life_death_expectancy

April 2, 2019

Life and Death Expectancy 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% Confidenc
                '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]: lde_ds = dataset.loc[dataset["indicator_category"] == "Life Expectancy and Death Rate
```

4.And then we remove empty columns and unnecessary columns

```
In [7]: lde_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]: lde_ds.dropna(axis=0, how='any',inplace= True)

In [9]: lde_ds.to_csv("data/life_death.csv")

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

Out[10]:

	indicator	year	sex	\
24934	All-Cause Mortality Rate (Age-Adjusted; Per 10...	2010	Both	
24935	All-Cause Mortality Rate (Age-Adjusted; Per 10...	2010	Both	
24936	All-Cause Mortality Rate (Age-Adjusted; Per 10...	2010	Both	

	race_ethnicity	value	place	95pc_con_lvl-low	\
24934	All	583.3	San Francisco, CA	574.3	
24935	All	606.0	Seattle, WA	586.2	
24936	All	630.1	San Diego County, CA	621.1	

	95pc_con_lvl-high
24934	592.4
24935	626.3
24936	639.1

Analysis

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

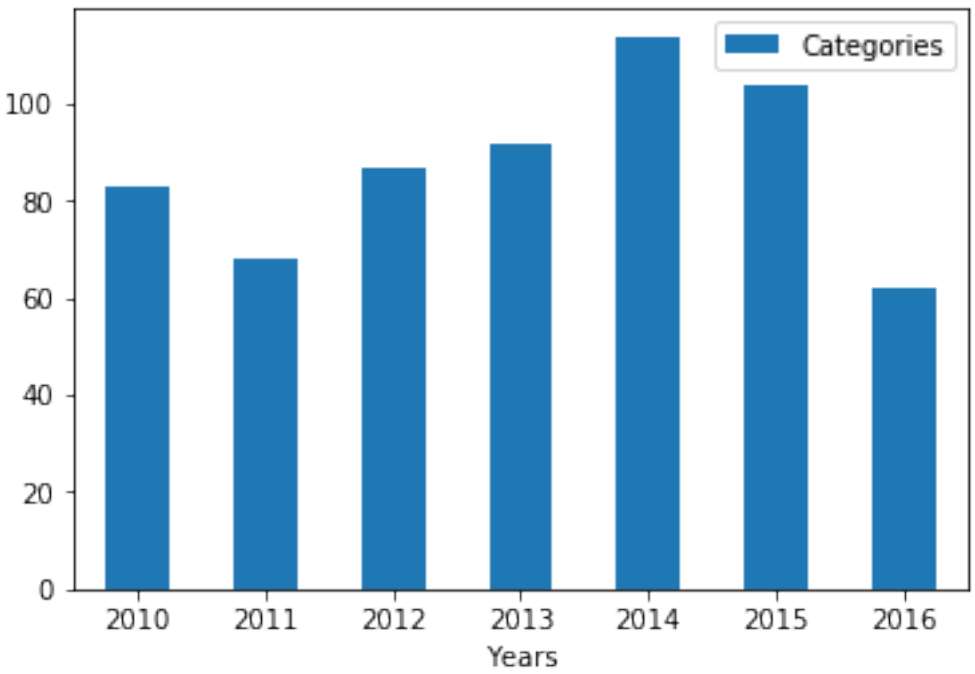
Following is the process to do the same

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

In [12]: c_year_2010.shape
```

Out[12]: (83, 8)

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

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

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

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

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

         c_year_2016=lde_ds[lde_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)
```



