Shaikat_303527_exercise1

October 31, 2019

1 1.1 Python and Numpy

1.1 Read the data from the csv

pandas dataframe can be used to import the data from csv file

```
In [86]: import matplotlib.pyplot as plt
         import numpy as np
         import pandas as pd
         pd.options.mode.chained_assignment = None
         df = pd.read_csv(r'E:\Documents\University of Hildesheim\Machine learning lab\lab1\Gra
         print(df.head())
```

```
  First Name  Last Name  English    Math  Science  German  Sports  Final Grade
0      Robyn    Hobgood    60.95   24.77    20.60   69.32    8.36       184.00
1       Eddy  Swearngin   100.00   12.99   100.00   52.24  100.00       365.23
2      Leoma   Bridgman    83.37  100.00    78.69  100.00   19.50       381.56
3    Arnetta      Peart    87.75  100.00    86.93   87.90   41.73       404.31
4   Maryland      Colby   100.00  100.00   100.00   18.87   88.72       407.59
```

1.2 Computing sum of all the subjects for each students

By accessing each value in dataframe and putting it in the for loop to find the sum on after another.The loop goes on till the maximum number of students is reached.

```
In [87]: num_students=len(df)
         sum=[]
         for i in range (0,num_students):
             sum.append(df['English'][i]+df['Math'][i]+df['Science'][i]+df['German'][i]+df['Spo
         print(np.round(sum))
```

```
[184. 365. 382. 404. 408. 263. 404. 367. 476. 341. 188. 274. 429. 395.
 339. 430. 227. 285. 271. 373. 462. 343. 455. 440. 355. 373. 219. 361.
 241. 225.]
```

1.3 Computation of the average of the point for each student

Dividing the sum of all the subjects of each student with the total points which is 500

```
In [88]: avg=[]
         for i in range (0,num_students):
             avg.append(sum[i]/500)
         print(np.round(avg,2))
```

```
[0.37 0.73 0.76 0.81 0.82 0.53 0.81 0.73 0.95 0.68 0.38 0.55 0.86 0.79
 0.68 0.86 0.45 0.57 0.54 0.75 0.92 0.69 0.91 0.88 0.71 0.75 0.44 0.72
 0.48 0.45]
```

1.4 Compute the standard deviation of point for each student

Numpy standard deviation library made it easy to compute the standard deviation at a glance.Using (np.std) by putting all the values in the array named points and calculate the standard deviation

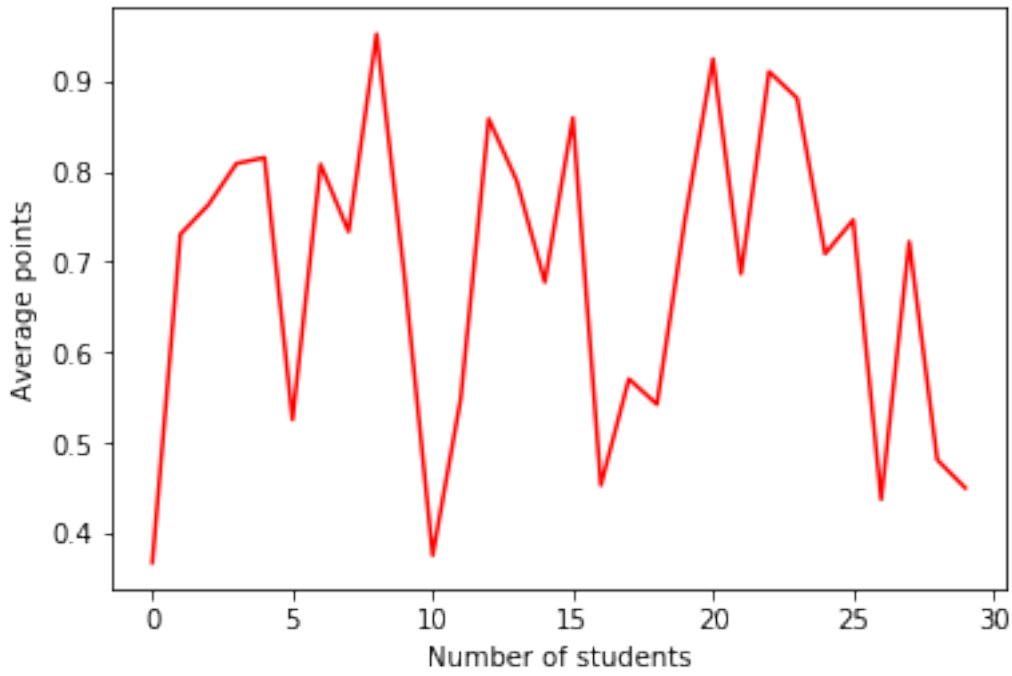
```
In [89]: stdev=[]
         for i in range (0,num_students):
             points=[df['English'][i],df['Math'][i],df['Science'][i],df['German'][i],df['Sports
             stdev.append(np.std(points))
         print(np.round(stdev,2))
```

```
[23.9  35.27 29.68 20.16 31.63 56.32 28.84 28.44  6.55 30.03 33.43 32.38
 12.27 19.92 28.   25.55 39.03 15.32 35.68 36.03 14.92 53.3  17.96 23.82
 45.44 23.34 30.27 16.43 28.47 22.  ]
```

