**Mathematics for AI – Assignment 2**

**Name : Dario Prawara Teh Wei Rong Class : DAAA / FT / 2A / 04 Admission No : 2201858**

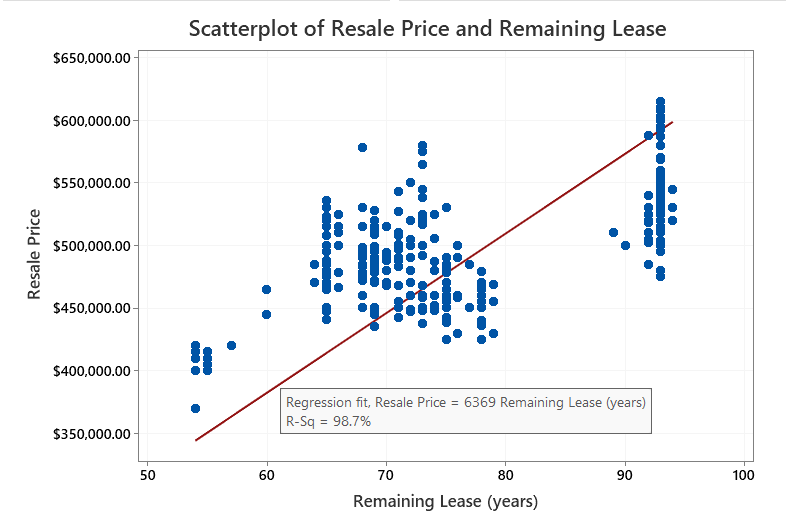
**Dataset Used : 5. ChoaChuKang Resale transactions 4\_r**

**Question 1**

**1 a)**

Plot the scatterplot of *Resale Price* and *Remaining Lease (years)* with a fitted regression line.

* Equation of the Regression Line : **= .**

****

**1 b)**

Expressing the Error function in terms of *b* only when *a = 0*.

* Value of n = 294 -***‘n’ represents the number of terms in the dataset used for regression***

Upon obtaining the Error function , the equation below represents .

**1 c) (i)**

With reference to the univariate gradient descent algorithm, use Python code to **find the value of b** where **is at its minimum**.

* Define the **y variable as ‘Resale Price’** and **x variable as ‘Remaining Lease (years)’**.
* To perform univariate gradient descent, define and calculate the loss function, partial loss function (derivative of *f*) and initial loss function with respect to the set parameters.

# *Use univariate gradient descent algorithm to find the value of b*

# *Define the predicted variable*

y = df['Resale Price']

# *Define the predictor variable*

x = df['Remaining Lease (years)']

# *Define the parameters for gradient descent*

b = 1 # *Starting value of b*

rate = 0.0001 # *Set learning rate*

epsilon = 0.0001 # *Stop algorithm when absolute difference between 2 consecutive b-values is less than epsilon*

diff = 1 # *Difference between 2 consecutive iterates*

max\_iter = 1000 # *Set maximum number of iterations*

iter\_count = 1  # *Iterations counter*

# *Functions*

n = len(x)

f = lambda b: (1/n) \* np**.**sum((y - (x \* b)) \*\* 2) # *Loss Function*

deriv = lambda b: (-2/n) \* np**.**sum(x \* (y - (x \* b))) # *Derivative of f*

# *Perform the Gradient Descent Function using a while loop*

while diff > epsilon and iter\_count < max\_iter:

    iter\_count += 1

    # *Update value of b*

    b\_new = b - rate \* deriv(b)

    # *Stopping Criterion*

    diff = abs(f(b\_new) - f(b))

    b = b\_new

# *Calculate the minimum error*

minimum\_error = diff

# *Printing the desired output*

print("Number of iterations is", iter\_count - 1)

print("The local minimum occurs when b is", round(b,4))

print("Minimum error is", minimum\_error)

* Based on the code provided above, below is the following output obtained, displaying the values for number of iterations, local minimum, and minimum error respectively.

|  |
| --- |
| Number of iterations is 13  The local minimum occurs when b is 6369.4159  Minimum error is 7.152557373046875e-06 |

The result signifies that the gradient descent algorithm took **13 iterations to converge** to a solution, where the local minimum occurs when ***b*** = 6369.4159, with a minimum error of 7.152557373046875e-06 when predicting Resale Price based on Remaining Lease (years).

