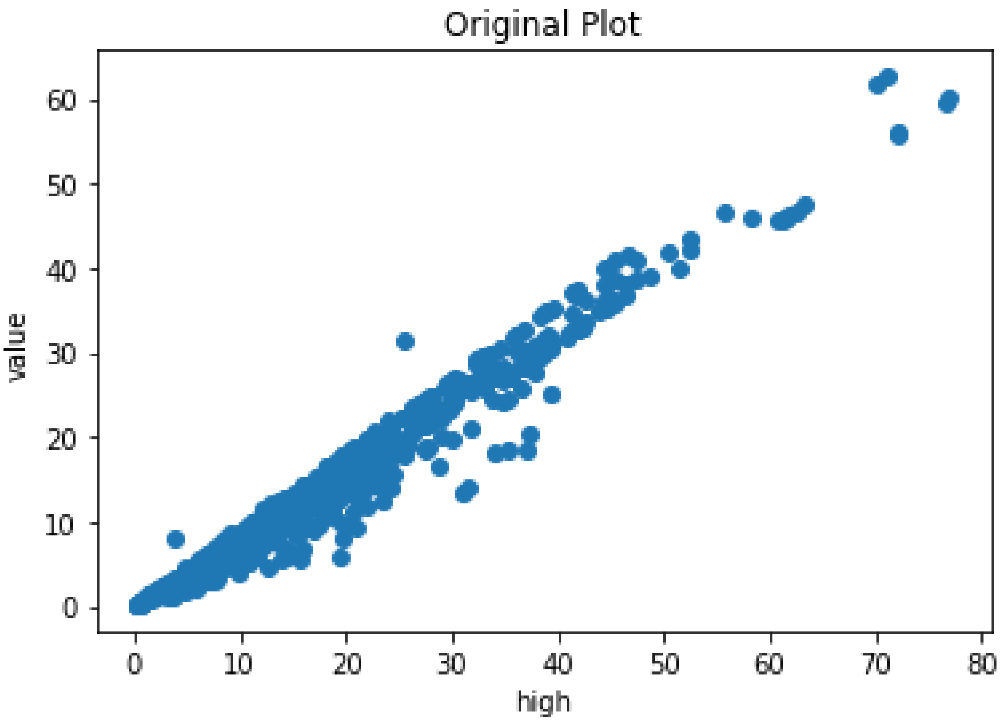
```
Out[21]:        value  low  high
         21158    1.2  1.0   1.3
         21161    4.7  4.5   5.0
         21163    0.2  0.0   0.3
         21164    0.3  0.0   0.6
         21166    3.1  2.4   3.8
```

```
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

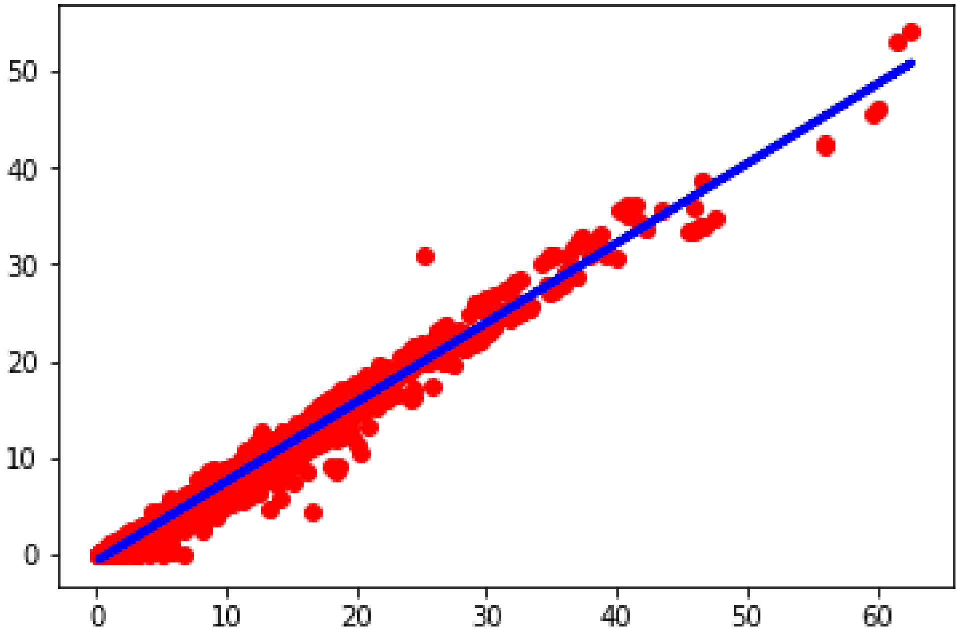
```
In [28]: #very simple plotting
         fig = plt.figure(1)
         ax1 = fig.add_subplot(111)
         ax1.set_xlabel('high')
         ax1.set_ylabel('value')
         ax1.set_title('Original Plot')
         ax1.scatter('high', 'value', data = ds);
```



```
In [29]: 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 = [[0.8238202]]x + [-0.75984772]
```

```
In [30]: plt.scatter(x, y,  color='red')
         plt.plot(x, predictions, color='blue',linewidth=3)
         plt.show()
```



```
In [ ]:
```