1.5 Plot the average points for all the students (in one figure)

```
In [90]: plt.xlabel("Number of students")
         plt.ylabel("Average points")
         plt.plot( np.arange(num_students),avg,'r')
```

```
Out[90]: [<matplotlib.lines.Line2D at 0x1d922814278>]
```



1.6 For each student assign a grade based on the following rubric

Assigning number of students accoring to their percentage acquired which is (average*100)

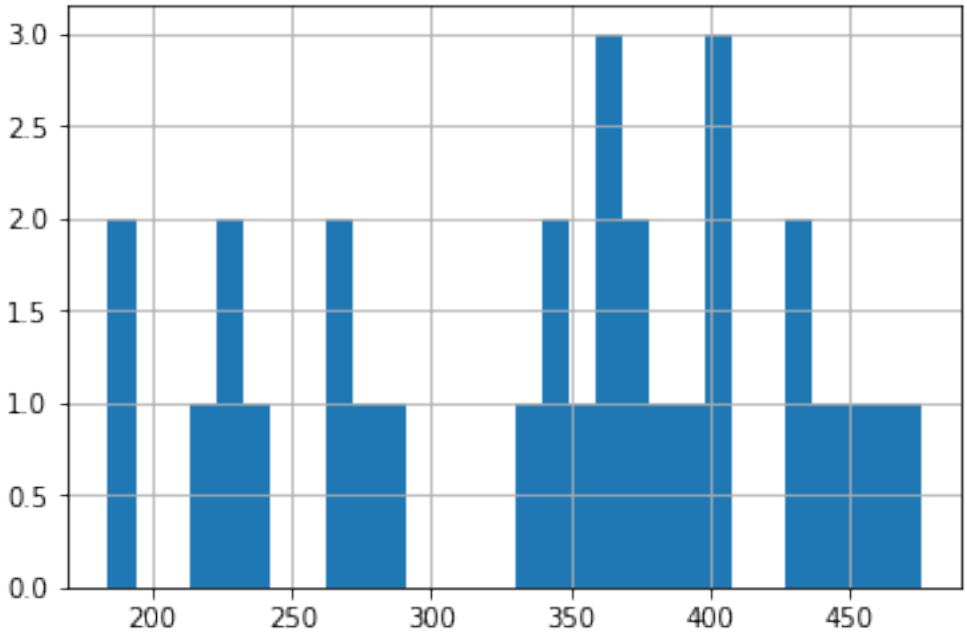
```
In [91]: #creating instances for dataframe
         gradsys = [ ('96-100','A+',0) ,
                     ('90-95','A',0) ,
                     ('86-89','A-',0),
                     ('80-85','B+',0),
                     ('76-79','B',0) ,
                     ('70-75','B-',0),('66-69','C+',0),('60-65','C',0),('56-59','D',0),('0-55

         #creating dataframe columns with and also exporting the values in to it.
         grad = pd.DataFrame(gradsys , columns = ['%range' , 'Grade', '#of students'])

         for i in range (0,num_students):
             num=avg[i]*100
             if num<=100 and num>=96:
                 grad['#of students'][0]+=1
             elif num<95 and num>90:
                 grad['#of students'][1]+=1
             elif num < 89 and num > 86:
                 grad['#of students'][2]+=1
             elif num < 85 and num > 80:
                 grad['#of students'][3]+=1
```

```
            elif num < 79 and num > 76:
                grad['#of students'][4]+=1
            elif num <75 and num >70:
                grad['#of students'][5]+=1
            elif num <69 and num >66:
                grad['#of students'][6]+=1
            elif num <65 and num >60:
                grad['#of students'][7]+=1
            elif num <59 and num >56:
                grad['#of students'][8]+=1
            elif num <55 and num >0:
                grad['#of students'][9]+=1

        print(grad)
```

```
   %range Grade  #of students
0  96-100    A+             0
1   90-95     A             2
2   86-89    A-             1
3   80-85    B+             3
4   76-79     B             2
5   70-75    B-             6
6   66-69    C+             3
7   60-65     C             0
8   56-59     D             1
9    0-55     F             9
```

1.7 Plot the histogram of the final grades.

```
In [92]: hist = df['Final Grade'].hist(bins=30)
```



1.8 Matrix Multiplication

Iterative multiply (element-wise)

```
In [93]: matA=np.random.rand(100,20)
         v=np.random.normal(5, 0.01, 20)
         for i in range(len(matA)):
             #creating empty list for the resulting vector
             c = []
             #variable for each row of action
             elements = 0
             for j in range(len(v)):
                 #adding the result of matrix vector element-/wise multiplication
                 elements += matA[i][j] * v[j]
                 c.append(elements)
         print(c)
```

```
[1.0984501586781539, 2.08949885809247, 4.700620823324904, 7.068489213867251, 8.072699662168013
```

