Statistics Software Lab Report - 10

Name of the Student: Shatansh Patnaik
Roll No: 20MA20067

IIT Kharagpur
Statistics Software Lab

Linear Regression

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the variables, aiming to find the best-fitting line that predicts the dependent variable based on the independent variables.

Algorithm to Compute Coefficients

The following algorithm outlines the steps to compute the coefficients of a linear regression model using Ordinary Least Squares (OLS) method:

Algorithm 1 Compute Coefficients of Linear Regression Model

1: **procedure** OLS($(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, n)
2: Calculate the means: $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$
3: Calculate the slope (coefficient): $\beta_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$
4: Calculate the intercept (coefficient): $\beta_0 = \bar{y} - \beta_1 \bar{x}$
5: **return** β_0, β_1
6: **end procedure**

Analysis of the Algorithm

The algorithm employs the Ordinary Least Squares (OLS) method to estimate the coefficients of the linear regression model. It starts by calculating the means of the independent and dependent variables. Then, it computes the slope (coefficient) of the regression line by finding the ratio of the covariance of the variables to the variance of the independent variable. Finally, it calculates the intercept (coefficient) using the means and the slope. The resulting coefficients represent the parameters of the linear regression model.

Computation of Confidence Intervals of the Parameters of the Model

Confidence intervals for the parameters of a linear regression model provide a range of values within which we can be reasonably confident that the true value of the parameter lies. These intervals are essential for assessing the uncertainty associated with the estimated coefficients of the model. The computation involves estimating the standard error of the coefficients and then using critical values from the t-distribution to determine the bounds of the interval.

Algorithm for the Computation of Confidence Intervals of the Parameters of the Model

The following algorithm outlines the steps for computing confidence intervals for the parameters of a linear regression model:

Algorithm 2 Compute Confidence Intervals

procedure ConfidenceIntervals($\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_n$, $SE(\hat{\beta}_0), SE(\hat{\beta}_1), ..., SE(\hat{\beta}_n)$, *df*)
    Set desired confidence level (e.g., 95%)
    Find critical value t^* from t-distribution with *df* degrees of freedom
    for $i \leftarrow 0$ **to** n **do**
        Calculate margin of error $ME_i = t^* \times SE(\hat{\beta}_i)$
        Compute confidence interval $[\hat{\beta}_i - ME_i, \hat{\beta}_i + ME_i]$
    end for
    return Confidence intervals for each parameter
end procedure

This algorithm provides a systematic approach to calculate confidence intervals for the parameters of a linear regression model, allowing researchers to make inferences about the true values of the coefficients with a specified level of confidence.

Significance Testing

Significance testing for the parameters of a linear regression model assesses whether the estimated coefficients are significantly different from zero. This helps determine the importance of each predictor variable in explaining the variation in the dependent variable. The computation involves comparing the t-statistic for each coefficient to a critical value from the t-distribution at a specified significance level (e.g., $\alpha = 0.05$).

Algorithm for performing significance testing

The following algorithm outlines the steps for conducting significance testing for the parameters of a linear regression model:

Algorithm 3 Significance Testing

procedure SignificanceTesting($\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_n$, $SE(\hat{\beta}_0), SE(\hat{\beta}_1), ..., SE(\hat{\beta}_n)$, *df*, α)
    for $i \leftarrow 0$ **to** n **do**
        Calculate the t-statistic: $t_i = \frac{\hat{\beta}_i}{SE(\hat{\beta}_i)}$
        Find critical value t^* from t-distribution with *df* degrees of freedom at significance level α
        if $|t_i| > t^*$ **then**
            $\hat{\beta}_i$ is statistically significant
        else
            $\hat{\beta}_i$ is not statistically significant
        end if
    end for
end procedure

This algorithm provides a systematic approach to conduct significance testing for the parameters of a linear regression model. It helps identify which predictors have a significant impact on the

dependent variable, allowing researchers to make informed decisions about the model's explanatory power.