This result aligned well with Minitab’s ***b*** value of 6369, indicating that the results obtained from Python is valid, and this shows consistency and accuracy between the two methods applied.

**1 c) (ii)**

The equation of the regression line obtained from univariate gradient descent can be expressed in the following form : **= .**

Given that the value of ***b*** = 6369 ( rounded to nearest whole number ), the final equation can be re-written as :

**=.**

**Question 2**

**2 a)**

Plot the regression plot of Resale Price and Remaining Lease (years) with a fitted regression line and equation of the regression line.

* Regression Equation : **= .**

A graph of a line graph

Description automatically generated with medium confidence

To estimate the value of the response variable, we will select a suitable predictor value for the variable : Remaining Lease (years).

So, the **mean value for ‘Remaining Lease (years)’** will be taken to estimate the response variable ‘Resale Price’ to find out how the response variable varies on average.

* Mean value for ‘Remaining Lease (years)’ : 75.52 76 years

A close up of a sign

Description automatically generated

Thus, based on the **predictor value of 76**, the resale price is estimated to be at **$490,000.00**.

**2 b)**

Expressing the Error function **in terms of *a* and *b* only**.

* Value of n = 294 -***‘n’ represents the number of terms in the dataset used for regression***

Now, from , **derive** .

Similarly, from , **derive** .

**2 c) (i)**

With reference to the gradient descent algorithm, use Python code to **find the value of a and b** where **is at its minimum**.

* Define the **y variable as ‘Resale Price’** and **x variable as ‘Remaining Lease (years)’**.
* To perform gradient descent, define and calculate the loss function and partial loss functions for a and b (derivative of *f*) with respect to the set parameters.

# *Use gradient descent algorithm to find the value of a and b*

# *Define the predicted variable*

y = df['Resale Price']

# *Define the predictor variable*

x = df['Remaining Lease (years)']

# *Scale the predicted variable (y)*

scaler\_y = StandardScaler()

y\_scaled = scaler\_y**.**fit\_transform(y**.**values**.**reshape(-1, 1))

# *Scale the predictor variable (x)*

scaler\_x = StandardScaler()

x\_scaled = scaler\_x**.**fit\_transform(x**.**values**.**reshape(-1, 1))

# *Define the parameters for gradient descent*

a = 2 # *Starting value of a*

b = 1 # *Starting value of b*

rate = 0.1 # *Set learning rate*

epsilon = 0.0000000001  # *Stop algorithm when absolute difference between 2 consecutive a or b-values is less than epsilon*

diff = 1  # *Difference between 2 consecutive iterates*

max\_iter = 1000  # *Set maximum number of iterations*

iter\_count = 1  # *Iterations counter*

n = len(x)

f = lambda a, b: (1/n) \* np**.**sum((y\_scaled - a - (x\_scaled \* b)) \*\* 2) # *Loss Function*

deriv\_a = lambda a, b: (-2/n) \* np**.**sum(y\_scaled - a - (x\_scaled \* b)) # *Partial Loss Function for a*

deriv\_b = lambda a, b: (-2/n) \* np**.**sum(x\_scaled \* (y\_scaled - a - (x\_scaled \* b))) # *Partial Loss Function for b*

while diff > epsilon and iter\_count < max\_iter:

    iter\_count += 1

    # *Update value of a and b*

    a\_new = a - rate \* deriv\_a(a, b)

    b\_new = b - rate \* deriv\_b(a, b)

    # *Stopping Criterion*

    diff = abs(f(a\_new, b\_new) - f(a, b))

    a, b = a\_new, b\_new

# *Calculate the minimum error*

minimum\_error = diff

# *Unscale the coefficients a and b*

unscaled\_a = (a \* scaler\_y**.**scale\_[0]) - ((b \* (scaler\_y**.**scale\_[0] \* scaler\_x**.**mean\_[0])) / scaler\_x**.**scale\_[0]) + scaler\_y**.**mean\_[0]

unscaled\_b = b \* (scaler\_y**.**scale\_[0] / scaler\_x**.**scale\_[0])

# *Printing the desired output*

print("Number of iterations is", iter\_count - 1)

print("The local minimum occurs when a is", round(unscaled\_a, 4), "and b is", round(unscaled\_b, 4))

print("Minimum error is", minimum\_error)

* Based on the code provided above, below is the following output obtained, displaying the values for number of iterations, local minimum, and minimum error respectively.

|  |
| --- |
| Number of iterations is 54  The local minimum occurs when a is 316900.779 and b is 2334.104  Minimum error is 7.9747319858825e-11 |

The result signifies that the gradient descent algorithm took **54 iterations to converge** to a solution, where the local minimum occurs when ***a*** = 316900.779 and ***b*** = 2334.104, with a minimum error of 7.9747319858825e-11 when predicting Resale Price based on Remaining Lease (years).