The mean and standard deviation of the new vector c

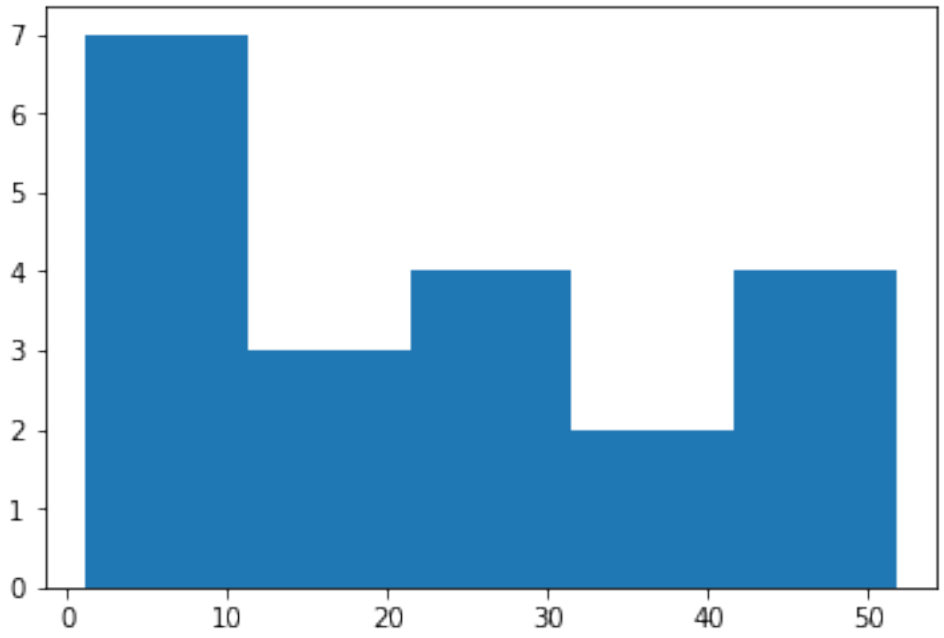
```
In [94]: mean=np.mean(c)
         std=np.std(c)
         print('The mean of vector C is',np.round(mean,2))
         print('The standard deviation of vector C is',np.round(std,2))
```

```
The mean of vector C is 23.37
The standard deviation of vector C is 15.92
```

Histogram of vector c using 5 bins

```
In [95]: plt.hist(c, bins=5)
```

```
Out[95]: (array([7., 3., 4., 2., 4.]),
          array([ 1.09845016, 11.23715402, 21.37585787, 31.51456173, 41.65326559,
                 51.79196945]),
          <a list of 5 Patch objects>)
```



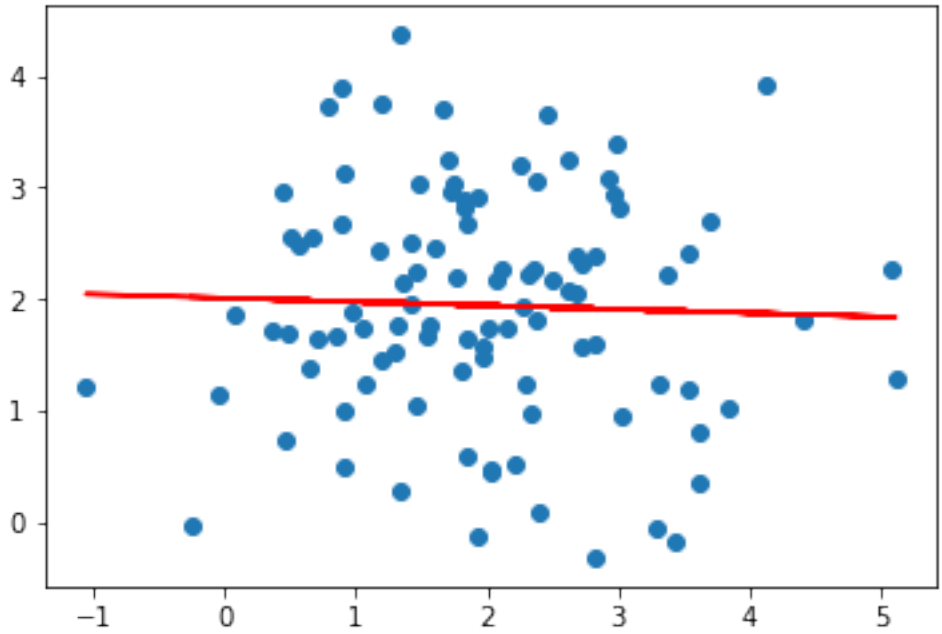
2 1.2 Linear Regression through exact form

```
In [126]: v=np.random.normal(2, 1,(100,2))
          X=v[:,0].reshape(-1,1).astype(float)
          y=v[:,1].reshape(-1,1).astype(float)
          n=len(X)
```

```
In [127]: beta0 = (np.sum(y)*np.sum(X**2) -  np.sum(X)*np.sum(X*y))/ (n * np.sum(X**2) - np.su
          beta1 = (n*np.sum(X*y) - np.sum(X)*np.sum(y)) / (n*np.sum(X**2) - np.sum(X)**2)
          print("Beta 0 is ", beta0)
          print("Beta 1 is ", beta1)
          y_pred = beta0 + beta1 * X
          plt.scatter(X,y)
          plt.plot( X,y_pred,'r')
```

```
Beta 0 is  2.0087939590229116
Beta 1 is  -0.033462829011926494
```

