**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.

|  |  |
| --- | --- |
| Name | Choo Weng Yan |
| Class | DIT/FT/2A/14 |
| Admission Number | p1940208 |

**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, data is collected from 35 staff employed in a pharmaceutical company. The data can be found in the file *salary.xlsx*. The following table lists the variables used in the file and their descriptions:

|  |  |
| --- | --- |
| **Variable** | **Description** |
| *salary* | Salary in dollars per hour earned by staff |
| *years* | Number of years staff has been with company |
| *gender* | 1 = male, 0 = female |

The response variable is *salary*, and the predictors are *years* and *gender*.

Use *pandas.read\_excel* to extract the data from *salary.xlsx* into a dataframe.

**Simple Linear Regression (SLR)**

We will first build a SLR model using *years* as the predictor to predict *salary*.

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 sum 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 *salary* (*y*) using *years* (*x*) as the predictor.

Suppose any new staff who is employed by the company will earn a minimal salary of $50/hour. This means that when *x* = 0 , . Then, in the SLR model, we will only need to determine slope *b*.

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

|  |
| --- |
|  |

(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 = 0 # Starting value of x  rate = 0.001 # Set learning rate  precision = 0.0001 # Stop algorithm when absolute difference between 2 consecutive x-values is less than precision  diff = 1 # difference between 2 consecutive iterates  max\_iter = 500 # set maximum number of iterations  iter = 1 # iterations counter  E = lambda b: np.mean((data["salary"] - (50 + b\*data["years"]))\*\*2)  deriv = lambda b: np.mean(2\*data["years"]\*(-data["salary"] + (50 + b\*data["years"]))) # derivative of b  #Gradient Descent  while diff > precision and iter < max\_iter:  b\_new = b - rate \* deriv(b)  print("Iteration ", iter, ": x-value is: ", b\_new,"E(b) is: ", E(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.

|  |
| --- |
| For epsilon, I have set it as 0.001 because I think that this is significant enough to find out what is the lowest minimum point in this model.  For learning rate (alpha ∝), I have set the learning rate with different values to see which is the best fit for this model.   1. Learning rate ∝ = 0.0001 with 500 iterations, epsilon = 0.0001:   The algorithm gets trapped at Iteration 244 where *b* = 2.973,  *E(b)* = 55.176   1. Learning rate ∝ = 0.0005 with 500 iterations, epsilon = 0.0001:   The algorithm gets trapped at Iteration 58 where *b* = 2.976,  *E(b)* = 55.174   1. Learning rate ∝ = 0.001 with 500 iterations, epsilon = 0.0001:   The algorithm gets trapped at Iteration 30 where *b* = 2.976,  *E(b)* = 55.174   1. Learning rate ∝ = 0.1 with 500 iterations, epsilon = 0.0001:   The algorithm failed to converge    If the learning rate is big , the value of *E(b)* will be big as well and this will cause the initial starting points to overshoot the minimum point without knowing it. If the learning rate is too small, it will require many iterations to reach convergence and the process will be slow.   * For ∝ = 0.0001, the algorithm converges after 244 iterations. Thus, a smaller learning rate has resulted in the algorithm to converge slower. * For ∝ = 0.0005, the algorithm converges after 58 iterations. Thus, a slightly larger learning rate has resulted in the algorithm to converge a little bit faster. * For ∝ = 0.1, the algorithm did not manage to converge even after 500 iterations, the value of the function is increasing, and the value of the function goes infinity after Iteration 107 onwards. Thus, too large a learning rate has resulted in the algorithm failing to converge. * For ∝ = 0.001, It can be seen that from Iteration 29 to Iteration 30, the reduction in the value of *E(b)* is 55.174377 - 55.174372 = 0.000005, which is smaller than the epsilon.   After trying different learning rates, I have decided to choose the learning rate to be 0.001 as it has the fastest iterations (30 iterations) with the smallest value of *E(b)* . Hence, the algorithm converges and the output shows that the value of *b* = 2.976 and *E(b)* = 55.174 |

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

|  |
| --- |
| Final MODEL 1 equation is:  Minimum value of Error function is: 55.174  Number of iterations ran to reach convergence: 30 |

**MODEL 2: SLR ⇒**  (25 marks)

Now we apply the SLR model where both intercept *a* and slope *b* are to be determined, when predicting *salary* (*y*) using *years* (*x*) as the predictor.

