**Assignment 2**

**Instructions:**

* Type your answers in the spaces provided in this Word document. Your submission should not exceed 11 pages, including this page.
* Submit the *Declaration of Academic Integrity* before submitting your assignment.

--------------------------------------------------------------------------------------------------------------------

**Introduction**

Given a set of data points with at least one predictor and one continuous response variable, we want to construct a linear model to predict the response. This is the aim of **Linear Regression**, which is a supervised learning technique.

In the context of this assignment, the transaction price of 30 flats in the same district of Singapore are collected. The data can be found in the file *housing\_price.csv*.

The response variable is *price per square metre* (measured in $ in thousands)*,* and the predictors are *inverse age of flat* (measured in year-1)and *inverse* *distance to the nearest MRT station* (measured in km-1). The *inverse age of flat* and *inverse* *distance to the nearest MRT station* are derived fields.

**Simple Linear Regression (SLR)**

We will first build a SLR model using *inverse* *distance to the nearest MRT station* as the predictor to predict *price per square metre*.

In SLR notations, let:

= predictor value of the *i*-th data point

= actual response value of the *i*-th data point

= predicted response value of the *i*-th data point based on model

Thus, , where values of *a* (intercept) and *b* (slope) are to be determined.

The squared-error of the *i*th prediction is . Errors (also known as residuals) are squared to remove the signs, so that errors of opposite signs do not cancel out each other, giving the false impression of small aggregated errors.

Then, we define **Error function** as the mean of squared-error (of the whole data set):

We want to find the values of *a* and *b* such that the Error function is **minimised**.

The resultant equation will give the best-fit line that passes through the data points.

**MODEL 1: SLR with intercept *a* fixed ⇒**  (25 marks)

We will first build a SLR model to predict *price per square metre* (*y*) using *inverse distance to the nearest MRT station* (*x*) as the predictor.

Suppose it is believed that price is proportional to inverse distance. This means that is a constant multiple of and . Then, in the SLR model, we will only need to determine slope *b*.

(a) Express Error function in terms of *b* only. Hence, derive .

|  |
| --- |
| where number of data points  where number of data points |

(b) Use univariate gradient descent algorithm to find the value of *b* for which is at its minimum. Write your Python code in a single cell and copy-paste your code below.

|  |
| --- |
| import numpy as np import pandas as pd import matplotlib.pyplot as plt from matplotlib.pyplot import figure from termcolor import colored df = pd.read\_csv('housing\_price.csv')   x = df[list(df.columns)[1]] y = df[list(df.columns)[2]] b = 1 # Starting value of b rate = 0.001 # Set learning rate epsilon = 0.000001 # Stop algorithm when absolute difference between 2 consecutive x-values is less than epsilon diff = 1 # difference between 2 consecutive iterates max\_iter = 2000 # set maximum number of iterations iter = 1 # iterations counter  def e\_b(*x*,*y*,*b*):  mse = 0  for (xi,yi) in zip(*x*,*y*):  mse += (yi-*b*\*xi)\*\*2  mse = mse/30  return mse  def e\_deriv(*x*,*y*,*b*):  deriv = 0  for (xi,yi) in zip(*x*,*y*):  deriv += (yi-*b*\*xi)\*(-2\*xi)  deriv = deriv/30  return deriv   # Now Gradient Descent  while diff > epsilon and iter < max\_iter:  b\_new = b - rate \* e\_deriv(x,y,b)  print("Iteration ", iter, ": b =", round(b\_new,4),"| MSE =", round(e\_b(x,y,b\_new),4) )  diff = abs(e\_b(x,y,b\_new) - e\_b(x,y,b))  iter = iter + 1  b = b\_new  print("Minimum Value of B:", colored(b, 'green'), "MSE:",colored(e\_b(x,y,b), 'green')) |

(c) Describe the changes and decisions you made on the parameters for your solution to reach convergence.