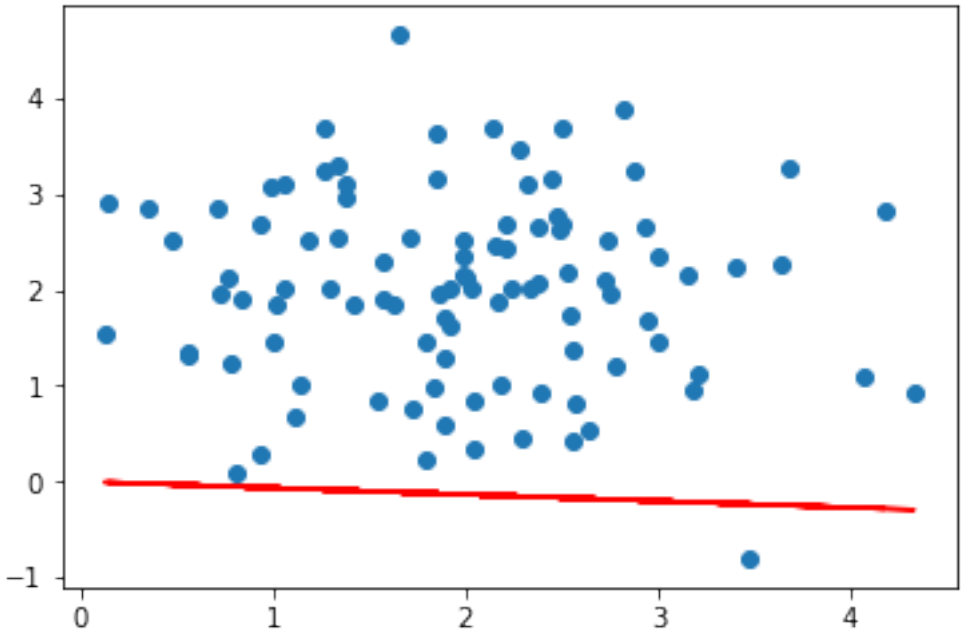
```
Out[127]: [<matplotlib.lines.Line2D at 0x1d923e82eb8>]
```



2.0.1 When Beta0 is zero the predicted line goes down the standard deviation between the line and the points increaseas and gives a bad predicted line.

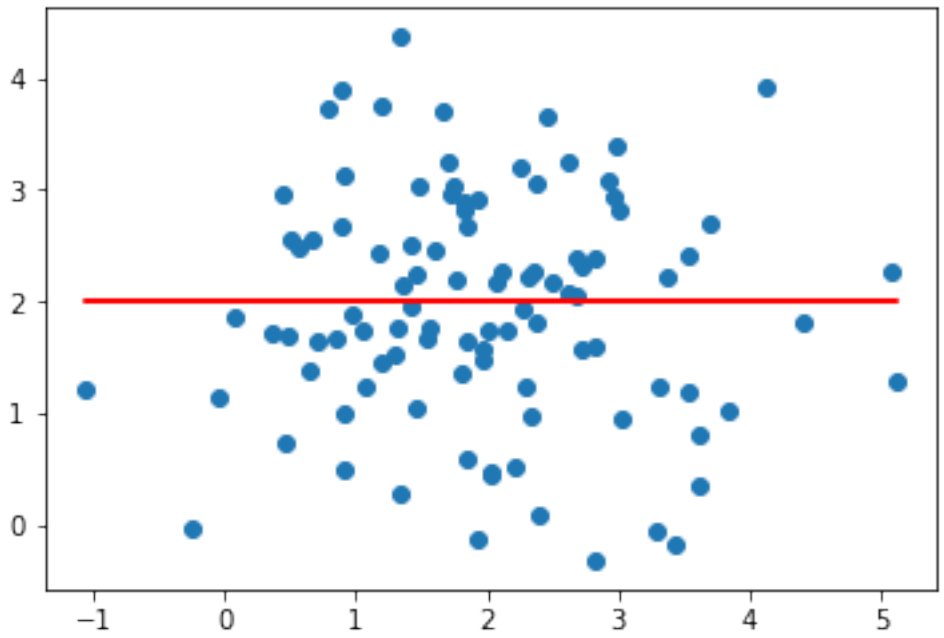
```
In [125]: beta0=0
          y_pred = beta0 + beta1 * X
          plt.scatter(X,y)
          plt.plot( X,y_pred,'r')
```

```
Out[125]: [<matplotlib.lines.Line2D at 0x1d923e2f0b8>]
```