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

|  |
| --- |
|  |

(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  next\_a = 61 # Initial starting point  next\_b = 2 # Initial starting point  alpha = 0.001 # Learning rate  epsilon = 0.0001 # Stopping criterion constant  max\_iters = 500 # Maximum number of iterations  # Partial derivatives and function  partialf\_a = lambda a,b: np.mean(2\*(-data["salary"] + a + b\*data["years"]))  partialf\_b = lambda a,b: np.mean((2\*data["years"])\*(-data["salary"] + a + b\*data["years"]))  func = lambda a,b: np.mean((data["salary"] - (a + b\*data["years"]))\*\*2)  # Initial value of function at the starting point  next\_func = func(next\_a,next\_b)  #for loop to loop through the algorithm  for n in range(max\_iters):  current\_a = next\_a #indicate the current point a  current\_b = next\_b #indicate the current point b  current\_func = next\_func #value of the function at the current point  next\_a = current\_a-alpha\*partialf\_a(current\_a,current\_b) # update of a (find the next a value)  next\_b = current\_b-alpha\*partialf\_b(current\_a,current\_b) # update of b (find the next b value)  next\_func = func(next\_a,next\_b) #find the value of the function at the next point  change\_func = abs(next\_func-current\_func) # stopping criterion: values of function converge  print("Iteration",n+1,": a = ",next\_a,", b = ",next\_b,", E(a,b) = ",next\_func)  if change\_func<epsilon:  break #break the for lop and iteration will end |

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

|  |
| --- |
| In this model 2, I have set the initial starting point for *a* (next\_a) and *b* (next\_b) as 61 and 2. This is because I used Minitab to produce a regression graph as well as a regression analysis to find out the what is the equation to produce the line of best fit between ‘*salary’* and ‘*years’* in this dataset. Thus, it shows that is the best fit. Hence, I have decided to get the whole number of 60.74 (which is the initial starting point of a) as 61 and the whole number of 2.167 (which is the initial starting point of b) as 2.  For epsilon, I have set it as 0.0001 because I think that this is significant enough to find out what is the lowest minimum point in this model.  For learning rate (alpha ∝), I have set the learning rate with different values to see which is the best fit for this model.   1. Learning rate ∝ = 0.01 with 500 iterations, epsilon = 0.0001:   The algorithm failed to converge   1. Learning rate ∝ = 0.0001 with 500 iterations, epsilon = 0.0001:   The algorithm gets trapped at Iteration 77 where *a* = 60.698, *b* = 2.174,  *E(a,b)* = 29.011   1. Learning rate ∝ = 0.001 with 500 iterations, epsilon = 0.0001:   The algorithm gets trapped at Iteration 11 where *a* = 60.698 , *b* = 2.171,  *E(a,b)* = 29.009    If the learning rate is big , the value of *E(a,b)* will be big as well and this will cause the initial starting points will have overshoot the minimum point without knowing it.   * For ∝ = 0.0001, the algorithm converges after 77 iterations. Thus, a smaller learning rate has resulted in the algorithm to converge slower. * For ∝ = 0.01, the algorithm did not manage to converge even after 500 iterations and the value of the function is increasing. Thus, too large a learning rate has resulted in the algorithm failing to converge. * For ∝ = 0.001, It can be seen that from Iteration 10 to Iteration 11, the reduction in the value of *E(a,b)* is 29.00918 - 29.00908 = 0.0001, which is not more than the epsilon.   After trying different learning rates, I have decided to choose the learning rate to be 0.001. This is because it has the fastest iteration (11 iterations) with the smallest value of *E(a,b).* Hence, the algorithm converges and the output shows that the value of *a* = 60.698, *b* = 2.171 and *E(a,b)* = 29.009  The screenshot of the findings in Minitab and the output of the code are as follows: |

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

|  |
| --- |
| Final MODEL 2 equation is:  Minimum value of Error function is: 29.009  Number of iterations ran to reach convergence: 11 |

**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 *salary* (*y*) using *years* (*x*) and *gender* (*g*) as the predictors.