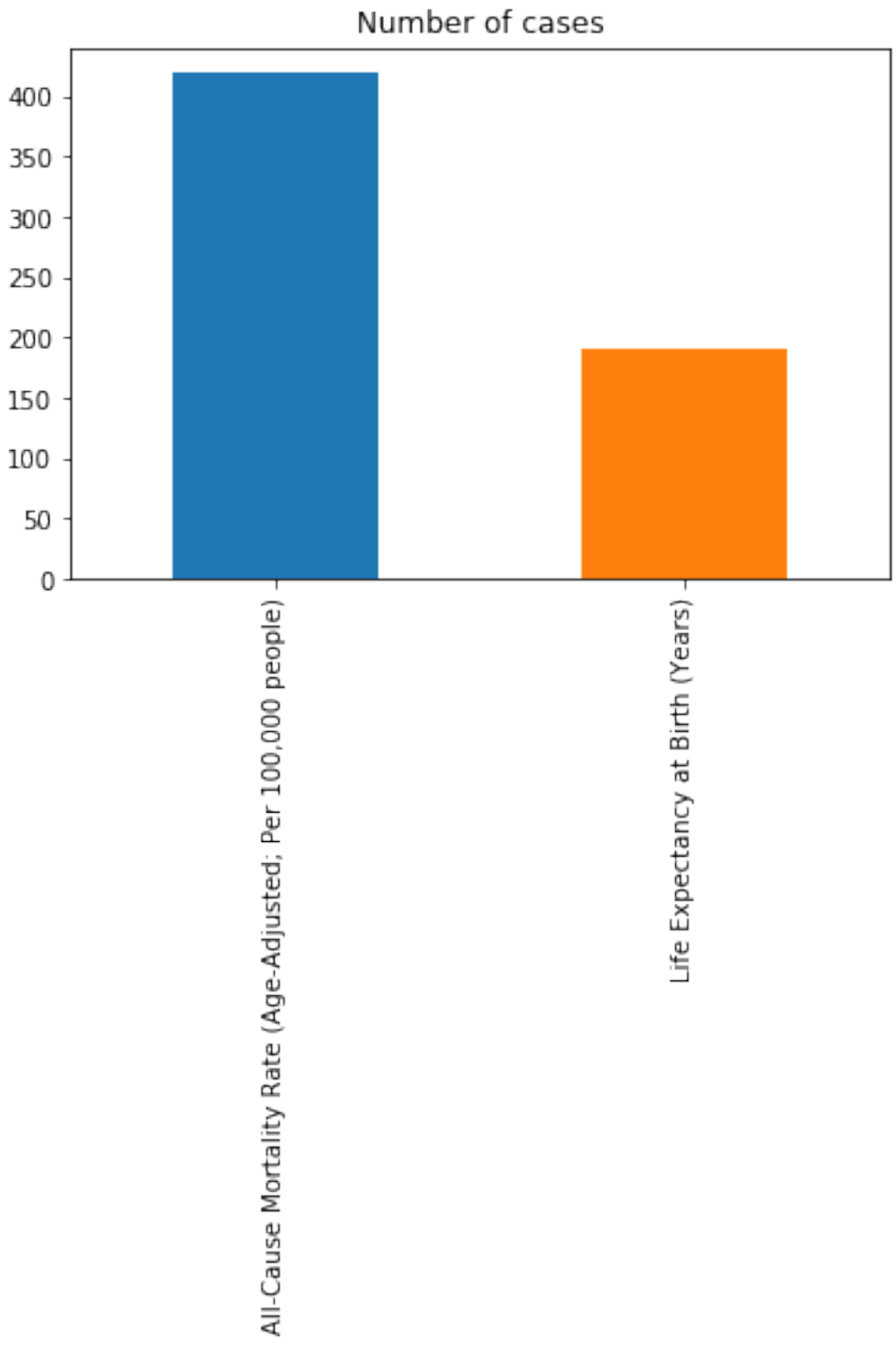
Now we calculate the number of cases for each type of cancer. In order to that we will group according to the indicator and take the count.

```
In [15]: sorted_lde = lde_ds['indicator'].value_counts()
         sorted_lde
```

```
Out[15]: All-Cause Mortality Rate (Age-Adjusted; Per 100,000 people)    420
         Life Expectancy at Birth (Years)                              190
         Name: indicator, dtype: int64
```

And we plot a histogram to see.