2.0.2 When beta1 is zero there is no significant change in the predicted line

```
In [128]: beta1=0
          y_pred = beta0 + beta1 * X
          plt.scatter(X,y)
          plt.plot( X,y_pred,'r')

Out[128]: [<matplotlib.lines.Line2D at 0x1d923eedc88>]
```



Using numpy.linalg lstsq there is very small difference between the values of beta0 and beta1

```
In [129]: Z = np.vstack([X.T,(np.ones((len(X),1))).T]).T
          beta1,beta0 = np.linalg.lstsq(Z, y, rcond=None)[0]
          print("Beta 0 is ", beta0)
          print("Beta 1 is ", beta1)

Beta 0 is  [2.00879396]
Beta 1 is  [-0.03346283]
```

3 1.2.2 OLS using a Real dataset 6 Points

```
In [227]: filename = r"E:\Documents\University of Hildesheim\Machine learning lab\lab1\auto-mpg
          column_names = ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'accelera
          auto_dat = pd.read_csv(filename, delim_whitespace=True, names=column_names)
          auto_dat.head(5)

Out[227]:     mpg  cylinders  displacement horsepower  weight  acceleration  year  \
          0  18.0          8         307.0      130.0  3504.0          12.0    70
          1  15.0          8         350.0      165.0  3693.0          11.5    70
          2  18.0          8         318.0      150.0  3436.0          11.0    70
          3  16.0          8         304.0      150.0  3433.0          12.0    70
          4  17.0          8         302.0      140.0  3449.0          10.5    70

             origin                       name
          0       1  chevrolet chevelle malibu
```

```
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

Split it into our uni-variate case 'Displacement' as the independent variable and 'MPG' as the dependant variable.

```
In [228]: X=auto_dat['displacement']
          y=auto_dat['mpg']
```

LEARN-SIMPLE-LINREG algorithm

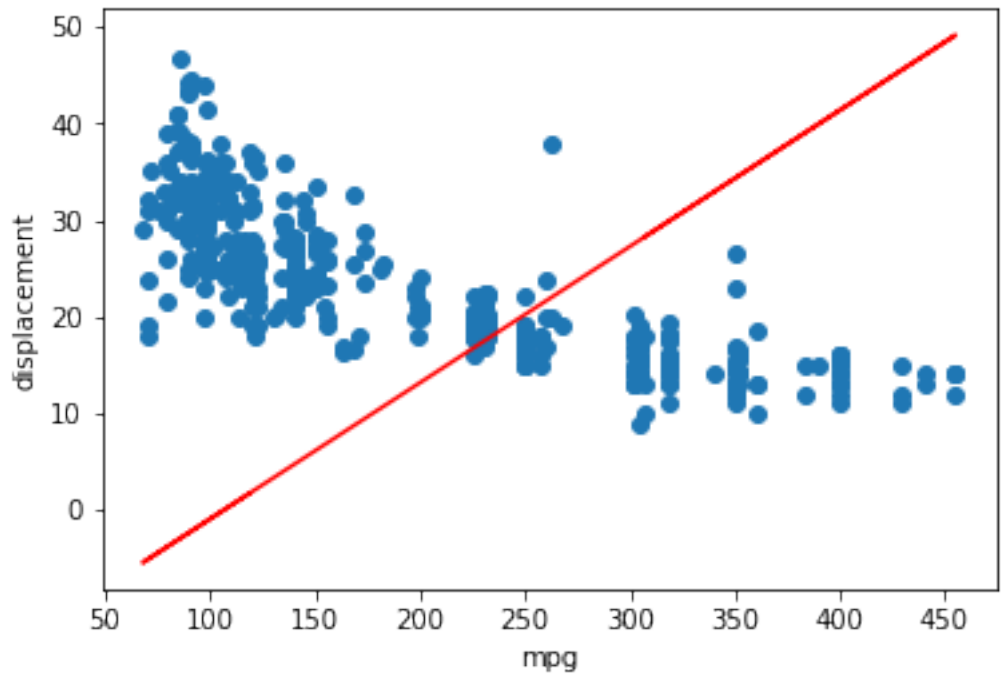
```
In [229]: beta0 = (np.sum(y)*np.sum(X**2) -  np.sum(X)*np.sum(X*y))/ (n * np.sum(X**2) - np.su
          beta1 = (n*np.sum(X*y) - np.sum(X)*np.sum(y)) / (n*np.sum(X**2) - np.sum(X)**2)
          print("Beta 0 is ", beta0)
          print("Beta 1 is ", beta1)
```

```
Beta 0 is  -15.084665380984553
Beta 1 is  0.14116358100240253
```

PREDICT-SIMPLE-LINREG