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

|  |
| --- |
| Gradient descent algorithm can be extended for Model 3 by extending the equation in vector form.  The actual regression formula for this model is:where a, b and c are the parameters that we have to be determined in such a way that they minimize the error, it should be as minimum as possible. In multiple regression, the error function formula is and for this model is:  And it can be extended in vector form:    In Model 3, the gradient descent algorithm can be extended by doing another partial differentiation and now instead of 2 partial differentials, Error function would be 3 partial differentiations which depends on three variables (where the Error Function is expressed in terms of *a, b* and *c* ) and they will be derived as , and  Hence, the derivatives of the Error function in Model 3 are: |

(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  next\_a = 57.7 # Initial starting point a  next\_b = 2.2 # Initial starting point b  next\_c = 6.5 # Initial starting point c  alpha = 0.001 # Learning rate  epsilon = 0.001 # Stopping criterion constant  max\_iters = 500 # Maximum number of iterations  # Partial derivatives and function  partialf\_a = lambda a,b,c: np.mean(2\*(-data["salary"] + a + b\*data["years"] + c\*data["gender"]))  partialf\_b = lambda a,b,c: np.mean((2\*data["years"])\*(-data["salary"] + a + b\*data["years"] + c\*data["gender"]))  partialf\_c = lambda a,b,c: np.mean((2\*data["gender"])\*(-data["salary"] + a + b\*data["years"] + c\*data["gender"]))  func = lambda a,b,c: np.mean((data["salary"] - (a + b\*data["years"] + c\*data["gender"] ))\*\*2) # original full expression of the function  # Initial value of function at the starting point  next\_func = func(next\_a,next\_b, next\_c)  #for loop to loop through the algorithm  for n in range(max\_iters):  current\_a = next\_a #indicate the current point a  current\_b = next\_b #indicate the current point b  current\_c = next\_c #indicate the current point c  current\_func = next\_func #value of the function at the current point  next\_a = current\_a-alpha\*partialf\_a(current\_a,current\_b, current\_c) # update of a (find the next a value)  next\_b = current\_b-alpha\*partialf\_b(current\_a,current\_b, current\_c) # update of b (find the next b value)  next\_c = current\_c-alpha\*partialf\_c(current\_a,current\_b, current\_c) # update of c (find the next c value)  next\_func = func(next\_a,next\_b, next\_c) #find the value of the function at the next point  change\_func = abs(next\_func-current\_func) # stopping criterion: values of function converge  print("Iteration",n+1,": a = ",next\_a,", b = ",next\_b,", c= ", next\_c, ", f(x,y) = ",next\_func) #print result  if change\_func<epsilon:  break #break the for lop and iteration will end |

