**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, 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.

|  |
| --- |
| import numpy as np  import pandas as pd  import sympy as sp  salary = pd.read\_excel("./salary.xlsx")  n = salary["salary"].count()  y = salary["salary"]  x = salary["years"]  b = sp.symbols("b")  E1 = (1/n)\*sum( ((y[i] - (50 + b \* x[i]))\*\*2) for i in range(n))  display(sp.simplify(E1))  display(sp.diff(E1))  b = 0  rate = 0.001  epsilon = 0.001  diff = 1  max\_iter = 1000  iter = 1  Eb = lambda b: 136.085714285714 \* (b\*\*2) - 810.085714285715 \* b + 1260.736  Eb\_deriv = lambda b: 272.171428571428 \* 𝑏 - 810.085714285715  while diff > epsilon and iter < max\_iter:  b\_new = b - rate \* Eb\_deriv(b)  print("Iteration ", iter, ": b-value is: ", b\_new,"E(b) is: ", Eb(b\_new) )  diff = abs(b\_new - b)  iter = iter + 1  b = b\_new    print("The minimum occurs at: ", b) |

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

|  |
| --- |
| Initial parameters:  b\_1 = 0  rate = 0.01  epsilon = 0.001  max\_iter = 1000  Final parameters:  b\_1 = 0  rate = 0.001  epsilon = 0.001  max\_iter = 1000  At first I set the learning rate as 0.01 and it was too big, so the E(b) values went to infinity and the model did not converge. Then, I changed the learning rate to 0.001 and the model converged. |

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

|  |
| --- |
| Final MODEL 1 equation is: y = 50 + 2.974x  Minimum value of Error function is: 55.17490833615716  Number of iterations ran to reach convergence: 23 iterations. |

**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  import pandas as pd  import sympy as sp  salary = pd.read\_excel("./salary.xlsx")  n = salary["salary"].count()  y = salary["salary"]  x = salary["years"]  a, b = sp.symbols("a b")  E2 = (1/n)\*sum( ((y[i] - (a + b \* x[i]))\*\*2) for i in range(n))  display(sp.simplify(E2))  display(sp.diff(E2, a))  display(sp.diff(E2, b))  a\_2 = 60 # Initial point  b\_2 = 2.9 # Initial point  alpha = 0.001 # Learning rate  epsilon = 0.001 # Stopping criterion constant  max\_iters = 1000 # Maximum number of iterations  # Partial derivatives and function  E2 = lambda a,b: 1.0 \* a\*\*2 +20.5142857142857\*a\*b - 165.931428571429 \* a + 136.085714285714 \* b\*\*2 - 1835.8 \* b + 7057.30742857143  partialf\_a = lambda a,b: 2.0 \* a + 20.5142857142857 \* b - 165.931428571429  partialf\_b = lambda a,b: 20.5142857142857 \* a + 272.171428571428\* b - 1835.8    for n in range(max\_iters):  a\_n = a\_2 - alpha \* partialf\_a(a\_2, b\_2)  b\_n = b\_2 - alpha \* partialf\_b(a\_2, b\_2)  print("Iteration", n+1, ": a = ", a\_n, ", b = ", b\_n, ", E(a,b) = ", E2(a\_n, b\_n))  diff = abs(E2(a\_n, b\_n)-E2(a\_2, b\_2))  a\_2 = a\_n  b\_2 = b\_n  if diff < epsilon:  print("The local minimum occurs at iteration {} a={}, b={}, E(a,b)={}".format(n+1, a\_2, b\_2, E2(a\_2,b\_2)))  break  if n+1 == max\_iters:  print("Did not converge after {} iterations".format(max\_iters)) |

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

|  |
| --- |
| Initial parameters:  a\_2 = 50  b\_2 = 2.9  rate = 0.001  epsilon = 0.001  max\_iter = 1000  Final parameters:  a\_2 = 60  b\_2 = 2.9  rate = 0.001  epsilon = 0.001  max\_iter = 1000  As a start, I set as 50 and as the value close to the model 1’s . Then, I slowly changed the starting value to achieve smaller mean square error. The initial value of ended up as 60 and is still the same (2.9). |

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

|  |
| --- |
| Final MODEL 2 equation is:  Minimum value of Error function is:  Number of iterations ran to reach convergence: 18 |

**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.

|  |
| --- |
| We can add a new variable to the mean square error function and use the same method with the new variable on each step. For example:   * E3 = (1/n)\*sum( ((y[i] - (a + b \* x[i] + c \* g[i]))\*\*2) for i in range(n)) * display(sp.diff(E3, c)) * c\_n = c - alpha \* partialf\_c(a, b, c) * etc. |

(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 sympy as sp  salary = pd.read\_excel("./salary.xlsx")  n = salary["salary"].count()  y = salary["salary"]  x = salary["years"]  g = salary["gender"]  a, b, c = sp.symbols("a b c")  E3 = (1/n)\*sum( ((y[i] - (a + b \* x[i] + c \* g[i]))\*\*2) for i in range(n))  display(sp.simplify(E3))  display(sp.diff(E3, a))  display(sp.diff(E3, b))  display(sp.diff(E3, c))  a = 60  b = 2.9  c = 6  alpha = 0.001  epsilon = 0.001  max\_iters = 1000  # Partial derivatives and function  func = lambda a,b,c: 1.0\*a\*\*2 + 20.5142857142857\*a\*b + 0.971428571428571\*a\*c - 165.931428571429\*a + 136.085714285714\*b\*\*2 +10.1142857142857\*b\*c - 1835.8\*b + 0.485714285714286\*c\*\*2 - 84.1657142857143\*c + 7057.30742857143  partialf\_a = lambda a,b,c: 2.0\*a + 20.5142857142857\*b +0.971428571428572\*c - 165.931428571429  partialf\_b = lambda a,b,c: 20.5142857142857\*a + 272.171428571428\*b + 10.1142857142857\*c - 1835.8  partialf\_c = lambda a,b,c: 0.971428571428572\*a + 10.1142857142857\*b + 0.971428571428572\*c - 84.1657142857143    for n in range(max\_iters):  a\_n = a - alpha \* partialf\_a(a, b, c)  b\_n = b - alpha \* partialf\_b(a, b, c)  c\_n = c - alpha \* partialf\_c(a, b, c)  print("Iteration", n+1, ": a = ", a\_n, ", b = ", b\_n, ", c = ", c\_n, ", E(a,b,c) = ", func(a\_n, b\_n, c\_n))  diff = abs(func(a\_n, b\_n, c\_n)-func(a, b, c))  a = a\_n  b = b\_n  c = c\_n  if diff < epsilon:  print("The minimum occurs at iteration {} a={}, b={}, c={}, E(a,b,c)={}".format(n+1, a, b, c, func(a,b,c)))  break  if n+1 == max\_iters:  print("Did not converge after {} iterations".format(max\_iters)) |

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

|  |
| --- |
| Initial parameters:  a\_3 = 60  b\_3 = 2.9  c\_3 = 10  rate = 0.001  epsilon = 0.001  max\_iter = 1000  Final parameters:  a\_3 = 60  b\_3 = 2.9  c\_3 = 6  rate = 0.001  epsilon = 0.001  max\_iter = 1000  I set the initial values of and the close to the second model and begin searching for by slowly looking at the change in . If the value goes lower, I will lower the initial point for to the rounded value of on the last iteration (1000). I started with 10 and kept lowering the initial value until the model converges. |

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

|  |
| --- |
| Final MODEL 3 equation is:  Minimum value of Error function is:  Number of iterations ran to reach convergence: 21 |

**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?

|  |
| --- |
| Model 1:   * Does not represent the data accurately because it assumes that the minimal salary would be $50 when it is actually closer to $60. * Only consider the year variable. * There are many data points that do not align with the predicted line. * The Mean Square Error (55.262) is larger than the other models.   Model 2:   * Predicts the minimal salary. * Only consider year variable. * There are still many data points that are far away from the regression line. * The Mean Square Error (29.148) has decreased significantly compared to model 1.   Model 3:   * Predicts the minimal salary. * Considers year and gender variables. * Produces different lines for different genders. * Most of the data points on this model are on, or relatively close to the regression lines compared to the models 1 and 2. * Has the smallest Mean Square Error (19.374) among the 3 models.   I will use multiple linear regression model to predict the salary because it has the smallest mean square error. |