Conclusion

The lack of fitness test is an important diagnostic tool for evaluating the adequacy of a simple linear regression model. By comparing the variation explained by the model to the residual variation, researchers can determine whether the linear model adequately captures the relationship between the variables. If the lack of fit is significant, it indicates that the model does not adequately explain the data, and alternative modeling approaches may be necessary.

Implementation of the above algorithms using R

The following code demonstrates the implementation of computation of parameters of the model, computation of the respective confidence and prediction interval, significance testing and confidence interval testing.

Code for Exercise-2

```r
# Exercise - 2
# a) Finding the coefficients of the model
X1 <- c(152, 183, 171, 165, 158, 161, 149, 158, 170, 153, 164, 190, 185)
X2 <- c(50, 20, 20, 30, 30, 50, 60, 50, 40, 55, 40, 40, 20)
Y <- c(120, 141, 124, 126, 117, 129, 123, 125, 132, 123, 132, 155, 147)

X <- cbind(rep(1, length(X1)), X1, X2)
Y <- matrix(Y)
coefficients <- solve(t(X) %*% X) %*% t(X) %*% Y
for(i in 1:length(coefficients)) {
  print(paste("The coefficient", i, "is", coefficients[i]))
}

# b) ANOVA Testing
residuals <- Y - X %*% coefficients
n <- length(Y)
p <- ncol(X) - 1
df_residual <- n - p - 1
SE <- sqrt(diag(solve(t(X) %*% X) * sum(residuals^2) / df_residual))
ts <- coefficients / SE
pValues <- 2 * pt(abs(ts), df = df_residual, lower.tail = FALSE)

for(i in 1:length(coefficients)) {
  if(pValues[i] < 0.05) {
    print(paste("The coefficient", i, "is significant"))
  } else {
    print(paste("The coefficient", i, "is not significant"))
  }
}

```

```r
# c) Finding 95% confidence intervals for the coefficients
residuals <- Y - X %*% coefficients

n <- length(Y)
p <- ncol(X) - 1
df_residual <- n - p - 1
SE <- sqrt(diag(solve(t(X) %*% X) * sum(residuals^2) / df_residual))
t <- qt(0.975, df = df_residual)
confidence_intervals <- cbind(coefficients - t * SE, coefficients + t * SE)
result <- data.frame(coefficients, confidence_intervals)
colnames(result) <- c("Coefficient", "Lower Bound (95% CI)", "Upper Bound
    (95% CI)")
print(result)

# d) Calculating the R-squared
yPredicted <- X %*% coefficients
mean_Y <- mean(Y)
TSS <- sum((Y - mean_Y)^2)
RSS <- sum((Y - yPredicted)^2)
R_squared <- 1 - (RSS / TSS)
cat("The R-squared value is", R_squared, "\n")
```

Multicollinearity

Multicollinearity occurs when two or more predictor variables in a regression model are highly correlated with each other. This can cause issues in the estimation of regression coefficients, leading to inflated standard errors and difficulty in interpreting the results. High multicollinearity may also result in unstable coefficient estimates, making it challenging to identify the true relationships between predictors and the response variable.

Resolving Multicollinearity

To resolve multicollinearity in a regression model, the following steps can be taken:

Algorithm 4 Steps to Resolve Multicollinearity

1. Identify highly correlated predictor variables using correlation coefficients or variance inflation factors (VIF).
2. Consider dropping one of the highly correlated variables if they represent similar information or if one is theoretically less important.
3. Collect more data if possible to increase the sample size and improve the stability of coefficient estimates.
4. Use regularization techniques such as ridge regression or LASSO regression, which penalize large coefficients and can mitigate the effects of multicollinearity.
5. Transform variables if appropriate, such as using principal component analysis (PCA) to create orthogonal predictors or creating interaction terms between correlated variables.

These steps can help alleviate multicollinearity issues and improve the robustness and interpretability of regression models.

Implementation of the above algorithm using R

The following code demonstrates the implementation of computation of parameters of the model, computation of the respective confidence and prediction interval, significance testing and confidence interval testing.