This result aligned very well with Minitab’s ***a*** value of 316901 and ***b*** value of 2334, indicating that the results obtained from Python is considered to be valid, and this shows consistency and accuracy between the two methods applied.

**2 c) (ii)**

The equation of the regression line obtained from gradient descent can be expressed in the following form : **= .**

Given that the value of **a** = 316901 and ***b*** = 2334 ( rounded to nearest whole number ), the final equation can be re-written as :

**=.**

**2 d)**

Starting Value for and :

For initializing the start values of , I experimented with various combinations of ‘a’ and ‘b’ values, such as 1, 50, 100, 1000 for and 1, 2, 5, 10, 20 for . Through these trials, I found that ‘a’ needed to be slightly larger than ‘b’ due to the magnitudinal differences in their values. From this, I found that the best ‘a’ value was 2 and the best ‘b’ value was 1 to provide optimal convergence with the best possible results that also could align well with the values provided by Minitab.

Learning Rate Value as :

After testing multiple learning rate values between 0.1 to 0.00001, I settled on using 0.1 as this alpha value was not too high for the loss function to oscillate around and diverge and was also not too low such that it only required 54 iterations till convergence (which is considered acceptable). Hence, the learning rate of 0.1 was used for the final model.

Epsilon Value as :

Initially, I tested the model with a range of epsilon values between 0.01 to 0.0000000001 but I found that 0.0000000001 provided the best stopping point for the algorithm when two consecutive values is less than epsilon of 0.0000000001. In this way, the model was able to converge precisely at the optimal solution for values, preventing the values from deviating excessively from the accurate values provided.

Maximum Number of Iterations as :

For maximum iterations, I decided to set it to 1000 for this model, but the model managed to converge before with only 54 iterations before reaching the maximum number of allowed iterations. This condition was used in conjunction with the condition and having both conditions ensured that convergence was achieved optimally and with precision, avoiding both premature stopping and overtraining of the model, while saving computational resources and time.

**Question 3**

**3 a)**

Based on the dataset overview below, the context is to extract and display the resale transacted prices of the 4-Room HDB flats in Choa Chu Kang from January to June 2023.

A screenshot of a computer

Description automatically generated

Suitable variable ‘***w***’ chosen

From the context of the dataset, the suitable variable ***w (predictor)*** would be : **Floor Area (sqm)**.

‘Floor Area (sqm)’ quantifies the floor area of a property in square meters. It would be suitable as a predictor in the Multiple Linear Regression (MLR) model as it is likely to have a significant influence on Resale Price. In real estate, larger properties would generally command higher market resale prices due to the expanded living and functional spaces offered within the property, assuming the remaining lease remains constant during comparison.

Hence, ‘Floor Area (sqm)’ complements the existing predictor ‘Remaining Lease (years)’ by providing a direct measure of the property’s size. By considering both the floor area and remaining lease of a property, the MLR model **gains a more comprehensive perspective on the property resale** **value**, contributing to a more accurate prediction of ‘Resale Price’.

Data Collection Process

Number of Rows Extracted for MLR : 15 Rows

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| S/No | Block | Street Name | Storey | Floor Area (sqm) | Remaining Lease (years) | Resale Price | Resale Registration Date |
| 256 | 488A | Choa Chu Kang Ave 5 | 07 to 09 | 93 | 92 | 502000.0 | Jan-23 |
| 7 | 476C | Choa Chu Kang Ave 5 | 13 to 15 | 92 | 89 | 510000.0 | Jun-23 |
| 83 | 152 | Jln Teck Whye | 10 to 12 | 100 | 73 | 522000.0 | May-23 |
| 228 | 640 | Choa Chu Kang St 64 | 01 to 03 | 100 | 73 | 538000.0 | Feb-23 |
| 126 | 635 | Choa Chu Kang Nth 6 | 04 to 06 | 113 | 73 | 580000.0 | Apr-23 |
| 202 | 807B | Choa Chu Kang Ave 1 | 16 to 18 | 92 | 93 | 560000.0 | Feb-23 |
| 278 | 708 | Choa Chu Kang St 53 | 13 to 15 | 108 | 71 | 497000.0 | Jan-23 |
| 183 | 687B | Choa Chu Kang Dr | 04 to 06 | 90 | 78 | 460000.0 | Mar-23 |
| 10 | 489C | Choa Chu Kang Ave 5 | 07 to 09 | 93 | 92 | 520000.0 | Jun-23 |
| 32 | 506 | Choa Chu Kang St 51 | 07 to 09 | 108 | 70 | 494000.0 | Jun-23 |
| 197 | 6 | Teck Whye Ave | 01 to 03 | 104 | 60 | 445000.0 | Mar-23 |
| 249 | 440 | Choa Chu Kang Ave 4 | 01 to 03 | 106 | 69 | 487000.0 | Jan-23 |
| 260 | 488C | Choa Chu Kang Ave 5 | 04 to 06 | 93 | 92 | 505000.0 | Jan-23 |
| 241 | 278 | Choa Chu Kang Ave 3 | 01 to 03 | 104 | 68 | 482888.0 | Jan-23 |
| 147 | 115 | Teck Whye Lane | 04 to 06 | 103 | 65 | 470000.0 | Apr-23 |

Data collection was conducted with the focus on preventing any instances of identical duplicates in terms of floor area, remaining lease, and resale price. This approach was adopted to guarantee a diverse and randomized dataset, thereby ensuring a balanced representation of various properties.

As for the number of rows, 15 rows was used instead of the minimum requirement of 10 to allow for a faster convergence process by providing a better estimate of true gradients and parameter values.

**3 b)**

Let intercept a = Resale Price, slope b = Remaining Lease (years), slope c = Floor Area (sqm).

Expressing the Error function **in terms of *a*, *b* and *c***.

* Value of n = 15 -***‘n’ represents the number of terms in the dataset used for regression***
* is the actual resale price of the property.
* is the remaining lease of the property.
* is the floor area of the property.
* , , and are the coefficients to be determined using the error functions.

Now, from , **derive** .

Next, from , **derive** .

Lastly, from , **derive** the last function.

How the above expressions will be used in the gradient descent algorithm

* The gradient descent algorithm will use the above derived error function and its partial derivatives to optimize the coefficients in the regression line 𝑦̂ = 𝑎 + 𝑏𝑥 + 𝑐𝑤.
* Starting with the initial coefficient estimates, the algorithm will calculate the total error using (error function) and update the coefficients by considering the gradients and the learning rate, as well as the epsilon value set. This error reflects how well the existing coefficients predict the actual ‘Resale Price’ values based on the predictors.
* The partial derivatives , and then provide the direction and magnitude of change required by the coefficients to minimize the error function. It guides the adjustments for individually until a stopping criterion is met where further adjustments are unlikely to significantly improve the model’s performance.
* Through numerous iterations, the values of would slowly converge to the optimal coefficients that minimizes the loss function, which would accurately predict resale prices based on the predictors ‘Remaining Lease (years)’ and ‘Floor Area (sqm)’.

**3 c)**

With reference to the gradient descent algorithm, use Python code to **find the value of a, b and c** where **is at its minimum**.

* Define the **y variable as ‘Resale Price’**, **x variable as ‘Remaining Lease (years)’** and **w variable as ‘Floor Area (sqm)’**.
* To perform gradient descent, define and calculate the loss function and partial loss functions for a, b, and c (derivative of *f*) with respect to the set parameters to obtain the regression line.

# *Use gradient descent algorithm to find the value of a, b and c*

# *Define the predicted variable*

y = model3\_df['Resale Price']

# *Define the predictor variables*

x = model3\_df['Remaining Lease (years)']

w = model3\_df['Floor Area (sqm)']

# *Scale the predicted variable (y)*

scaler\_y = StandardScaler()

y\_scaled = scaler\_y**.**fit\_transform(y**.**values**.**reshape(-1, 1))

# *Scale the predictor variables (x and w)*

scaler\_x = StandardScaler()

x\_scaled = scaler\_x**.**fit\_transform(x**.**values**.**reshape(-1, 1))

scaler\_w = StandardScaler()

w\_scaled = scaler\_w**.**fit\_transform(w**.**values**.**reshape(-1, 1))

# *Define the parameters for gradient descent*

a = 1 # *Starting value of a*

b = 1  # *Starting value of b*

c = 1 # *Starting value of c*

rate = 0.1 # *Set learning rate*

epsilon = 0.000000000000001  # *Stop algorithm when absolute difference between 2 consecutive values is less than epsilon*

diff, a\_diff, b\_diff, c\_diff = 1, 1, 1, 1  # *Difference between 2 consecutive iterates*

max\_iter = 1000  # *Set maximum number of iterations*

iter\_count = 1  # *Iterations counter*

# *Functions*

n = len(x)

f = lambda a, b, c: np**.**mean((y\_scaled - (a + x\_scaled \* b + w\_scaled \* c)) \*\* 2)  # *Loss Function*

deriv\_a = lambda a, b, c: (-2 / n) \* np**.**sum(y\_scaled - (a + x\_scaled \* b + w\_scaled \* c))  # *Partial Loss Function for a*

deriv\_b = lambda a, b, c: (-2 / n) \* np**.**sum(x\_scaled \* (y\_scaled - (a + x\_scaled \* b + w\_scaled \* c)))  # *Partial Loss Function for b*

deriv\_c = lambda a, b, c: (-2 / n) \* np**.**sum(w\_scaled \* (y\_scaled - (a + x\_scaled \* b + w\_scaled \* c)))  # *Partial Loss Function for c*

# *Perform the Gradient Descent Function using a while loop*

while diff > epsilon and iter\_count < max\_iter:

    iter\_count += 1

    # *Update value of a, b & c*

    a\_new = a - rate \* deriv\_a(a, b, c)

    b\_new = b - rate \* deriv\_b(a, b, c)

    c\_new = c - rate \* deriv\_c(a, b, c)

    # *Stopping Criterion*

    diff = abs(f(a\_new, b\_new, c\_new) - f(a, b, c))

    # *Update the old loss*

    a, b, c = a\_new, b\_new, c\_new

# *Calculate the minimum error*

minimum\_error = diff

# *Unscale the coefficients a, b and c*

unscaled\_a = (a \* scaler\_y**.**scale\_[0]) - ((b \* (scaler\_y**.**scale\_[0] \* scaler\_x**.**mean\_[0])) / scaler\_x**.**scale\_[0]) - ((c \* (scaler\_y**.**scale\_[0] \* scaler\_w**.**mean\_[0])) / scaler\_w**.**scale\_[0]) + scaler\_y**.**mean\_[0]

unscaled\_b = b \* (scaler\_y**.**scale\_[0] / scaler\_x**.**scale\_[0])

unscaled\_c = c \* (scaler\_y**.**scale\_[0] / scaler\_w**.**scale\_[0])

# *Printing the desired output*

print("Number of iterations is", iter\_count - 1)

print("The local minimum occurs when a is", round(unscaled\_a, 4), "and b is", round(unscaled\_b, 4), "and c is", round(unscaled\_c, 4))

print("Minimum error is", minimum\_error)

* Based on the code provided above, below is the following output obtained, displaying the values for number of iterations, local minimum, and minimum error respectively.

|  |
| --- |
| Number of iterations is 275  The local minimum occurs when a -246956.5951 is and b is 3715.5665 and c is 4652.8425  Minimum error is 9.43689570931383e-16 |

The result signifies that the gradient descent algorithm took **275 iterations to converge** to a solution, where the local minimum occurs when ***a*** = -246956.5951, ***b*** = 3715.5665 , and ***c =*** 4652.8425, with a minimum error of 9.43689570931383e-16 when predicting Resale Price based on Remaining Lease (years) and Floor Area (sqm).

Verifying the accuracy of the regression line by checking with Minitab

A screenshot of a calculator

Description automatically generated

With reference to the output provided by Minitab, we can verify that the results obtained by our regression line obtained by the Python code is accurate as the values are well-aligned with the provided output values by Minitab. Hence, the convergence was optimized in the code and the results obtained were optimal and accurate.

Final equation obtained from gradient descent

The equation of the regression line obtained from gradient descent can be expressed in the following form : **= .**

Given that the value of ***a*** = -246957, ***b*** = 3716 and ***c*** = 4653 ( rounded to nearest whole number ), the final equation can be re-written as :

**=**