Machine Learning Lab - Exercise 1

November 9, 2018

Question 1:

Pandas and Numpy - Word Count Program:

Function Name : Readlinesfromtext
Parameter : Dataframe of size nx2

This function is used to split the entire content into lines and remove the non alpha-numeric words from the given data.

```
with open(text_file) as f:
     line_list.append(pd.DataFrame({'lines': f.readlines()}))
 lines = pd.concat(line_list)
 lines['words'] = lines.lines.str.strip().str.split('[\W_]+')
return lines
```

Function Name : Listofwords
Parameter : List of lines

This function is used to obtain a list of entire words within the document and it also changes the entire case to lower.

```
for row in lines[['words']].iterrows():
    r = row[1]
    for word in r.words:
        rows.append(word)
words = pd.DataFrame(rows, columns=['Words'])
words = words[words.Words.str.len()>0]
words['Words'] = words.Words.str.lower()
return words
```

Function Name : Ignorewords
Parameter : Set of 'english' stopwords

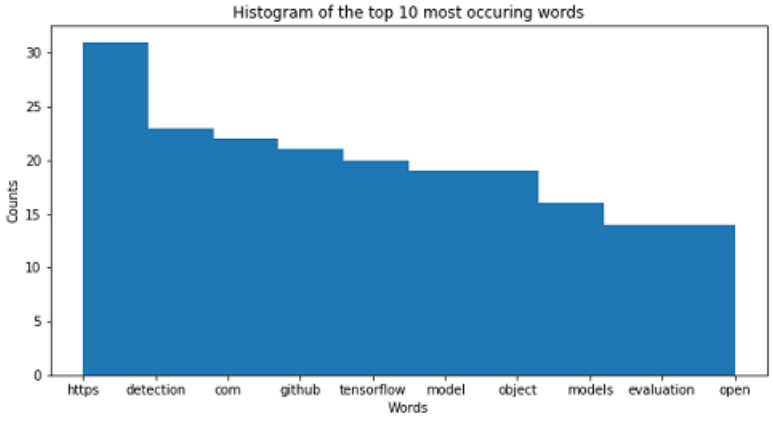
This function is used to remove the 'english' stopwords from the text.

```
for word in ignore:
    if word in Count:
        del Count[word]
return Count
```

Function Name : value_counts
Parameter : List of words

value_counts is used to find the number of occurances of a word within the document. The output of the function is filtered for the top 10 occurances and the corresponding histogram is plotted.

```
https         31
detection     23
com           22
github        21
tensorflow    20
model         19
object        19
models        16
evaluation    14
open          14
Name: Words, dtype: int64
```



Pandas and Numpy - Matrix Multiplication:

Function Name : Matrix_Multiplication
Parameter : A matrix and a vector

In this function, Iterative multiplication (element-wise) is done for each row of matrix A with vector v and sum the result of each iteration is stored in another vector c.

```
    Vector_c = np.zeros((n_A,n_v))
    for i in range (0, n_A):
        for j in range (0,m_A):
            Vector_c[i]+= (Matrix_A[i][j] * Vector_v[j])
    return Vector_c
```

```
[20.96162182 19.56128573 20.75263873 22.09995454 16.32978885 22.31756084
 18.77156794 15.12949876 21.78268593 20.20400218 21.77405606 18.73522606
 14.90231368 23.99122341 17.82379525 22.14313899 21.99760998 17.32301123
 19.03948651 21.49589893 22.60065732 20.16808417 14.67826945 16.97366468
 20.23623369 17.26524136 22.85673769 19.21600885 23.32548529 19.59021416
 21.02253267 16.36011863 19.12939128 19.69555951 20.19005332 13.98585096
 17.97767196 21.04218385 21.0287408  12.44852775 17.2916532  20.21053966
 18.54546587 24.79076029 18.25231387 22.21894605 17.91374321 25.2323708
 22.97210353 19.06165341 18.00985139 17.94862754 21.51943467 18.91889688
 16.39735938 18.58650516 18.11139563 17.20924416 19.607072   22.66669609
 22.17612591 19.46685762 17.5129615  22.85475754 18.97431278 16.61819263
 23.80929392 20.75902655 20.16511893 18.33580083 21.44682306 17.44158478
 20.04711787 14.4902775  19.15677529 18.43735319 21.78585561 18.35214676
 22.86277592 20.48326481 21.50047253 20.31393964 19.36880564 20.89648382
 23.48673156 21.59609461 16.08407309 18.27123393 21.78135299 18.5149742
 19.39773124 22.97120736 20.81048611 20.29007649 23.87998265 18.65202898
 18.1594295  23.85161909 16.76806737 23.92121874]
```

Function Name : Vector_Mean

Parameter : A vector

This function is used to find the average of the values within the given vector.

```
    for i in range (0,n_c):
        for j in range (0,m_c):
            total+= Vector_c[i][j]
    Mean = total/(n_c*m_c)
    return Mean
```