Code for Exercise-3

```r
# Exercise - 3
Conversion <- c(49.0, 50.2, 50.5, 48.5, 47.5, 44.5, 28.0, 31.5, 34.5, 35.0, 38.0, 38.5, 15.0, 17.0, 20.5, 29.5)
Reactor_Temperature <- c(1300, 1300, 1300, 1300, 1300, 1300, 1200, 1200, 1200, 1200, 1200, 1200, 1100, 1100, 1100, 1100)
Contact_Time <- c(0.0120, 0.0120, 0.0115, 0.0130, 0.0135, 0.0120, 0.0400, 0.0380, 0.0320, 0.0260, 0.0340, 0.0410, 0.0840, 0.0980, 0.0920, 0.0860)
Ratio_H2_to_n_Heptane <- c(7.5, 9.0, 11.0, 13.5, 17.0, 23.0, 5.3, 7.5, 11.0, 13.5, 17.0, 23.0, 5.3, 7.5, 11.0, 17.0)

Reactor_Temperature_sq <- Reactor_Temperature^2
Ratio_H2_to_n_Heptane_sq <- Ratio_H2_to_n_Heptane^2
Contact_Time_sq <- Contact_Time^2

X <- cbind(1, Reactor_Temperature, Reactor_Temperature_sq, Ratio_H2_to_n_Heptane, Ratio_H2_to_n_Heptane_sq, Contact_Time, Contact_Time_sq)

model <- lm(Conversion ~ ., data = as.data.frame(X))
summary(model)

data <- data.frame(
  Reactor_Temperature = Reactor_Temperature,
  Ratio_H2_to_n_Heptane = Ratio_H2_to_n_Heptane,
  Contact_Time = Contact_Time
)

correlation_matrix <- cor(data)
print(correlation_matrix)

library(corrplot)
corrplot(correlation_matrix, method = "circle")

newData <- data.frame(
  Conversion = Conversion,
  Reactor_Temperature = Reactor_Temperature,
  Contact_Time = Contact_Time
)

model <- lm(Conversion ~ Reactor_Temperature + Contact_Time, data = newData)
summary(model)
```

Polynomial Regression

Polynomial regression is a type of regression analysis in which the relationship between the independent variable x and the dependent variable y is modeled as an nth degree polynomial. It is used when the relationship between the variables is nonlinear, and a straight line cannot adequately capture the relationship.

Algorithm for the Computation of Polynomial Regression Coefficients

Algorithm 5 Polynomial Regression Algorithm

Input: Data points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ and degree of polynomial d
Output: Coefficients $a_0, a_1, ..., a_d$ of the polynomial regression model $y = a_0 + a_1x + a_2x^2 + ... + a_dx^d$
Initialize matrix X with dimensions $n \times (d + 1)$
for $i \leftarrow 1$ to n **do**
    for $j \leftarrow 0$ to d **do**
        $X[i, j] \leftarrow x_i^j$
    end for
end for
Compute the transpose of X: X^T
Compute the product of X^T and X: X^TX
Compute the inverse of X^TX: $(X^TX)^{-1}$
Compute the product of $(X^TX)^{-1}$ and X^T: $(X^TX)^{-1}X^T$
Compute the product of $(X^TX)^{-1}X^T$ and y: $(X^TX)^{-1}X^Ty$
return Coefficients $a_0, a_1, ..., a_d$

Analysis of the Algorithm

The algorithm for computing the coefficients of polynomial regression follows the least squares method. It involves constructing a design matrix X where each row corresponds to a data point and each column corresponds to a power of the independent variable x. Then, the coefficients are computed by performing matrix operations involving the transpose and inverse of the design matrix.

Algorithm for the Computing Confidence Intervals for Polynomial Regression Parameters

Algorithm 6 Confidence Intervals for Polynomial Regression Parameters

Input: Data points $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, polynomial regression coefficients $a_0, a_1, ..., a_d$, significance level α
Output: Confidence intervals for the polynomial regression coefficients $a_0, a_1, ..., a_d$
Compute the residuals: $e_i = y_i - (a_0 + a_1 x_i + a_2 x_i^2 + ... + a_d x_i^d)$
Compute the mean squared error: $MSE = \frac{1}{n-d-1} \sum_{i=1}^{n} e_i^2$
Compute the standard error of the coefficients: $SE(a_j) = \sqrt{\frac{MSE}{\sum_{i=1}^{n} (x_i - \bar{x})^{2(j+1)}}}$, where $\bar{x}$ is the mean of the x values
Compute the t-statistic for the given significance level: $t_{\alpha/2, n-d-1}$
Compute the confidence interval for each coefficient: $(a_j - t_{\alpha/2,n-d-1} \times SE(a_j), a_j + t_{\alpha/2,n-d-1} \times SE(a_j))$
return Confidence intervals for the polynomial regression coefficients $a_0, a_1, ..., a_d$

Analysis of the Algorithm

The algorithm for computing confidence intervals for parameters of a polynomial regression model is based on the standard error of the coefficients and the t-statistic. It involves computing the residuals, mean squared error, standard error of the coefficients, and the t-statistic for the given significance level. The confidence intervals are then calculated based on the standard error and t-statistic.

Algorithm for Significance Testing for Polynomial Regression

Algorithm 7 Significance Testing

Input: Coefficients $a_0, a_1, ..., a_d$ from the polynomial regression model, standard errors $SE(a_0), SE(a_1), ..., SE(a_d)$, degrees of freedom df
Output: Results of significance testing for the polynomial regression coefficients
for $j \leftarrow 0$ to d **do**
    Compute the t-statistic: $t_j = \frac{a_j}{SE(a_j)}$
    Compute the critical value from the t-distribution: $t_{\alpha/2,df}$
    if $|t_j| > t_{\alpha/2,df}$ **then**
        return Reject the null hypothesis: a_j is significant
    else
        return Accept the null hypothesis: a_j is not significant
    end if
end for

Analysis of Significance Testing

The significance testing for polynomial regression coefficients is used to determine whether each coefficient in the polynomial regression model is statistically significant. It involves computing the t-statistic for each coefficient by dividing the coefficient by its standard error. The test then compares each t-statistic to a critical value from the t-distribution at a given significance level. If the absolute value of the t-statistic exceeds the critical value, the null hypothesis is rejected, indicating that the coefficient is significant. Otherwise, if the absolute value of the t-statistic does not exceed the critical value, the null hypothesis is accepted, indicating that the coefficient is not significant.

Implementation of the above algorithms using R

The following code demonstrates the implementation of computation of parameters of the model, computation of the respective confidence and prediction interval, significance testing and confidence interval testing.

Code for Exercise-1

```r
# Exercise - 1
# a) Finding the coefficients of the model
y <- c(2.6, 2.4, 17.32, 15.6, 16.12, 5.36, 6.19, 10.17, 2.62, 2.98, 6.92, 7.06)
x1 <- c(31, 31, 31.5, 31.5, 31.5, 30.5, 31.5, 30.5, 31, 30.5, 31, 30.5)
x2 <- c(21, 21, 24, 24, 24, 22, 22, 23, 21.5, 21.5, 22.5, 22.5)
X <- cbind(rep(1, length(x1)), x1, x2, x1^2, x2^2, x1*x2)

coefficients <- solve(t(X) %*% X) %*% t(X) %*% y
for (i in 1:length(coefficients)) {
  print(paste("The coefficient", i, "is", coefficients[i]))
}

# b) Finding 95% confidence intervals for the coefficients
residuals <- y - X %*% coefficients
RSS <- sum(residuals^2)
df <- length(y) - ncol(X)
sigma <- sqrt(RSS / df)

se <- sqrt(diag(sigma^2 * solve(t(X) %*% X)))

t <- qt(0.975, df)
confidence_intervals <- cbind(coefficients - t * se, coefficients + t * se)
for (i in 1:length(coefficients)) {
  print(paste("The 95% confidence interval for the coefficient", i, "is",
      confidence_intervals[i, 1], "to", confidence_intervals[i, 2]))
}

# c) Testing the hypothesis that the coefficients are equal to zero
ts <- coefficients / se
pValues <- 2 * pt(abs(ts), df = df, lower.tail = FALSE)