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

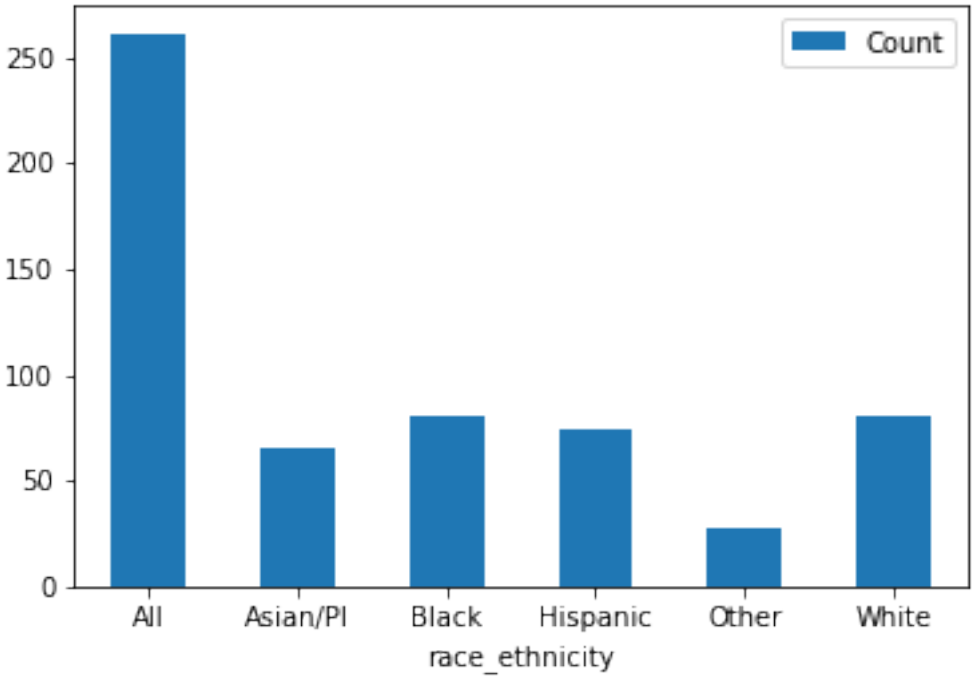


Now we find out the distribution of cancer patients with respect to the race and ethnicity.

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

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

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

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

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

white=lde_ds[lde_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]:
```
lde_ds=lde_ds.rename(columns={'95pc_con_lvl-low':'low','95pc_con_lvl-high':'high'})
```

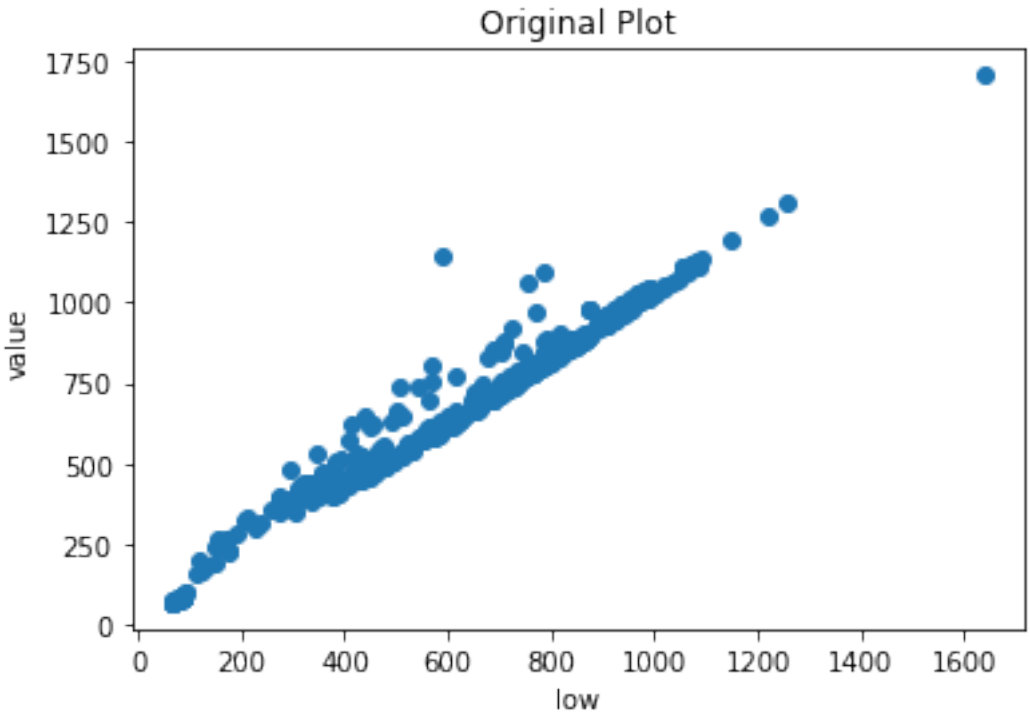
In [20]:
```
x='95pc_con_lvl-low'
y='95pc_con_lvl-high'
ds=lde_ds.drop(['indicator','year','sex','race_ethnicity'
                ,'place','high'],
                    axis = 1)
```

