**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

1. Thus, , where values of *a* (intercept) and *b* (slope) are to be determined.
2. 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 .

|  |
| --- |
| Since  E(b) =  E’(b) = |

(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 = 1 # Starting value  rate = 0.005 # Set learning rate  precision = 0.005 # Stop algorithm when absolute difference between 2 consecutive x-values is less than precision  diff = 1 # difference between 2 consecutive iterates  max\_iter = 1000 # set maximum number of iterations  iter = 1 # iterations counter  y = df['price per square metre ($ in thousands)']  x = df['inverse distance to the nearest MRT station (km-1)']  def f(b):  # for i in range (30):  err = 1/30 \* np.sum((y - b \* x)\*\*2)  return err  def deriv(b):  # for i in range(30):  der = 1/15 \* np.sum(-1 \* x \* (-b \* x + y))  return der  # Gradient Descent  while diff > precision and iter < max\_iter:  b\_new = b - rate \* deriv(b)  print("Iteration ", iter, ": b-value is: ", b\_new,"f(b) is: ", f(b\_new) )  diff = abs(b\_new - b)  iter = iter + 1  b = b\_new    print("The local minimum occurs at: ", b) |

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

|  |
| --- |
| I changed both the learning rate and precision from 0.01 to 0.05. I kept the starting value of b=1 as well as the diff and the max\_iter which remains the same. For this code we have function f for E(b) and function deriv for E’(b) . |

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

|  |
| --- |
| Final MODEL 1 equation is: ( all to 3s.f.)  Minimum value of Error function is: 19.5 (3s.f.)  Number of iterations ran to reach convergence: 5 |

**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) =  =  = |

(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.

|  |
| --- |
| y = df['price per square metre ($ in thousands)']  x = df['inverse distance to the nearest MRT station (km-1)']  a = 5 # Initial point  b = 0 # Initial point  alpha = 0.01 # Learning rate  epsilon = 0.0001 # Stopping criterion constant  max\_iters = 500 # Maximum number of iterations  def partiala(a,b):  return -1/15 \* np.sum(y - a - (b \* x))  def partialb(a,b):  return -1/15 \* np.sum(x\*(y - a - (b \* x)))  def funcab(a,b):  return 1/30 \* np.sum((y - a - (b \* x))\*\*2)  next\_func = funcab(a,b) # Initial value of function  print(next\_func)  for n in range(max\_iters):  current\_a = a  current\_b = b  current\_func = next\_func  a = current\_a-alpha\*partiala(current\_a,current\_b) # update of a  b = current\_b-alpha\*partialb(current\_a,current\_b) # update of b  next\_func = funcab(a,b)  change\_func = abs(next\_func-current\_func) # stopping criterion: values of function converge  print("Iteration",n+1,": a = ",a,", b = ",b,", f(a,b) = ",next\_func)  if change\_func<epsilon:  break    print("The local minimum occurs at: ",next\_func) |

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

|  |
| --- |
| I changed the starting values of a and b from 4 to 5 and 5 to 0 respectively. I kept the alpha and the max\_iters the same. I also changed the epsilon from 0.01 to 0.0001.For this code, we have a partiala for the partial derivative and partial for the partial derivative . funcab is for E(a,b). |

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

|  |
| --- |
| Final MODEL 2 equation is: (all to 3s.f.)  Minimum value of Error function is: 0.466 (3s.f.))  Number of iterations ran to reach convergence: 184 |

**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.

|  |
| --- |
| The linear graph for model 2 seems to be a best fit line that passes through 2 data points and has almost equal number of data points above and below the line whereas the linear graph for model 1 is not a best fit line and does not pass through any data point. |

(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 equation for model 2 is , then supposedly, if = 0,  However, we must see the context for this dataset. It will not make sense in this context for to have a value if = 0 since is the predictor. Therefore, we cannot interpret the value of a as such. |

**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.

|  |
| --- |
| E(a,b,c) =  =  =  =  We will now be adding w into the equation and I will be adding the partial derivative of c as well as current\_c and use the same algorithm to get the minimum value and get the values of a, b and c respectively. |

(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.

|  |
| --- |
| y = df['price per square metre ($ in thousands)']  x = df['inverse distance to the nearest MRT station (km-1)']  w = df['inverse age of flat (year-1)']  a = 5 # Initial point  b = 0 # Initial point  c = 15 # Initial point  alpha = 0.01 # Learning rate  epsilon = 0.0001 # Stopping criterion constant  max\_iters = 500 # Maximum number of iterations  def partiala(a,b,c):  return -1/15 \* np.sum(y - a - (b \* x) - (c \* w))  def partialb(a,b,c):  return -1/15 \* np.sum(x\*(y - a - (b \* x) - (c \* w)))  def partialc(a,b,c):  return -1/15 \* np.sum(w\*(y - a - (b \* x) - (c \* w)))  def funcabc(a,b,c):  return 1/30 \* np.sum((y - a - (b \* x) - (c \* w))\*\*2)  next\_func = funcabc(a,b,c) # Initial value of function  print(next\_func)  for n in range(max\_iters):  current\_a = a  current\_b = b  current\_c = c  current\_func = next\_func  a = current\_a-alpha\*partiala(current\_a,current\_b,current\_c) # update of a  b = current\_b-alpha\*partialb(current\_a,current\_b,current\_c) # update of b  c = current\_c-alpha\*partialc(current\_a,current\_b,current\_c) # update of c  next\_func = funcabc(a,b,c)  change\_func = abs(next\_func-current\_func) # stopping criterion: values of function converge  print("Iteration",n+1,": a = ",a,", b = ",b,", c = ",c,", f(a,b,c) = ",next\_func)  if change\_func<epsilon:  break    print("The local minimum occurs at: ",next\_func) |

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

|  |
| --- |
| I made a new function called partialc for the partial derivative . I also changed it from funcab(a,b) to funcabc(a,b,c) so it now takes c as one of the params as well. I changed the epsilon 1 from 0.001 to 0.0001 and changed the max\_iters from 1000 to 500. Alpha was kept the same at 0.1. I changed the  starting values of a from 4 to 5, b from 5 to 0 and c from 6 to 15. |

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

|  |
| --- |
| Final MODEL 3 equation is: (all to 3s.f)  Minimum value of Error function is: 0.165 (3s.f.)  Number of iterations ran to reach convergence: 146 |

**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.

|  |
| --- |
| Yes, it is possible although the chances are very low. It could happen in the situation where one data point produces a small residual for model 1. This could happen when the distance from the nearest MRT station of one of the housing flats is inversely proportional to the price per square metre. Nonetheless, when doing gradient descent on this dataset, Model 1 will still produce the biggest error. |

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

|  |
| --- |
| Model 3. First of all, model 3 is the only model that also takes into account the inverse age if the flat to predict the house prices. Secondly, the minimum error value for model 3 of 0.165 is the lowest among all 3 models.Thirdly, Model 3 also takes the least number of iterations of 146 to reach convergence. Model 3 is the most accurate model for predicting housing price in this context with the dataset given. |