|  |
| --- |
| The calculation of the error function required 3 variables:  This meant that I had to create my own custom error function and derivative function to account for the summation formula in the code.  Additionally, instead of comparing the difference in slope value (b) against the epsilon, I compared the difference of MSE value.  This helps me fine tune the model against the Mean Squared Error and provide a consistent value to compare against when I incorporate more than 1 slope later in Multiple Linear Regression  I also added the final results to be printed in separate lines and different colours so that it was easy to interpret the results. |

(d) Describe your MODEL 1 by filling the information below.

|  |
| --- |
| Final MODEL 1 equation is:  Minimum value of Error function is: 19.467  Number of iterations ran to reach convergence: 59 |

**MODEL 2: SLR ⇒**  (25 marks)

Now we apply the SLR model where both intercept *a* and slope *b* are to be determined, when predicting *price per square metre* (*y*) using *inverse* *distance to the nearest MRT station* (*x*) as the predictor.

(a) Express Error function in terms of *a* and *b*. Hence, derive and .

|  |
| --- |
| where n = number of data points  =  = |

(b) Use gradient descent algorithm to find the values of *a* and *b* for which is at its minimum. Write your Python code in a single cell and copy-paste your code below.

|  |
| --- |
| import numpy as np import pandas as pd import matplotlib.pyplot as plt from matplotlib.pyplot import figure from termcolor import colored import sympy as sp from sympy import \* df = pd.read\_csv('housing\_price.csv')  # data points x = df[list(df.columns)[1]] y = df[list(df.columns)[2]]  next\_a = 5 # Initial point next\_b = 0 # Initial point alpha = 0.001 # Learning rate epsilon = 0.00000001 # Stopping criterion constant max\_iters = 100000 # Maximum number of iterations   # Error Function def e\_ab(*x*,*y*,*a*,*b*):  mse = 0  for (xi,yi) in zip(*x*,*y*):  mse += ((yi-(*a*+*b*\*xi))\*\*2)  mse = mse/30  return mse   # Derivative Function def a\_deriv(*x*,*y*,*a*,*b*):  deriv = 0  deriv = sum(-2\*(*y*-*a*-*b*\**x*))  deriv = deriv/30  return deriv   # Derivative Function def b\_deriv(*x*,*y*,*a*,*b*):  deriv = 0  deriv = sum(-2\**x*\*(*y*-*a*-*b*\**x*))  deriv = deriv/30  return deriv  # Initial value of error function next\_func = e\_ab(x,y,next\_a,next\_b)   for n in range(max\_iters):  current\_a = next\_a  current\_b = next\_b  current\_func = next\_func  # update a  next\_a = current\_a-alpha\*a\_deriv(x,y,current\_a,current\_b)  # update b  next\_b = current\_b-alpha\*b\_deriv(x,y,current\_a,current\_b)  next\_func = e\_ab(x,y,next\_a,next\_b)  change\_func = abs(next\_func-current\_func) # stopping criterion: values of function converge  print("Iteration",n+1,": a = ",next\_a,", b = ",next\_b,", f(a,b) = ",next\_func)  if change\_func<epsilon:  print("Minimum Value of a:", colored(next\_a, 'green'),"\nMinimum Value of b:", colored(next\_b, 'green'), "\nMSE:",colored(next\_func, 'green'))  break |

(c) Describe the changes and decisions you made on the parameters for your solution to reach convergence.

|  |
| --- |
| I changed the function names and added individual functions for the derivative of A and derivative of B respectively.  This allowed me to calculate the partial derivative more conveniently without having to bloat the code with lambda functions, thus increasing readability  Additionally, I lowered the learning rate (∂) and epsilon value to make sure that the MSE was at the  lowest.  I also added the final results to be printed in separate lines and different colours so that it was easy to interpret the results. |

(d) Describe your MODEL 2 by filling the information below.

|  |
| --- |
| Final MODEL 2 equation is:  Minimum value of Error function is: 0.462  Number of iterations ran to reach convergence: 4863 |
|  |

**Conclusion on SLR** (15 marks)

(a) Using Python (or other software), in a single figure, plot the data points (scatterplot) together with the linear lines representing the two models. Insert the figure below and describe what you observe regarding the location of the data and the linear lines.

|  |
| --- |
| When we look at the scatter plot data, we can see that it resembles that of a log curve. There is very quick growth initially, but over time, the rate of growth decreases. So therefore, the rate of change of price per sqft decreases with time.  The red line demonstrates the gradient descent-tuned slope value that produces the lowest MSE.  However, since we assume that the line starts at 0, it is not a good representation of the data as there is no data near 0 for any axes, so it is not reasonable to interpolate these values.  When we look at the green line, where we perform a gradient descent and tune both the slope and y intercept values, we can see that the new line performs much better with the data points.  However, since it is a linear equation, it is unable to account for the initial exponential growth at the beginning, and this is evident in the plot. We can observe that the green line is able to fit well with values that come after the initial exponential growth, at roughly |

(b) In a linear regression model, the constant  is commonly interpreted as the value of the response variable when the predictor variable is zero. In your Model 2, can you interpret your value of  as such? Explain.

|  |
| --- |
| Since the value of *x* is inverse distance to the nearest MRT, it is not mathematically possible for the constant value to be 0. Additionally, there will be no housing at no distance from the MRT, as the 0 distance from the MRT means we are referring to the MRT itself.  Hence, it is both mathematically and logically impossible for the predictor variable to be 0. And therefore, we cannot interpret our value of *a* as such for Model 2. |

**MODEL 3: MLR ⇒**  (25 marks)

We can extend the SLR model to include more predictors. A linear regression model with more than 1 predictor is called **Multiple Linear Regression** (MLR) model.

Apply the MLR model where intercept *a*, and slopes *b* and *c* are to be determined, when predicting *price per square metre* (*y*) using *inverse* *distance to the nearest MRT station* (*x*) and *inverse age of flat* (*w*) as the predictors.

(a) Explain how gradient descent algorithm can be extended for MODEL 3.

|  |
| --- |
| =  =  =  I added an additional parameter *c* to account for the new variable in the equation “inverse age of flat”. I then calculated the partial derivative for the Error Function in respect to *c*. |

(b) Use gradient descent algorithm to find the values of *a*, *b* and *c* for which Error function is at its minimum. Write your Python code in a single cell and copy-paste your code below.

|  |
| --- |
| import numpy as np import pandas as pd import matplotlib.pyplot as plt from matplotlib.pyplot import figure from termcolor import colored import sympy as sp from sympy import \* df = pd.read\_csv('housing\_price.csv')  # data points x = df[list(df.columns)[1]] y = df[list(df.columns)[2]] w = df[list(df.columns)[0]]  next\_a = 4 # Initial point next\_b = 4 # Initial point next\_c = 15 # Initial point  alpha = 0.0001 # Learning rate epsilon = 0.00001 # Stopping criterion constant max\_iters = 100000 # Maximum number of iterations   # Error Function def e\_abc(*x*,*y*,*w*,*a*,*b*,*c*):  mse = 0  mse = sum((*y*-*a*-*b*\**x*-*c*\**w*))\*\*2  mse = mse/30  return mse   # Derivative Function def a\_deriv(*x*,*y*,*w*,*a*,*b*,*c*):  deriv = 0  deriv = sum((*y*-*a*-*b*\**x*-*c*\**w*))  deriv = -2\*deriv/30  return deriv   # Derivative Function def b\_deriv(*x*,*y*,*w*,*a*,*b*,*c*):  deriv = 0  deriv = sum((*y*-*a*-*b*\**x*-*c*\**w*)\**x*)  deriv = -2\*deriv/30  return deriv  # Derivative Function def c\_deriv(*x*,*y*,*w*,*a*,*b*,*c*):  deriv = 0  deriv = sum((*y*-*a*-*b*\**x*-*c*\**w*)\**w*)  deriv = -2\*deriv/30  return deriv  # Initial value of error function next\_func = e\_abc(x,y,w,next\_a,next\_b,next\_c)   for n in range(max\_iters):  current\_a = next\_a  current\_b = next\_b  current\_c = next\_c  current\_func = next\_func  # update a  next\_a = current\_a-alpha\*a\_deriv(x,y,w,current\_a,current\_b,current\_c)  # update b  next\_b = current\_b-alpha\*b\_deriv(x,y,w,current\_a,current\_b,current\_c)  # update c  next\_c = current\_c-alpha\*c\_deriv(x,y,w,current\_a,current\_b,current\_c)  next\_func = e\_abc(x,y,w,next\_a,next\_b,next\_c)   # stopping criterion: values of function converge  change\_a = abs(current\_a-next\_a)  change\_b = abs(current\_b-next\_b)  change\_c = abs(current\_c-next\_c)  print(  "Iteration",  n+1,  ": a = ",  next\_a,  ", b = ",  next\_b,  ", c = ",  next\_c,  " f(a,b,c) = ",  next\_func)  if change\_a < epsilon and change\_b < epsilon and change\_c < epsilon:  print("Minimum Value of a:",  colored(next\_a, 'green'),  "\nMinimum Value of b:",  colored(next\_b, 'green'),  "\nMinimum Value of c:",  colored(next\_c, 'green'),  "\nMSE:",colored(next\_func, 'green'))  break |