Function Name: Vector_SD

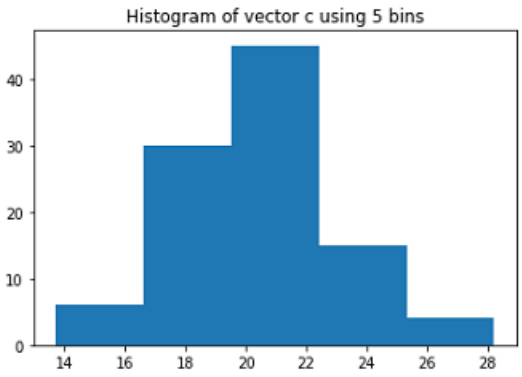
Parameter : A vector

This function is used to find the standard deviation the given vector. After finding the standard deviation, a histogram is plotted with the values of vector c by dividing into 5 bins.

```
    n_c,m_c = Vector_c.shape
    total = 0
    for i in range (0,n_c):
        for j in range (0,m_c):
                    total+= pow((Vector_c[i][j] - Mean),2)
    SD = pow(total/(n_c*m_c),0.5)
    return SD
```

Mean of Vector C : 19.7808

Standard Deviation of Vector C : 2.5523



Linear Regression through exact form:

Function Name : generatematrix
Parameter : Value of mean and standard deviation

In this function, 3 sets of matrices say a,b and c are generated with random values and normal distribution using the values of mean and standard deviation.

```
a = np.random.normal(mean,sd[0],(100,2))
b = np.random.normal(mean,sd[1],(100,2))
c = np.random.normal(mean,sd[2],(100,2))
return a,b,c
```

Function Name : matrixmean
Parameter : Matrix

Mean of the training values is required during the calculation of predicting a value and this function returns the mean of the matrix being sent as input.

```
return Matrix.mean(0)
```

A_Mean : [2.0002 2.0002]
B_Mean: [2.0111 1.9751]
C_Mean: [2.0299 2.0726]

Function Name : Slope
Parameter : Matrix and Mean of the matrix

To predict a value from training values, slope and intercepts play a vital role in determining the model. This function returns the slope of the training values.

```
return ((np.sum((Matrix[:,0]-Matrix_mean[0])*(Matrix[:,1]-Matrix_mean[1])))/
                    (np.sum(pow((Matrix[:,0]-Matrix_mean[0]),2))))
```

A: Slope β_1 = -0.068
B: Slope β_1 = 0.08
C: Slope β_1 = 0.106

Function Name : Intercept
Parameter : Slope and Mean of the matrix

While predicting a value from training values, slope and intercepts is required to determine the model. This function returns the intercept of the training values.

```
return (Matrix_mean[1] - (b1*Matrix_mean[0]))
```

A: Intercept β_0 = 2.136
B: Intercept β_0 = 1.8
C: Intercept β_0 = 1.857

Function Name : predict
Parameter : Slope, Intercept and Matrix

With the obtained slope and intercept, using this function we will be predicting the value of y. Which will be further used to calculate MSE and determining the best model for classification.

```
Matrix_predict = np.zeros((100,2))
Matrix_predict[:,0] = Matrix[:,0]
for i in range (0,len(Matrix)):
    Matrix_predict[i][1] = b0+(b1*Matrix[i][0])
 return Matrix_predict
```

Predicted value of Matrix A

```
[1.99948045 1.99977265 2.00046112 1.99861974 1.99993796 2.00121316
 2.00034185 2.0008403  2.00030618 1.99994112 1.99992845 1.99998968
 2.00095143 2.00014074 1.99987069 2.00042296 1.99932464 1.99984797
 1.9995676  1.99997671 2.00022009 2.00013981 1.99929124 2.00076949
 1.99888066 1.99950116 2.00060605 2.00062572 2.00001423 1.99989912
 2.00001781 2.00038838 1.99988569 1.99927785 2.00047603 1.99926495
 1.99953117 1.9995755  1.99965867 1.99859838 2.00036531 2.00033435
 1.99966308 2.00049634 2.00054862 1.99921706 2.00038289 2.00041481
 1.99970699 1.99991409 2.0006854  2.00023651 1.99891129 2.00133413
 1.99994842 2.00018957 2.00016717 1.99995324 1.99927268 2.00068381
 1.99964963 2.00020387 1.99944835 1.99999662 1.99988579 2.00051632
 1.99997067 1.99979708 2.00006384 2.00048188 1.9998961  2.00061528
 1.9995733  2.00106672 2.00010326 1.99871621 2.0001059  1.99923325
 1.99983191 1.99866413 1.9995648  1.99933928 2.00156648 1.99982324
 2.00029542 2.00044118 1.99895193 2.00029555 1.99947357 1.99942095
 2.00050136 1.99990282 2.00029452 1.99909434 2.00064938 1.99983462
 2.00024731 2.00072832 2.00006975 2.00021965]
```

Function Name : calculateMSE
Parameter : Matrix and Predicted Matrix

This function calculates the Mean Square Error (MSE) of the estimator and the estimator with least value of MSE is taken as the best.