```
In [230]: y_pred = beta0 + beta1 * X
          plt.scatter(X,y)
          plt.xlabel("mpg")
          plt.ylabel("displacement")
          plt.plot( X,y_pred,'r')
```

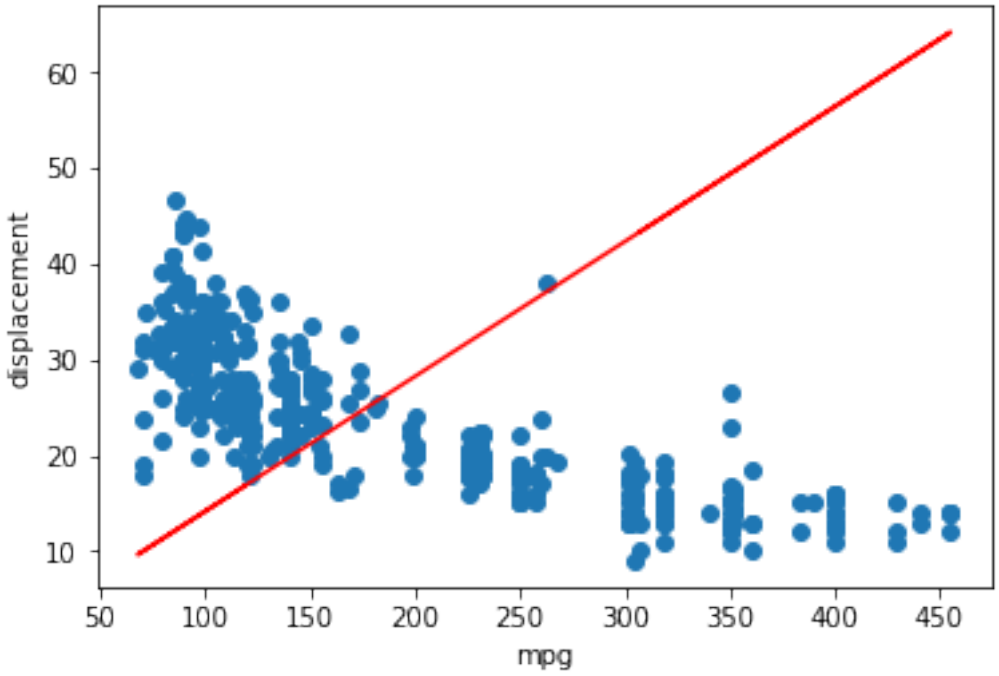
```
Out[230]: [<matplotlib.lines.Line2D at 0x1d926b91438>]
```



3.0.1 When Beta0 is zero the predicted line goes more towards the concentrated data points

```
In [226]: beta0=0
          y_pred = beta0 + beta1 * X
          plt.scatter(X,y)
          plt.xlabel("mpg")
          plt.ylabel("displacement")
          plt.plot( X,y_pred,'r')
```

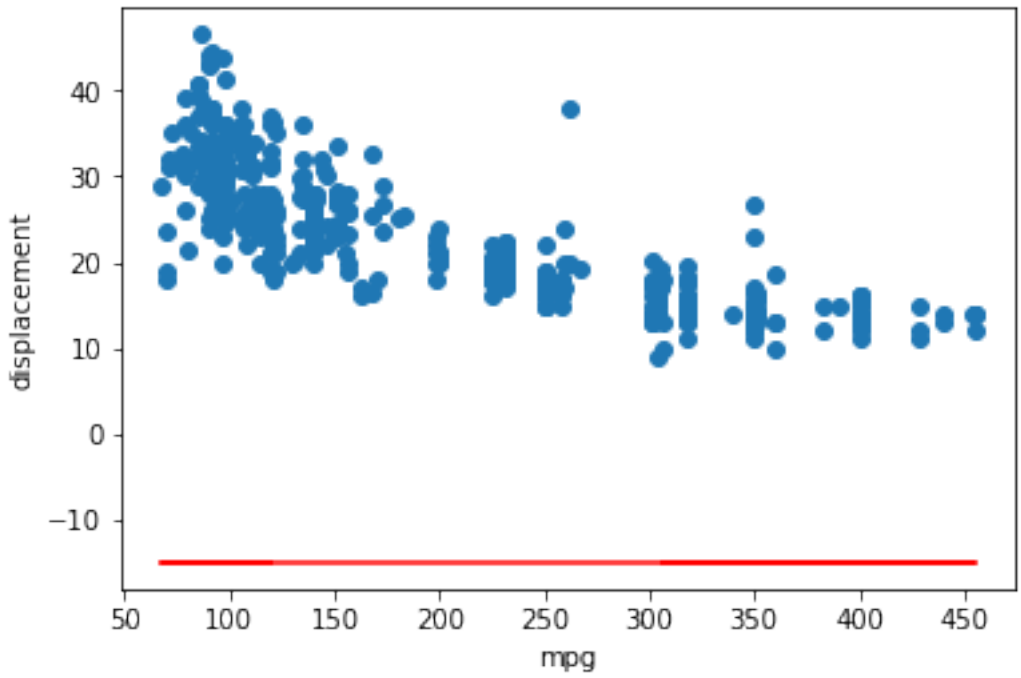
```
Out[226]: [<matplotlib.lines.Line2D at 0x1d926f3dcf8>]
```



3.0.2 When beta1 is zero the line goes away from the data points gives a bad predicted line