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

Out[21]:

	value	low
24934	583.3	574.3
24935	606.0	586.2
24936	630.1	621.1
24937	673.5	647.4
24939	751.0	736.9

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);
```

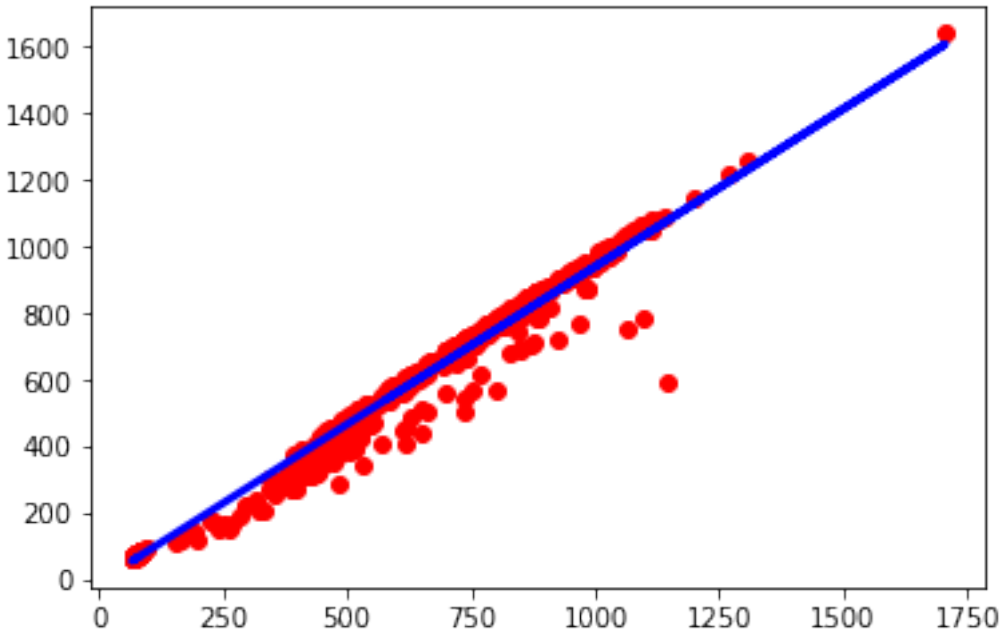


In [23]:
```
x_y = np.array(ds)
x, y = x_y[:,0], x_y[:,1]

# 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 = [[0.94706288]]x + [-6.94855848]
```

In [24]:
```
plt.scatter(x, y,  color='red')
plt.plot(x, predictions, color='blue',linewidth=3)
plt.show()
```



In [ ]: