**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 .

|  |
| --- |
| n=30  E(b)=  Substitute  E(b)=  E(b) = 52.9560529666666*b*2−86.1205782*b*+54.4808258  E’(b)=  E’(b)= 105.912105933333*b*−86.1205782 |

(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.

|  |
| --- |
| b = sp.symbols('b')  rate = 0.01 #Set Learning Rate  epsilon = 0.0001 #Stop algorithm when absolute difference between 2 consecutive x-values is less than equation  max\_iter = 1000 # maximum number of iterations  f = lambda b: 52.9560529666666\*b\*\*2-86.1205782\*b+54.4808258  deriv = lambda b: 105.912105933333\*b -86.1205782# derivative of f  def gradient\_desc(x):      diff=1 # differenece between 2 consecutive iterates      iter = 1 #iteration counter      #Now Gradient descent      while diff > epsilon and iter < max\_iter:          x\_new = x - rate\*deriv(x)          diff = abs(x\_new - x)          f\_value = f(x\_new)          iter += 1          x = x\_new      print("The local minimum occurs at x = ",round(x\_new, 3),"\nNumber of iterations: ",iter,"\nf(x) = ",round(f\_value, 3))  gradient\_desc(0.8) |

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

|  |
| --- |
| Getting a rough start x-value: 0 since the graph shows that the minimum error is around -2.5 to 2.5.  I then proceeded perform gradient descent ,x=0, alpha = 0.01 and epsilon value of 0.0001. The local minimum occurs at x =0.81313311, f(x) = 19.4671.  I then proceeded to find where the best starting x-value should be based on the least f-value(error) and least iterations.    As shown above x-value(starting value of 0.8 is the best)  I then proceeded to tune the alpha value such that iterations is at minimum.    A good value would be alpha = 0.009 or alpha = 0.01  Finally, I chose alpha = 0.01 which have iterations of 4. The answer below is my final code after tuning. |

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

|  |
| --- |
| Final MODEL 1 equation is: y\_hat = 0 + 0.8131  Minimum value of Error function is: 19.4671  Number of iterations ran to reach convergence: 4 |

**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 .

|  |
| --- |
| E(a,b)=  E(a,b): 1.0\*a\*\*2 + 9.48166666666667\*a\*b - 14.3246666666667\*a + 52.9560529666666\*b\*\*2 - 86.1205782\*b + 54.4808258  E’(a)=  E'(a): 2.0\*a + 9.48166666666667\*b - 14.3246666666667  E’(b) =  E'(b): 9.48166666666667\*a + 105.912105933333\*b - 86.1205782 |

(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.

|  |
| --- |
| *# testing for best alpha*  import numpy as np  alpha = 0.01 *# learning rate*  epsilon = 0.0001 *# stopping criterion*  max\_iters = 1000 *# maximum number of iterations*  *# partial derivatives and function*  next\_a = sp.symbols('a')  next\_b = sp.symbols('b')  next\_a = 5.7 *# initial a value*  next\_b = 0.3 *# initial b value*  func = lambda a,b: 1.0\*a\*\*2 + 9.48166666666667\*a\*b - 14.3246666666667\*a + 52.9560529666666\*b\*\*2 - 86.1205782\*b + 54.4808258  partialf\_a = lambda a,b: 2.0\*a + 9.48166666666667\*b - 14.3246666666667*# partial derivative of f with respect to a*  partialf\_b = lambda a,b: 9.48166666666667\*a + 105.912105933333\*b - 86.1205782*# partial derivative of f with respect to b*  diff = 1  iter =1  next\_func = func(next\_a, next\_b) *# initial value of function*  while diff > epsilon and iter<max\_iters:      current\_a = next\_a      current\_b = next\_b      current\_func = next\_func      next\_a = current\_a - alpha\*partialf\_a(current\_a, current\_b) *# update of a*      next\_b = current\_b - alpha\*partialf\_b(current\_a, current\_b) *# update of b*      next\_func = func(next\_a, next\_b)      diff = abs(next\_func - current\_func) *# stopping criterion values of function converges*      iter += 1  print("The local minimum occurs at a = ",round(next\_a, 4),"b =",round(next\_b,4),"\nf(x) = ",round(next\_func, 4),"\nIter = ",iter) |

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

|  |
| --- |
| Based on initial glance at data, I estimated that the starting point for intercept ‘a’ =5 and gradient ‘b’ = 0.3 . I estimated the gradient by taking (11-5)/20 = 0.3  I tested this starting point with gradient descent and got:  The local minimum occurs at a = 5.4707 b = 0.3236 f(x) = 0.5058  I then proceeded to create a dataframe and calculated the Mean Squared error for each a and b  a = np.arange(-10,11,0.1)  b = np.arange(-10,11,0.1)    Based on the above 3d plot and getting minimum error function, I can have my starting point a =5.75 and b =0.31 and used it to tune my alpha values by looping through a range of alpha.  alpha = np.linspace(0,0.01,100)  I took those alpha values that gave below 0.47 mean squared error and also the minimum iterations.  It took 3 iterations from starting point a =5.75 and b=0.31 |

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

|  |
| --- |
| Final MODEL 2 equation is: y\_hat = 5.7192 + 0.3012  Minimum value of Error function is: 0.4621  Number of iterations ran to reach convergence: 3 |
|  |

**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.

|  |
| --- |
| Predictions of Model 1 with intercept = 0 is much further away from the actual data points  While predictions with Model 2 fits the actual data points much better which can give a more accurate prediction of the data. |

(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.

|  |
| --- |
| If it is based on Model 2, I think ‘a’ may be interpreted as 0 since it gives a predicted value which is not 0. Furthermore, there are indeed flats which are build right on top of MRT such as North Park Residences.  However, I also do not think we can interpret the value of ‘a’ as such if predictor variable is zero.  1/x= 0  x = 0/1  x is undefined  The main reason I would argue so is because (distance from nearest Mrt) should not be undefined. Thus, it cannot be fitted into any sort of algorithm that takes in that as a continuous variable.  Example would be: Predicted\_Value = 5.7013 + 0.3028\*undefined  Which cannot be defined unless we treat it as a 0.  In conclusion, there are no free or flats with undefined price in Singapore which just doesn’t make sense but since Model 2 would predict $5.7013thousand/sq meter when x is treated as a 0. I would take 5.7013 as the prediction but it is also not right to interpret x as 0 . |

**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 extended gradient descent algorithm by including the 3rd predictor (‘inverse age of flat (w)’)  As for the algorithm I simply added a 3rd partial derivative. I will explain my decision by first explaining how gradient descent worked for Model 1 and 2.      Gradient Descent is an iterative algorithm which aims to find the minimum of a function.  Looking at Model 1, It finds the tangent line of the function. Since it is 2 dimensional with only 1 parameter ‘b’, it is pretty straightforward by only finding the derivative where it is = 0    Looking at Model 2,it is the same concept of finding the minimum point. However, there are 3-dimensions with 2 parameters now. The algorithm cannot move both parameters at once, even if it could have done so, it might not get the correct minimum of the function.    We are still interested in the tangent line, but we can only find the tangent for each parameter one by one  Partial derivatives does just that by holding other variables constant while changing one.  Thus, there is a need for partial derivatives.    *Fig 1 Visualization of partial derivative*  *A Visualization of partial derivative (Misa.,O 2017) shows that it keeps θ2 constant(blue line) while finding the minimum point.*  Misa.,O 2017 *‘Intuition (and maths!) behind multivariate gradient descent’* Available at: <https://towardsdatascience.com/machine-learning-bit-by-bit-multivariate-gradient-descent-e198fdd0df85> [Accessed 8/25/2022]  Therefore, when deciding how to extend gradient descent for Model 3 which has a 3rd variable, I just added a 3rd partial derivative to find the minimum of the function. The gradient descent algorithm iteratively finds the minimum by holding 2 variables constant while moving 1 till it finds the minimum. It works the same for higher dimensional functions. |

(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.

|  |
| --- |
| diff = 1  iter =1  next\_a = 4.5  next\_b = 1.4  next\_c = 17.0  epsilon = 0.0001 *# stopping criterion*  max\_iters = 10000 *# maximum number of iterations*  alpha =0.01  func = lambda a,b,c: 1.0\*a\*\*2 + 9.48166666666667\*a\*b + 0.226733333333333\*a\*c - 14.3246666666667\*a + 52.9560529666667\*b\*\*2 + 1.5078038\*b\*c - 86.1205782\*b + 0.0154496333333333\*c\*\*2 - 1.78927586666667\*c + 54.4808258  partialf\_a = lambda a,b,c: 2.0\*a + 9.48166666666667\*b + 0.226733333333333\*c - 14.3246666666667  *# partial derivative of f with respect to a*  partialf\_b = lambda a,b,c: 9.48166666666667\*a + 105.912105933333\*b + 1.5078038\*c - 86.1205782 *# partial derivative of f with respect to b*  partialf\_c = lambda a,b,c: 0.226733333333333\*a + 1.5078038\*b + 0.0308992666666667\*c - 1.78927586666667*# partial derivative of f with respect to c*  next\_func = func(next\_a,next\_b,next\_c) *# initial value of function*  while diff > epsilon and iter<max\_iters:      current\_a = next\_a      current\_b = next\_b      current\_c = next\_c      current\_func = next\_func      next\_a = current\_a - alpha\*partialf\_a(current\_a, current\_b, current\_c) *# update of a*      next\_b = current\_b - alpha\*partialf\_b(current\_a, current\_b, current\_c) *# update of b*      next\_c = current\_c - alpha\*partialf\_c(current\_a, current\_b, current\_c) *# update of c*      next\_func = func(next\_a, next\_b, next\_c)      diff = abs(next\_func - current\_func) *# stopping criterion values of function converges*      iter += 1  print("The local minimum occurs at a = ",round(next\_a, 4),"b =",round(next\_b,4),"c =",round(next\_c, 4), "\nf(x) = ",round(next\_func, 4),"\nIter = ",iter) |

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

|  |
| --- |
| Initially I went straight to using gradient descent trying out where a =1 b =1 and c=1 but I did not know where would be a good starting value and I also realised different various starting values produces different minimums and I will have to do a exhaustive search to look for the global minimum.  I used brute force method where I use a range of 0 to 50 for a , b and c as starting points for gradient descent to find out roughly where the best starting point will be. I also rejected those where a or b or c = 0 because I felt that if any of them were 0, it would not be significant if they were 0 since we are trying to predict with all predictor variables  The best starting points to achieve minimum was a= 4.5 b= 1.4 and c = 17.0  I then proceeded to find the best alpha value by looping through and got 0.01 as it gave the least iterations and lowest mean squared error. |

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

|  |
| --- |
| Final MODEL 3 equation is: y\_hat = 4.3896 +0.1784+16.9825  Minimum value of Error function is: 0.1562  Number of iterations ran to reach convergence: 5 |

**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.

|  |
| --- |
| No it is unlikely that it is possible.  It is not possible because the mean squared error of Model 1 is just too big as compared to Model 3 and Model 2.    Model 2 also covers most of the datapoint  On top of that, I did a for loop to check for this dataset whether it was possible and turns out Model 1 was never the best for all datapoints. |

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

|  |
| --- |
| I would use Model 3 to predict the housing price. Most of the time it generates the most accurate price in thousands per sq meter and since it considers the area of the building, it would be a better model to be used to predict housing price. |