```
In [231]: beta1=0
          y_pred = beta0 + beta1 * X
          plt.xlabel("mpg")
          plt.ylabel("displacement")
          plt.scatter(X,y)
          plt.plot( X,y_pred,'r')
```

```
Out[231]: [<matplotlib.lines.Line2D at 0x1d926b91f28>]
```

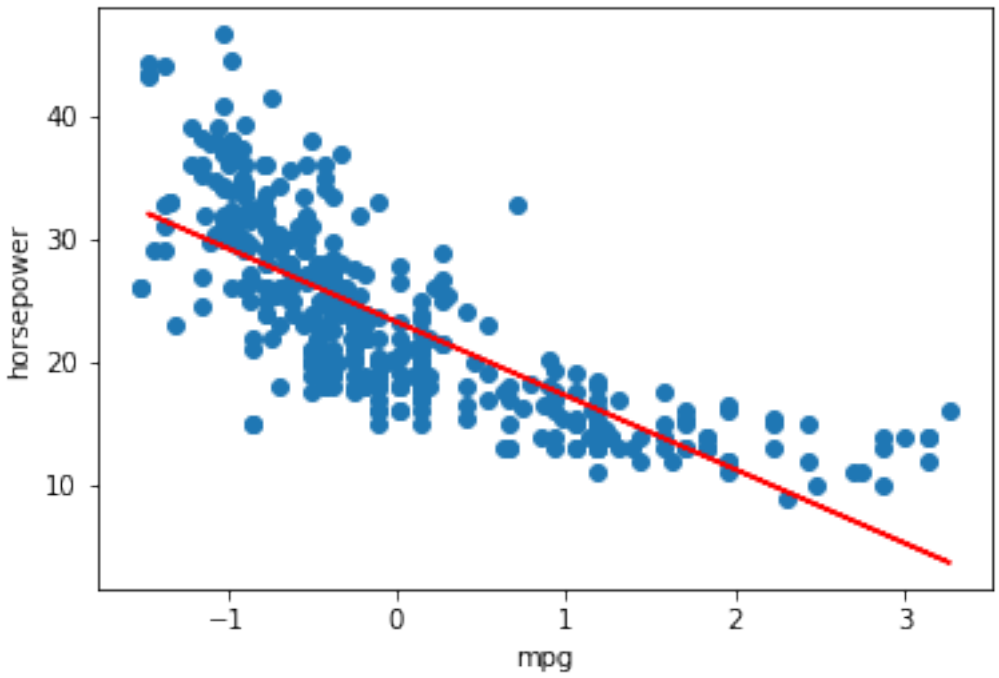


3.0.3 Cleaning the data by removing '?' and nan in the horsepower data

```
In [232]: auto_dat=auto_dat.dropna()
          auto_dat = auto_dat[auto_dat.horsepower != '?']

In [238]: X=auto_dat[['horsepower']].astype(float)
          y=auto_dat['mpg']
          from sklearn.preprocessing import StandardScaler
          from sklearn.model_selection import train_test_split
          X = StandardScaler().fit_transform(X)

In [239]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size = 0.33,random_state=324)
          regressor = LinearRegression().fit(X_train,y_train)
          y_pred = regressor.predict(X_test)
```

3.0.4 In this scatter plot we can understand that with the increase of mpg the horsepower decreases

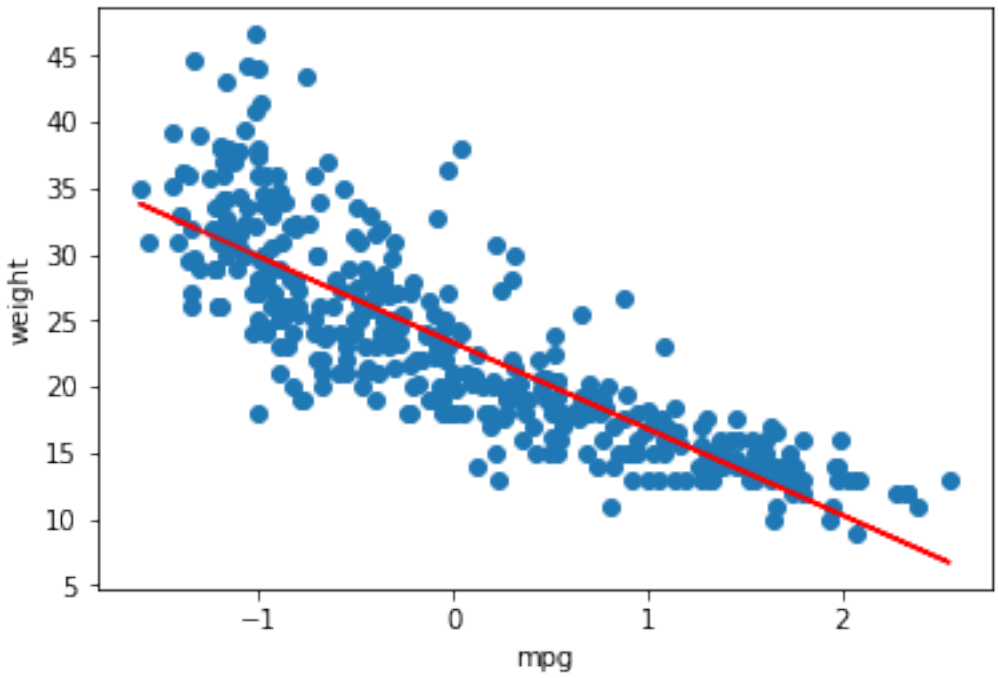
```
In [240]: plt.xlabel("mpg")
          plt.ylabel("horsepower")
          plt.scatter(X,y)
          plt.plot( X_test,y_pred,'r')