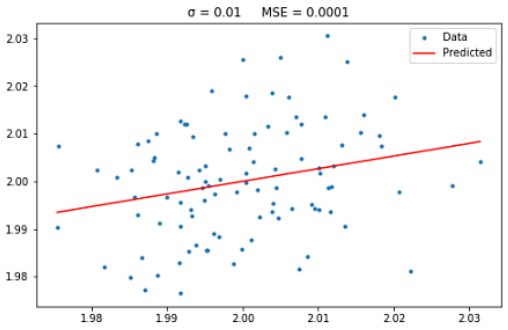
```
return np.sum(pow((Matrix[:,1] - Matrix_Predicted[:,1]),2))/len(Matrix)
```

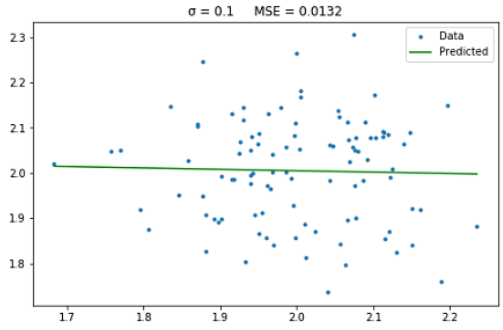
A: MSE (σ:0.01) = 0.0001
B: MSE (σ: 0.1) = 0.0132
C: MSE (σ: 1.0) = 1.0789

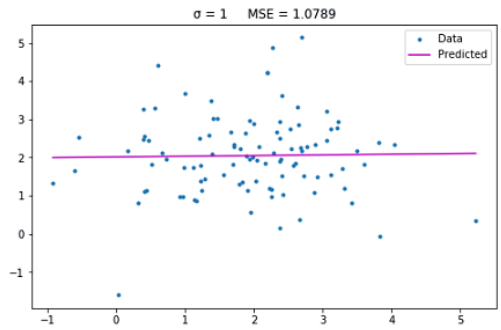
Function Name : LinReg
Parameter : Matrix

This function is the hub where all other functions are being called. When a matrix is given as input, this function calls other dependent functions and predicts the value. This function also plots the input data and the predicted value in forms of dots and line respectively.

```
b1 = round(slope(Matrix,Matrix_mean),3)
b0 = round(intercept(Matrix_mean,b1),3)
Matrix_predict = predict(b0,b1,Matrix)
MSE = calculateMSE(Matrix,Matrix_predict)
Matrix_predict = Matrix_predict[Matrix_predict[:,1].argsort()]
fig1,ax1 = plt.subplots(figsize = (8,5))
ax1.plot(Matrix[:,0],Matrix[:,1],'.',Matrix_predict[:,0],Matrix_predict[:,1],'bo')
ax1.legend(['Data','Predicted'])
plt.title('σ = '+str(sd)+'      MSE = ' +str(round(MSE,4)))
```





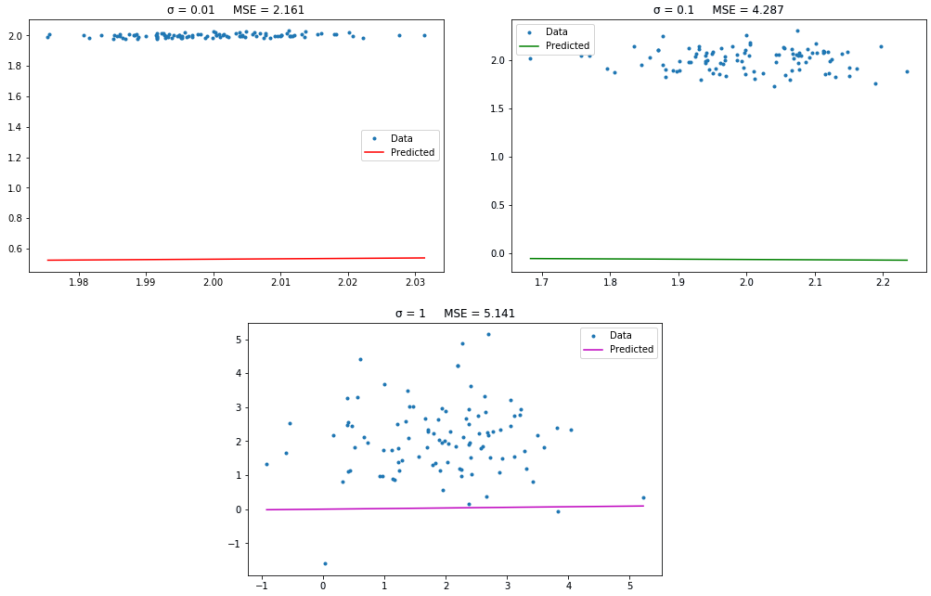


Effect of σ on the line:

As the standard deviation is increased, the range of the training values increases which results in the increase of distance between the training values, x and y, and hence the error component increases. When σ is 0.01, range of x is 1.97 to 2.03 and MSE is 0.0001 as the σ is 1,range of x is -1 to 6 andthe MSE is 1.0789 (high).

