**Assessment assignment for DS course at Spiced Academy.**

By Esenalieva Erkingul

March 28, 2022.

***Preamble***.

This is my first experience performing a mock analysis of a given data using Python. The knowledge of Python I have gained within the last two weeks is down to minimal basics, thus, I have used Google to search for the possible coding solutions to implement them with the purpose to accomplish the tasks.

In the assessment you will implement a linear model for a set of [20 x/y data points](https://spiced.space/datapoints.csv).

We assume that the data can be described by a straight line with the slope *a* through the origin and an intercept b. *y = a \* x + b*

***Task*** *1*

*Read the x/y data points from the file datapoints.csv into Python.*

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*import csv*

*#with open ("datapoints.csv", r) as data:*

*# Task1. Read the csv file*

*data = pd.read\_csv("datapoints.csv")*

*print(data.shape)*

*print(data.head)*

*Outcome*

*(20, 2)*

*<bound method NDFrame.head of x y*

*0 0.000000 0.496714*

*1 0.052632 0.019630*

*2 0.105263 0.963478*

*3 0.157895 1.996714*

*4 0.210526 0.397426*

*5 0.263158 0.555337*

*6 0.315789 2.526581*

*7 0.368421 1.872698*

*8 0.421053 0.793684*

*9 0.473684 1.963613*

*10 0.526316 1.115530*

*11 0.578947 1.271112*

*12 0.631579 2.136699*

*13 0.684211 0.139351*

*14 0.736842 0.485608*

*15 0.789474 1.806134*

*16 0.842105 1.513485*

*17 0.894737 2.998458*

*18 0.947368 1.934081*

*19 1.000000 1.587696>*

*Additional task 1.1. Correlation between X and Y*

*# Correlation between X and Y*

*cor\_xy = data.corr()*

*print(cor\_xy)*

*Outcome*

*x y*

*x 1.000000 0.414301*

*y 0.414301 1.000000*

*Interpretation: We observe that the correlation between X and Y is 0.4143 and lies at the border between weak (0.2-0.4) and the moderate (0.4 - 0.6) correlation. These two variables are weakly correlated with each other.*

***Task*** *2*

*Create a scatterplot of the data.*

*#Task2. Create a Scatterplot*

*x = data["x"]*

*y\_true = data["y"]*

*#Scatterplot for X and Y*

*plt.title("Scatterplot for X and Y")*

*plt.xlabel("X variable")*

*plt.ylabel("Y\_true variable")*

*plt.scatter(x,y\_true)*

*plt.show()*

*Outcome*

*See attachment for the scatterplot.*

***Task*** *3*

*Set the slope a to 10 and the intercept b to 0. Calculate y for every value of x.*

*Y\_pred = 10\*X + 0 = 10\*X*

*#Task3. Set the slope a to 10 and the intercept b to 0. Calculate y\_pred for every value of x*

*y\_pred = [item\*10 for item in x]*

*# displaying X and Y\_predicted*

*list\_X = x*

*list\_Y = y\_pred*

*print("X | Y\_predicted")*

*for col\_X, col\_Y in zip(list\_X, list\_Y):*

*print(col\_X , "|", col\_Y)*

*Outcome*

*X | Y\_predicted*

*0.0 | 0.0*

*0.0526315789473684 | 0.526315789473684*

*0.1052631578947368 | 1.052631578947368*

*0.1578947368421052 | 1.578947368421052*

*0.2105263157894736 | 2.105263157894736*

*0.2631578947368421 | 2.631578947368421*

*0.3157894736842105 | 3.1578947368421053*

*0.3684210526315789 | 3.6842105263157894*

*0.4210526315789473 | 4.210526315789473*

*0.4736842105263157 | 4.7368421052631575*

*0.5263157894736842 | 5.263157894736842*

*0.5789473684210527 | 5.7894736842105265*

*0.631578947368421 | 6.315789473684211*

*0.6842105263157894 | 6.842105263157894*

*0.7368421052631579 | 7.368421052631579*

*0.7894736842105263 | 7.894736842105264*

*0.8421052631578947 | 8.421052631578947*

*0.894736842105263 | 8.94736842105263*

*0.9473684210526316 | 9.473684210526317*

*1.0 | 10.0*

***Task*** *4*

*Calculate the Mean Squared Error (MSE) of y\_true and y\_pred using the formula:*

*MSE = 1/N∑(y\_true - y\_pred)2*

*# Task4. MSE calculation*

*from sklearn.metrics import mean\_squared\_error*

*mse = mean\_squared\_error(y\_true, y\_pred) # MSE calculated*

*print("MSE at a=10 and b=0 - ", mse) # MSE coded outcome confirmed by the semi-manual calculation in Excel*

*Outcome*

*MSE at a=10 and b=0 - 21.306499419071013*

***Task*** *5*

*Find a value for a that gives the lowest possible MSE.*

*Implement the following procedure:*

* *initially set a to 10*
* *repeat the following procedure 100 times:*
  + *decrease a by 0.1*
  + *re-calculate y using the modified a*
  + *re-calculate the MSEcheck if the new MSE is smaller than the previous one*
  + *if it is smaller, keep the new values for the MSE and a, otherwise discard it*
  + *print the final value for a and the corresponding MSE*

*#Task5. Find a value for a that gives the lowest possible MSE.*

*a = 10*

*tmp\_MSE = np.infty*

*tmp\_a = a*

*for i in range(100):*

*Y\_2 = (a-0.1\*(i-1))\*x*

*tmp\_a = (a-0.1\*(i-1))*

*MSE = np.square(np.subtract(y\_true, Y\_2)).mean()*

*if MSE < tmp\_MSE: # MSE comparison*

*tmp\_MSE = MSE*

*tmp\_A = tmp\_a*

*print("A for the minimal MSE - ", tmp\_A)*

*print("Minimal MSE - ", tmp\_MSE)*

*Outcome*

*A for the minimal MSE - 2.1999999999999993*

*Minimal MSE - 0.7080422372498429*

***Task*** *6*

*Also modify b in the above procedure.*

*#Task6. Repeat the procedure with b (initially b = 0)*

*b = 0*

*tmp\_MSE = np.infty*

*tmp\_b = b*

*for i in range(100):*

*Y\_pred = 10\*x+(0-0.1\*(i-1))*

*tmp\_b = (0-0.1\*(i-1))*

*MSE = np.square(np.subtract(y\_true, Y\_pred)).mean()*

*if MSE < tmp\_MSE: # temporary MSE comparison*

*tmp\_MSE = MSE*

*tmp\_B = b - tmp\_b*

*print("B for the minimal MSE - ", tmp\_B)*

*print("The minimal MSE - ", tmp\_MSE)*

*Outcome*

*B for the minimal MSE - 3.7*

*The minimal MSE - 7.828890064401479*

***Task*** *7*

*How could the algorithm be improved? Write down one or two ideas.*

1. *The coding solutions could be more elegant and more efficient. If I knew how, I would use them.*
2. *In task 3 I could not find a better solution to display X and Y\_predicted in the form of a table with names of the columns. The coding solution looks rather primitive.*
3. *Running the scatter plot causes the program PyCharm to slow down. I had to isolate this part in order to run the rest of the coding. How to do it better, I do not know.*