(c) Describe the changes and decisions you made on the parameters for your solution to reach convergence.

|  |
| --- |
| I changed the stopping criterion to account for the new *c* value. This is because the values for *a, b* and *c* are very sensitive to small changes, and any little change in the value can cause the MSE to swing  around wildly in the other direction, going from 5 to 0.2 to 0.05 then up to 13.  This is because the values from the dataset are not scaled. The range for *a* is about 20, whereas the  range for *c* is roughly 0.01. This means that similar changes in their respective values can cause a very different MSE.  Hence, it is important for us to calculate it based on each variable’s local scale, and therefore we  measure the difference in each variables change and ensure that the values for all variables have  converged. |

(d) Describe your MODEL 3 by filling the information below.

|  |
| --- |
| Final MODEL 3 equation is:  Minimum value of Error function is: 0.0750  Number of iterations ran to reach convergence: 20176 |

**Conclusion** (10 marks)

(a) David used gradient descent algorithm to find the 3 models. Next, he computed the predicted housing prices using the 3 models for all the data points in the dataset. He noticed that for one of the data points, the error of the predicted housing price in Model 1 from the actual housing price is the smallest, compared to the other 2 models. Is this possible, assuming he has done his gradient descent algorithm correctly? Explain.

|  |
| --- |
| This is possible since the price for the houses do not follow a linear pattern. Therefore, if we were to compare the error for a datapoint, it is highly likely that he selected a data point near the start, where the inverse distance to nearest MRT was low. This meant that the data point was in the initial stages of the exponential growth as mentioned above. As such, the data point that was predicted might have been closer to Model 1 as the y intercept for Model 2 is higher up that that of Model 1, therefore resulting in a lower error.  Additionally, the error function may have different local minimums and maximums, and this is dependent on the initial value of the function that he initiated. Hence, there is a possibility that the initial values that he used for Model 1 was near the local minimum, while the initial values used for Model 2 were not near the local minimum. This would then result in the error value for Model 1 being a lot lower.  Additionally, he may have used different epsilon and learning rate values, thus affecting the performance of the mode. |

(b) Compare the 3 models. Which model will you use to predict housing price in this context? Explain.

|  |
| --- |
| I will use Model 3. Since we have more columns, this allows us to use more data and gain a greater insight as to the representation of the data.   This allows the model to be more well-rounded and well-informed, and it allows it to make better predictions overall.  This can be seen in the Mean Square Error (MSE) of each of the models. We can observe that the more relevant information that is added to the equation, the lower the MSE.  This suggests that the dataset that was used was relevant and that these features can be used to generate more accurate predictions. |