**Probabilities.**

**Question 1**

***Answer:***

To solve this question, we will use the binomial distribution because we have two possible outcomes holiday accommodation is full or not-full and 25% full for each accommodation.

binomial dist. formula:

Where:

1. n = 90 (total number of holiday accommodations.)
2. x is the number of successes
3. p = 0.25 (the probability of success on a single trial)
4. q = 0.75 = 1 - p
5. =

**(a) What is the probability that no more than 60 of these are full?**

***Answer:***

Here We get a result 1, which means that there is a 100% certainty that no more than 60 of these accommodations will be full.

***Approach:***

in this question, we have no more than 60i.e.

Let,

similarly,

P(2) = 2.534391575198E-9

P(3) = 2.4780717624158E-8

P(4) = 1.7966020277515E-7

P(5) = 1.0300518292442E-6

P(6) = 4.8641336380975E-6

P(7) = 1.945653455239E-5

P(8) = 6.7287181993682E-5

P(9) = 0.00020435366383267

P(10) = 0.0005517548923482

P(11) = 0.0013375876178138

P(12) = 0.0029352617168692

P(13) = 0.0058705234337384

P(14) = 0.010762626295187

P(15) = 0.018176879965205

P(16) = 0.028401374945632

P(17) = 0.041209838156408

P(18) = 0.055709596026255

P(19) = 0.070370016033164

P(20) = 0.083271185639244

P(21) = 0.09252353959916

P(22) = 0.096729155035486

P(23) = 0.095327283223377

P(24) = 0.088707332999532

P(25) = 0.078062453039588

P(26) = 0.065052044199657

P(27) = 0.051399146034297

P(28) = 0.038549359525722

P(29) = 0.027471957363159

P(30) = 0.018619882212807

P(31) = 0.012012827234069

P(32) = 0.0073828834042718

P(33) = 0.0043253256307855

P(34) = 0.0024170937348507

P(35) = 0.001289116658587

P(36) = 0.00065649459465081

P(37) = 0.00031937574874904

P(38) = 0.00014848170775175

P(39) = 6.5991870111888E-5

P(40) = 2.8046544797552E-5

P(41) = 1.140103447055E-5

P(42) = 4.433735627436E-6

P(43) = 1.6497620939297E-6

P(44) = 5.8741529102042E-7

P(45) = 2.0015632138473E-7

P(46) = 6.5268365668935E-8

P(47) = 2.0367433258391E-8

P(48) = 6.0819418757696E-9

P(49) = 1.7376976787913E-9

P(50) = 4.7497069886962E-10

P(51) = 1.2417534610971E-10

P(52) = 3.1043836527426E-11

P(53) = 7.4192816858E-12

P(54) = 1.6945272986086E-12

P(55) = 3.6971504696916E-13

P(56) = 7.7023968118575E-14

P(57) = 1.5314707111296E-14

P(58) = 2.9045134176595E-15

P(59) = 5.2510977042432E-16

P(60) = 9.0435571573077E-17

Here We get a result 1, it means that there is a 100% certainty that no more than 60 of these accommodations will be full.

**(b) What is the probability that exactly 35 of these are full?**

***Answer:***

A probability of 0.0013 means that, based on the given information, there is a very low likelihood of exactly 35 out of the 90 holiday accommodations in Auckland being full.

***Approach:***

Here we have to find P(X = 35).

A probability of 0.0013 means that, based on the given information, there is a very low likelihood of exactly 35 out of the 90 holiday accommodations in Auckland being full.

**(c) How many of the 90 holiday accommodations are expected to be full? (i.e. mean)**

***Answer:***

On average, we would expect around 22 or 23 out of the 90 accommodations to be full.

***Approach:***

mean is denoted as E(x) and given as:

On average, we would expect around 22 or 23 out of the 90 accommodations to be full.

**Question 2**

**(a) use equations (4) and (5) to estimate β and its standard error in R.**

***Answer:***

β using equation (4) in R.

|  |  |
| --- | --- |
|  | ***[,1]*** |
|  | ***3.526667243*** |
| X1 | ***0.045764645*** |
| X2 | ***-0.001037493*** |
| X3 | ***0.188530017*** |

s.e(β) using equation (5) in R.

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***X1*** | ***X2*** | ***X3*** |
| ***0.374289884*** | ***0.001394897*** | ***0.005871010*** | ***0.00861123*** |

**Approach:**

> library(datarium) # Here we are loading ‘datarium’ library.

> data(marketing) # Loading data ‘marketing’ from datarium library.

> # Extract the variables from the dataset

> Y <- marketing$sales # sales column is store in Y

> X1 <- marketing$youtube # YouTube column is store in X1

> X2 <- marketing$newspaper # newspaper column is store in X2

> X3 <- marketing$facebook # Facebook column is store in X3

>

> # Calculate \_hat using equation (4) I.e.

> X <- cbind(1, X1, X2, X3) # The first Colum 1 is representing and Other columns are representing predictor variable.