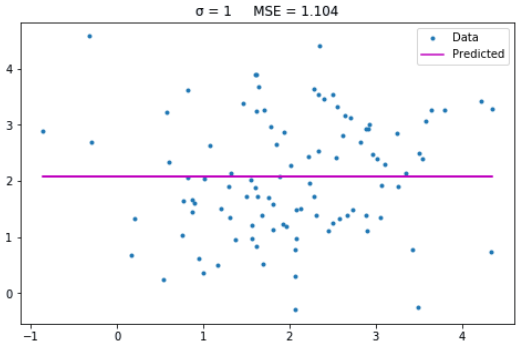
When β_0 is set to 0:

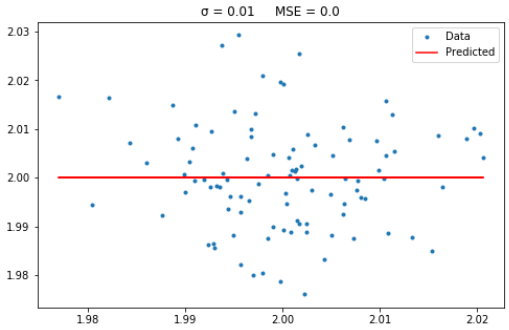
In this case, the value of prediction solely depends on the value of slope. Hence, as the slope increases, the error increases and vice-versa. Also, when varying the values of σ, the error increases because of the increase in distance between the training samples. In total, σ increases, the Mean Square Error of the estimator increases which makes it a bad model.

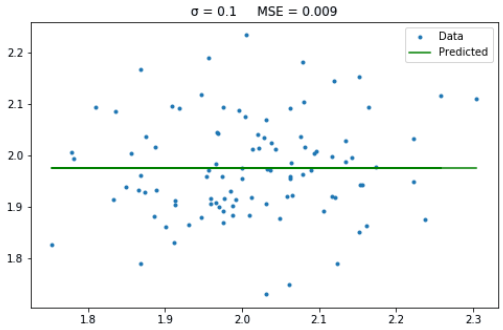


When β_1 is set to 0:

In this case, slope is zero and hence the predicted value is always the value of intercept which is the mean of vector y. Varying the value of σ, increases the value of y and thereby increases the MSE of the model.



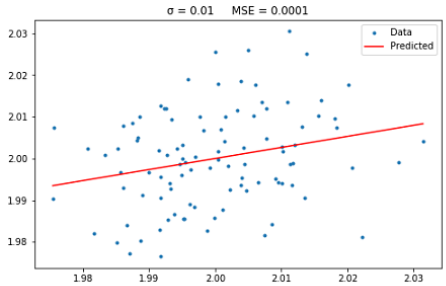


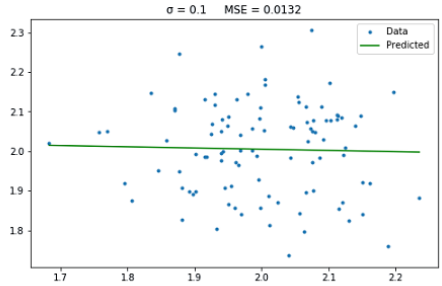


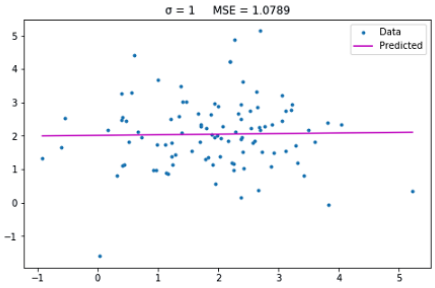
Function Name : LinReglstsq
Parameter : Matrix

This is a function which uses the pre-defined libraries using numpy to find the values of slope and intercept from the training values which is further used to predict the values.

```
b1,b0 = np.linalg.lstsq(A,y)[0]
Matrix_predict = predict(b0,b1,Matrix)
MSE = calculateMSE(Matrix,Matrix_predict)
```







From the graph, we can infer that the predicted values from both the inbuilt function numpy.linalg.lstsq and implemented code for linear regression are same and so will be the values of slope, intercept and errors.