```

```r
for(i in 1:length(coefficients)) {
  if(pValues[i] < 0.05) {
    print(paste("The coefficient", i, "is significant"))
  } else {
    print(paste("The coefficient", i, "is not significant"))
  }
}

# d) Calculating the R-squared
yPredicted <- X %*% coefficients
mean_y <- mean(y)
TSS <- sum((y - mean_y)^2)
RSS <- sum((y - yPredicted)^2)
R_squared <- 1 - (RSS / TSS)

cat("The R-squared value is", R_squared, "\n")
```

Logistic Regression

Logistic regression is a statistical model used for binary classification tasks, where the response variable is categorical with two possible outcomes. It estimates the probability that a given observation belongs to a particular class based on one or more predictor variables.

Gradient Descent for Coefficient Generation

Gradient descent is an optimization algorithm used to minimize the cost function in logistic regression. It iteratively adjusts the coefficients to find the optimal values that minimize the difference between predicted and actual outcomes.

Algorithm 8 Gradient Descent for Logistic Regression

1: Initialize coefficients $\beta_0, \beta_1, \ldots, \beta_n$ randomly or with zeros.
2: **repeat**
3: Compute the predicted probabilities using the logistic function.
4: Calculate the gradient of the cost function with respect to each coefficient.
5: Update coefficients using the gradient and the learning rate.
6: **until** Convergence criteria are met or maximum iterations are reached.

Gradient descent starts with an initial guess for the coefficients and iteratively adjusts them in the direction that minimizes the cost function. This process continues until convergence is achieved, or a maximum number of iterations is reached.

Significance Testing of Coefficients

After generating coefficients using logistic regression, it is essential to test their significance to determine whether they have a statistically significant impact on the prediction of the response variable.

Algorithm 9 Significance Testing of Coefficients

1: Calculate the standard error of each coefficient.
2: Compute the z-score for each coefficient using the standard error.
3: Determine the p-value associated with each z-score.
4: Reject the null hypothesis (coefficients are not significant) if the p-value is below a specified significance level.

Significance testing of coefficients helps assess the reliability of the estimated coefficients and their contribution to the model's predictive power.

Code for Exercise-4

```r
# Exercise - 4
temperature <- c(53, 57, 58, 63, 66, 67, 67, 67, 68, 69, 70, 70, 70, 70, 70,
    75, 75, 76, 76, 76, 76, 78, 79, 81)
failure <- c(1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
   0, 0, 0)

# Without using builtin Packages
logistic <- function(x) {
  return(1 / (1 + exp(-x)))
}

negative_log_likelihood <- function(beta, X, Y) {
  eta <- X %*% beta
  p <- logistic(eta)
  return(-sum(Y * log(p) + (1 - Y) * log(1 - p)))
}

beta <- rep(0, 2)
X <- cbind(rep(1, length(temperature)), temperature)

alpha <- 0.001
max_iter <- 1000

for (i in 1:max_iter) {
  eta <- X %*% beta
  p <- logistic(eta)
  gradient <- t(X) %*% (p - failure)
  beta <- beta - alpha * gradient

  if (max(abs(gradient)) < 1e-6) {
    break
  }
```

```
}

print(beta)


eta <- X %*% beta
p <- logistic(eta)

XtX_inv <- solve(t(X) %*% X)
se <- sqrt(diag(XtX_inv))
z_values <- beta / se
pValues <- 2 * (1 - pnorm(abs(z_values)))

for (i in 1:length(beta)) {
  if (pValues[i] < 0.05) {
    print(paste("The coefficient", i, "is significant"))
  } else {
    print(paste("The coefficient", i, "is not significant"))
  }
}
```