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

|  |
| --- |
| In this model 3, I have set the initial starting point for *a* (next\_a), *b* (next\_b) and c (next\_c) as 57.7, 2.2 and 6.5. This is because I used Minitab to produce a regression graph as well as a regression analysis to find out the what is the equation to produce the line of best fit between ‘*salary’* , ‘*years’* and *‘gender’* in this dataset. Thus, it shows that is the best fit. Hence, I have decided to get the number with one decimal place of 57.74 (which is the initial starting point of a) as 57.7, the number of 2.151 (which is the initial starting point of b) as 2.2 and the number of 6.50 (which is the initial starting point of c) as 6.5.  For epsilon, I have set it as 0.001 because I think that this is significant enough to find out what is the lowest minimum point in this model.  For learning rate (alpha ∝), I have set the learning rate with different values to see which is the best fit for this model.   1. Learning rate ∝ = 0.01 with 500 iterations, epsilon = 0.001:   The algorithm failed to converge   1. Learning rate ∝ = 0.0001 with 500 iterations, epsilon = 0.001:   The algorithm gets trapped at Iteration 51 where *a* = 57.698, *b* = 2.166,  *c* = 6.499, *E(a,b,c)* = 18.480   1. Learning rate ∝ = 0.0005 with 500 iterations, epsilon = 0.001:   The algorithm gets trapped at Iteration 16 where *a* = 57.697, *b* = 2.159,  *c* = 6.499, *E(a,b,c)* = 18.470   1. Learning rate ∝ = 0.001 with 500 iterations, epsilon = 0.001:   The algorithm gets trapped at Iteration 9 where *a* = 57.697 , *b* = 2.157,  *c* = 6.499, *E(a,b,c)* = 18.464    If the learning rate is big , the value of *E(a,b,c)* will be big as well and this will cause the initial starting points will have overshoot the minimum point without knowing it.   * For ∝ = 0.0001, the algorithm converges after 51 iterations. Thus, a smaller learning rate has resulted in the algorithm to converge slower. * For ∝ = 0.0005, the algorithm converges after 16 iterations. Thus, a slightly larger learning rate has resulted in the algorithm to converge a little faster. * For ∝ = 0.01, the algorithm did not manage to converge even after 500 iterations and the value of the function is increasing. Thus, too large a learning rate has resulted in the algorithm failing to converge. * For ∝ = 0.001, It can be seen that from Iteration 8 to Iteration 9, the reduction in the value of *E(a,b,c)* is 18.46469 - 18.46390 = 0.00079, which is less than the epsilon.   After trying different learning rates, I have decided to choose the learning rate to be 0.001. This is because it has the fastest iteration (9 iteration) with the smallest value of *E(a,b,c).* Hence, the algorithm converges and the output shows that the value of *a* = 57.697, *b* = 2.157, *c* = 6.499 and *E(a,b,c)* = 18.464  The screenshot of the findings in Minitab and the output of the code are as follows: |

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

|  |
| --- |
| Final MODEL 3 equation is:  Minimum value of Error function is: 18.464  Number of iterations ran to reach convergence: 9 |

**Conclusion** (25 marks)

(a) Using Python (or other software), in a single figure, plot the data points (scatterplot) together with the linear lines representing the three models. Insert the figure below.

Note:

* The categorical variable *gender* can be represented in a bivariate scatterplot as legend (typically in colour).
* MODEL 3 equation can be written as two separate equations, one representing male and one representing female.

|  |
| --- |
|  |

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

|  |
| --- |
| For Model 1, it is a regression model which specifies the minimal salary of a staff for at least $50 per hour in the regression equation and predict the salary of the staff using years by determining the slope (b). Based on the graph in part a, it can be seen that the data points in the graph have the furthest distance to the blue line (Model 1). Besides that, the mean squared error of model 1 is 55.174 which has the highest mean squared error value among all the 3 models. This indicates that the data does not fit well in Model 1.  For Model 2, it is a regression model which allows us to predict the salary of the staff using years without specify the minimum salary per hour but requires us to determine the interception (a) and slope (b). This model allows us to predict the general salary of the staff without knowing the gender of the staff. Based on the graph in part a, some of the data points in the graph are on the orange line (Model 2). Besides that, the mean squared error value of model 2 is 29.009 which indicates that the data fit Model 2 quite good.  For Model 3, it is a regression model which allows us to predict the salary of the staffs based on the years and the gender where interception (a), slope (b) and c have to be determined. This model allows us to predict the salary of the staff based on his or her number of years working with the company and the staff’s gender. Based on the graph in part a, it can be seen that the data points for female (0) in the graph have the nearest distance to the green line (Model 3-female) and the data points for male (1) have the nearest distance to the red line (Model 3-male). Besides that, the mean squared error value of model 3 is 18.464 which has the smallest mean squared error value among all the 3 models. This indicates that the data fit well in Model 3.  I would choose Model 3 to predict salary in this context. This is because in Model 3, I am able to predict the salary of the staff based on his or her gender as well as the number of years the staff has been with company which provides more information from the data and it enables us to predict the salary more accurately. On the other hand, the data points have the closest distance to the lines for Model 3-female and Model 3-male which indicates that the data is able to fit the best in Model 3. Also, it has the lowest mean squared error value among all the 3 models which indicates that it has made the least error.  Hence, I would choose Model 3 to predict the salary of the staff in the company. |

* **END -**