> n <- length(Y) # n stores the length of Y

> beta\_hat <- solve(t(X) %\*% X) %\*% (t(X) %\*% Y)

# Here,

# t(X) is transpose of X

# %\*% this operator uses for matrix multiplication.

# solve(X) is calculate inverse on matrix

# Y\_hat calculates the predicted values based on the estimated coefficients.

# residuals calculate the residuals (differences between observed and predicted values).

# s\_squared calculates s^2.

# se\_beta calculates the standard error of the coefficients using matrix operations.

> # Calculate s^2 and s.e() using equations (5) I.e. where

> Y\_hat <- X %\*% beta\_hat

> residuals <- Y - Y\_hat

> s\_squared <- sum(residuals^2) / (n - 4) #

> se\_beta <- sqrt(s\_squared \* diag(solve(t(X) %\*% X))) #

>

> # Print the results

> beta\_hat # to show beta\_hat

***[,1]***

***3.526667243***

***X1 0.045764645***

***X2 -0.001037493***

***X3 0.188530017***

> se\_beta # to show se\_beta

***X1 X2 X3***

***0.374289884 0.001394897 0.005871010 0.008611234***

**(b) compare the results obtained in Question 3(a) with those obtained from the function lm() in R.**

***Answer:***

Obtain coefficient using lm() function in R.

|  |  |  |  |
| --- | --- | --- | --- |
| ***(Intercept)*** | ***youtube*** | ***newspaper*** | ***facebook*** |
| ***3.526667243*** | ***0.045764645*** | ***-0.001037493*** | ***0.188530017*** |

Obtain S.E using lm() function in R.

|  |  |  |  |
| --- | --- | --- | --- |
| ***(Intercept)*** | ***youtube*** | ***newspaper*** | ***facebook*** |
| ***0.374289884*** | ***0.001394897*** | ***0.005871010*** | ***0.008611234*** |

***Approach:***

# fitting a linear regression model to the "marketing" dataset using lm().

> lm\_model <- lm(sales ~ youtube + newspaper + facebook, data = marketing)

# "sales" is the response variable and "youtube," "newspaper," and "facebook." are the predictor variable.

> lm\_coeffs <- coef(lm\_model) # Extracting the coefficients

> lm\_se <- summary(lm\_model)$coefficients[, "Std. Error"] # Extracting the standard errors

> lm\_coeffs # To show coefficients

***(Intercept) youtube newspaper facebook***

***3.526667243 0.045764645 -0.001037493 0.188530017***

> lm\_se # To show standard errors

***(Intercept) youtube newspaper facebook***

***0.374289884 0.001394897 0.005871010 0.008611234***

**Question 3**

**(a) Write down a step-by-step procedure of Classical Gradient Descent to estimate *β* in Equation (3)**

1. Initialize β: Start with an initial value for β, denoted as β\_0. This initial value is zero.
2. Learning Rate (α): The learning rate (α) is 0.01
3. Convergence Threshold (ε): The convergence threshold (ε) specifies the level of accuracy we desire in the solution.
4. Calculate the Gradient of the Cost Function: The gradient represents the direction of the steepest increase in the cost function L with respect to β.

In this case, we calculate it as follows:

Where:

* ∇L(β) is the gradient of L wrt β.
* X' is transpose X.
* Y is the vector of observed target values.
* β is the parameter to be estimated.
* n is number of data points.

1. Update β: Update the parameter vector β in the opposite direction of the gradient to minimize the cost function L. This is done using the following update rule:

Where:

* β\_new is the updated parameter vector.
* β\_old is the current parameter vector.
* α is the learning rate.
* ∇L(β\_old) is the gradient at the current parameter values.

1. Repeat Steps 4 and 5 Until Convergence: Continuously calculate the gradient and update β using the above update rule. Keep iterating until one of the following conditions is met:

* The change in the cost function between consecutive iterations is smaller than ε .
* You reach a predefined maximum number of iterations to prevent the algorithm from running indefinitely.

1. Output the Estimated β

* Once the algorithm converges or reaches the maximum number of iterations, the final β values represent the estimated parameters of the model that minimize the cost function L.

During the optimization process, it's important to monitor the cost function's value in every iteration. The goal is to observe a consistent decrease in the cost function, indicating that the algorithm is making progress toward the optimal solution. If the cost function increases or does not decrease sufficiently, you may need to adjust the learning rate (α) or check for errors in your implementation.

**(b) Write an R code to implement the Classical Gradient Descent procedure provided in Question 2(a).**

***Answer:***

Classical Gradient Descent by implementing Question 2(a) procedure.

|  |  |
| --- | --- |
|  | ***[,1]*** |
|  | **16.82699827** |
| **X1** | **4.71490555** |
| **X2** | **3.35882816** |
| **X3** | **-0.02705912** |

***Approach:***

# Gradient Descent function

> gradient\_descent <- function(X, Y, learning\_rate, max\_iterations, tolerance) {

# gradient\_descent that takes five parameters: X, Y, learning\_rate, max\_iterations, and tolerance.

+ n <- length(Y) # length of Y store in variable n

+ beta <- rep(0, ncol(X)) # Here, it initializes a vector beta with zeros.

+ for (i in 1:max\_iterations) {

# This line starts a loop that will run for a maximum of max\_iterations times.

+ # Calculate the gradient

+ gradient = (2/n) \* (t(X) %\*% (X %\*% beta - Y))

+ #print(gradient)

+ # Check for missing or NaN values in the gradient

+ if (any(is.na(gradient)) || any(is.nan(gradient))) {

+ cat("Gradient contains missing or NaN values. Exiting...\n")

+ break

+ }

+ # Update beta

+ beta = beta - (learning\_rate \* gradient)

+

+ # Calculate the norm of the gradient

+ gradient\_norm <- sqrt(sum(gradient^2))

+

+ # Check for convergence

+ if (gradient\_norm < tolerance) {

+ cat("Gradient Descent converged after", i, "iterations.\n")

+ break

+ }

+ }

+

+ return(beta)

+ }

> marketing\_1 = as.data.frame(scale(marketing))

> Y <- marketing$sales

> X1 <- marketing\_1$youtube

> X2 <- marketing\_1$facebook

> X3 <- marketing\_1$newspaper

# Above code assigns the target variable sales to Y, and assigns the scaled feature columns to X1, X2, and X3.

> X = cbind(1, X1, X2, X3) # combining1, X1, X2, and X3

> # Set hyperparameters

> learning\_rate <- .000005

> max\_iterations <- 10000000

> tolerance <- 1e-4

> # Use Gradient Descent to estimate beta

> beta\_gradient\_descent <- gradient\_descent(X, Y, learning\_rate, max\_iterations, tolerance)

Gradient Descent converged after 1609250 iterations.

> # Print the results

> beta\_gradient\_descent

**[,1]**

**16.82699827**

**X1 4.71490555**

**X2 3.35882816**

**X3 -0.02705912**

**(c) Discuss the results obtained from Question 3(b) and compare it with that obtained from Question 2(a).**

1. Estimates of Coefficients:

Gradient Descent (Question 3(b)):

* + Intercept (β0): 16.82699827
  + Coefficient for X1: 4.71490555
  + Coefficient for X2: 3.35882816
  + Coefficient for X3: -0.02705912

Matrix Approach (Question 2(a)):

* Intercept (β0): 3.526667243
* Coefficient for X1: 0.045764645
* Coefficient for X2: -0.001037493
* Coefficient for X3: 0.188530017

Comparison and Interpretation:

* + - The coefficients obtained from the gradient descent approach is larger in size for X1 and X2 compared to those obtained from the matrix approach.
    - The gradient descent approach estimates a negative coefficient for X3, whereas the matrix approach estimates a positive coefficient.
    - The intercept () obtained from gradient descent is larger than the one we obtained from the matrix approach.

2. Convergence:

Gradient Descent (Question 3(b)):

* + - Gradient descent converges after approximately 1,609,250 iterations in your implementation.

Matrix Approach (Question 2(a)):

* + - * The matrix approach provides estimates instantly.

Comparison:

* + - * + Gradient descent is an iterative optimization technique. The number of iterations required for convergence can change based on hyperparameters and the datasets.
        + The matrix approach, being an analytical method, provides estimates in a one step.

3. Sensitivity to Hyperparameters:

Gradient Descent (Question 3(b)):

* Gradient descent requires tuning hyperparameters such as learning rate and tolerance.
* The choice of these hyperparameters can significantly affect convergence and the final estimates.

Matrix Approach (Question 2(a)):

* The matrix approach directly computes the coefficients using matrix operations.

Comparison:

* Hyperparameter tuning is a critical aspect of gradient descent. Poorly chosen hyperparameters can lead to slow convergence.
* The matrix approach is more straightforward and does not have hyperparameters to tune.

4. Interpretability:

Gradient Descent (Question 3(b)):

* The coefficients obtained from gradient descent may be more challenging to interpret due to their larger size, especially for X1 and X2.

Matrix Approach (Question 2(a)):

* The matrix approach provides coefficients that are easier to interpret.

Comparison:

* + Larger coefficient magnitudes can make it harder to interpret the impact of predictor variables on the dependent variable in the context of gradient descent.

5. Computation Time:

Gradient Descent (Question 3(b)):

* + Gradient descent involves iterative updates and may take a longer time to converge, depending on the dataset and hyperparameters.

Matrix Approach (Question 2(a)):

* + The matrix approach is generally faster and provides instant results.

Comparison:

* + Gradient descent can be computationally intensive, especially for large datasets or when convergence is slow.

In summary, the choice between the classical gradient descent approach and the matrix approach depends on the specific characteristics of the problem. Gradient descent can handle complex scenarios but requires careful tuning and may produce different results compared to the straightforward matrix approach. The matrix approach is more straightforward, computationally efficient, and provides coefficients that are easier to interpret. The choice should align with the goals and constraints of our analysis.