Out[240]: [<matplotlib.lines.Line2D at 0x1d927fdc7f0>]
```




```
In [241]: X=auto_dat[['weight']].astype(float)
          y=auto_dat['mpg']
          from sklearn.preprocessing import StandardScaler
          from sklearn.model_selection import train_test_split
          X = StandardScaler().fit_transform(X)

In [242]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size = 0.33,random_state=324)
          regressor = LinearRegression().fit(X_train,y_train)
          y_pred = regressor.predict(X_test)
```

3.0.5 In this scatter plot we can understand that with the increase of mpg the weight also decreases

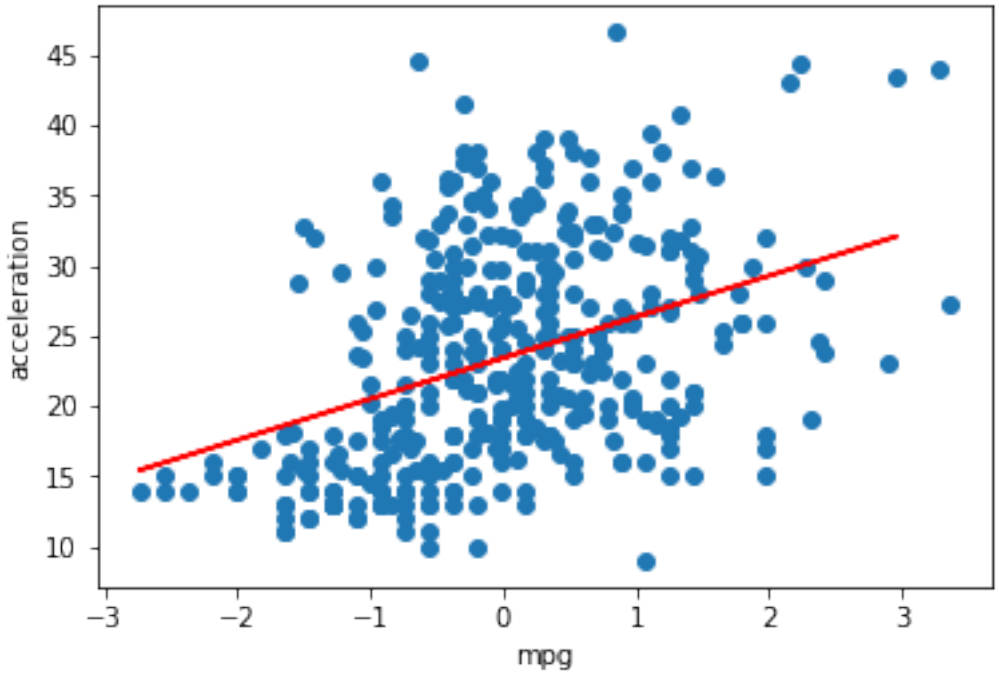
```
In [243]: plt.xlabel("mpg")
          plt.ylabel("weight")
          plt.scatter(X,y)
          plt.plot( X_test,y_pred,'r')

Out[243]: [<matplotlib.lines.Line2D at 0x1d92803a080>]
```




```
In [244]: X=auto_dat[['acceleration']].astype(float)
          y=auto_dat['mpg']
          from sklearn.preprocessing import StandardScaler
          from sklearn.model_selection import train_test_split
          X = StandardScaler().fit_transform(X)

In [245]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size = 0.33,random_state=324)
          regressor = LinearRegression().fit(X_train,y_train)
          y_pred = regressor.predict(X_test)
```

3.0.6 In this scatter plot we can understand that with the increase of mpg the acceleration increases

```
In [246]: plt.xlabel("mpg")
          plt.ylabel("acceleration")
          plt.scatter(X,y)
          plt.plot( X_test,y_pred,'r')

Out[246]: [<matplotlib.lines.Line2D at 0x1d92803a7f0>]
```



In [ ]: