**In-Class Learning Activity 2(LA2) answers**

**Part I:**

**a)**

In linear regression, a loss function quantifies the difference between the actual values and the values predicted by the model. The goal is to minimize this difference during the training process to improve the model's accuracy. Here's a breakdown of the key concepts related to this loss function and related terms such as SSR (Sum of Squares due to Regression), SSL (Sum of Squares due to Lack of Fit), and RSS (Residual Sum of Squares):

SSR, also known as the explained sum of squares, measures the variation explained by the regression line, or the difference between the predicted values (*y*^​*i*​) and the mean of the actual values (ˉ*y*ˉ​). It quantifies how well the regression model captures the data's variance. SSR is given by:

SSL is not as commonly referenced in the context of simple linear regression as in more complex models or analysis of variance (ANOVA). It measures the portion of the total variability in the response that is not explained by the model due to its inability to capture the data structure. In simple linear regression, this concept overlaps significantly with the concept of residuals and is less distinctly defined as in more complex scenarios.

RSS, also known as the sum of squared residuals or the sum of squared errors of prediction, measures the discrepancy between the actual values and the values predicted by the model. It represents the error not explained by the regression model. RSS is given by:

**b)**

The least squares estimation method is a fundamental approach used to find the best-fitting line through a set of data points in linear regression. It aims to minimize the sum of the squared differences between the observed values and the values predicted by the linear model. This method is widely used in linear regression to estimate the parameters (coefficients) of the model, ensuring that the overall error between the predicted outcomes and the actual data is as small as possible.

It is done in three steps:

Step-1: computing the residual ( ie, loss).

Step-2: Compute residual sum of squares (RSS).

Step-3: use calculus to find the values of β0  β1 that give the minimum RSS/SSR/SSL.

Consider a simple linear regression model where the relationship between a dependent variable

y and an independent variable x is described as:

yi = β0  + β1 xi

Where β0  β1 formulas are shown below:

A mathematical equation with numbers and symbols

Description automatically generated A black text on a white background

Description automatically generated

**c)**

Accuracy checking/quality of model fit: Residual standard error (RSE), R-squared (R2), Mean squared error (MSE):

Residual Standard Error (RSE): an estimate of the standard deviation of error term . Roughly speaking, it is the average amount that the response will deviate from the true regression line

R2: the fraction of variance in Y explained by X, it is the square of the correlation between the response and the fitted linear model. E.g., 89.7% of variance in Y is explained by X in this example.

R2 : the larger the better. In the simple linear regression setting, R2 = r2 (r: correlation coefficient)

Adjusted R2: adjusted for the number of predictors in the model: the larger the better.

R2 = 1-

MSE Definition: MSE is the average of the squares of the errors. It measures the quality of an estimator—it is always non-negative, and values closer to zero are better.

A mathematical equation with numbers and symbols

Description automatically generated

**d)**

Training set: The training set is a subset of the dataset used to train a model. It contains input data along with the corresponding target data (labels or values) that the model learns from. The learning process involves adjusting the model's parameters to minimize errors in predicting the target data from the inputs.

Testing set: The testing set is another subset of the dataset, separate from the training set, used to evaluate the performance of the model. Like the training set, it includes both input data and the corresponding target data. However, the data in the testing set is not used during the training phase.

Training error: Training error refers to the error or discrepancy between the model's predictions and the actual target values on the training dataset. It is a measure of how well the model fits the data it was trained on.

Testing error: Testing error, also known as generalization error, is the error or discrepancy between the model's predictions and the actual target values on the testing dataset. It measures the model's ability to perform on data it hasn't seen during training.

**Part II:**

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph with dots and lines

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph with dots and numbers

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

auto.mpg <- read.table('https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data', stringsAsFactors = FALSE)

> View(auto.mpg)

> head(auto.mpg,2)

V1 V2 V3 V4 V5 V6 V7 V8 V9

1 18 8 307 130.0 3504 12.0 70 1 chevrolet chevelle malibu

2 15 8 350 165.0 3693 11.5 70 1 buick skylark 320

> names(auto.mpg) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "model.year", "origin", "name")

> set.seed(490)

> Random.seed <- c("Mersenne-Twister", 1)

> training.indices <- sample(1:nrow(auto.mpg), 0.5 \* nrow(auto.mpg), replace = FALSE)

> training.data <- auto.mpg[training.indices, ]

> View(training.data)

> testing.data <- auto.mpg[-training.indices,]

> View(testing.data)

> simple.model <- lm(mpg ~ weight, data = training.data)

> print(summary(simple.model))

Call:

lm(formula = mpg ~ weight, data = training.data)

Residuals:

Min 1Q Median 3Q Max

-9.2761 -2.9686 -0.1867 2.5465 16.6375

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 46.136095 1.133323 40.71 <2e-16 \*\*\*

weight -0.007665 0.000371 -20.66 <2e-16 \*\*\*

---

Signif. codes: 0 ‘’ 0.001 ‘’ 0.01 ‘’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.338 on 197 degrees of freedom

Multiple R-squared: 0.6842, Adjusted R-squared: 0.6826

F-statistic: 426.8 on 1 and 197 DF, p-value: < 2.2e-16

> lm(formula = mpg ~ weight, data = training.data)

Call:

lm(formula = mpg ~ weight, data = training.data)

Coefficients:

(Intercept) weight

46.136095 -0.007665

> degree\_of\_freedom = 318 - 2

> mse <- mean(residuals(simple.model)^2)

> mse

[1] 18.63164

> rmse <- sqrt(mse)

> rmse

[1] 4.316438

> rss <- sum(residuals(simple.model)^2)

> rss

[1] 3707.696

> rse <- sqrt( rss / degree\_of\_freedom )

> rse

[1] 3.425378

> temp = unlist(training.data["mpg"])

> ssd = sum( (temp - mean(temp) )^2 )

> r.squared <- (ssd - rss)/ssd

> r.squared

[1] 0.6841955

> p = 1 # number of predictors

> n = 0.8\*nrow(auto.mpg)# number of of observations

> adj.r.squared <- 1 - (1 - r.squared) \* (n - 1)/(n - p - 1)

> adj.r.squared

[1] 0.6831974

> fstatistic = ( (ssd - rss)/p )/(rss/(n-p-1) )

> fstatistic

[1] 685.4858

> simple.model.predictions <- predict(simple.model, testing.data)

> test.simple.model.ssl <- sum((testing.data$mpg - simple.model.predictions)^2)

> sprintf("SSL/SSR/SSE: %f", test.simple.model.ssl)

[1] "SSL/SSR/SSE: 3775.806393"

> test.simple.model.mse <- test.simple.model.ssl / nrow(testing.data)

> sprintf("MSE: %f", test.simple.model.mse)

[1] "MSE: 18.973901"

> test.simple.model.rmse <- sqrt(test.simple.model.mse)

> sprintf("RMSE: %f", test.simple.model.rmse)

[1] "RMSE: 4.355904"

> plot(mpg ~ weight, data=testing.data)

>

> abline(simple.model)

>

> multi.var.model <- lm(mpg ~ cylinders + displacement + weight + acceleration + model.year, data = training.data)

>

> print(summary(multi.var.model))

Call:

lm(formula = mpg ~ cylinders + displacement + weight + acceleration +

model.year, data = training.data)

Residuals:

Min 1Q Median 3Q Max

-7.4977 -2.3684 -0.0917 2.0539 14.2244

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.632e+01 5.626e+00 -2.900 0.00416 \*\*

cylinders -3.666e-01 4.406e-01 -0.832 0.40637

displacement 1.436e-03 9.396e-03 0.153 0.87872

weight -6.266e-03 7.685e-04 -8.154 4.39e-14 \*\*\*

acceleration 2.768e-02 1.103e-01 0.251 0.80214

model.year 7.845e-01 6.892e-02 11.384 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘’ 0.001 ‘’ 0.01 ‘’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.286 on 193 degrees of freedom

Multiple R-squared: 0.8225, Adjusted R-squared: 0.8179

F-statistic: 178.8 on 5 and 193 DF, p-value: < 2.2e-16

>

> degree\_of\_freedom = 318 - 6

>

> mse <- mean(residuals(multi.var.model)^2)

> mse

[1] 10.47481

> rmse <- sqrt(mse)

> rmse

[1] 3.23648

> ss <- sum(residuals(multi.var.model)^2)

> rss

[1] 3707.696

> rse <- sqrt( rss / degree\_of\_freedom )

> rse

[1] 3.447266

> temp = unlist(training.data["mpg"])

> ssd = sum( (temp - mean(temp) )^2 )

> r.squared <- (ssd - rss)/ssd

> r.squared

[1] 0.6841955

> p = 5

> n = 0.5\*nrow(auto.mpg)

> adj.r.squared <- 1 - (1 - r.squared) \* (n - 1)/(n - p - 1)

> adj.r.squared

[1] 0.6760141

> fstatistic = ( (ssd - rss)/p )/(rss/(n-p-1) )

> fstatistic

[1] 83.62754

> multi.var.predictions <- predict(multi.var.model, testing.data)

>

> test.multi.var.ssl <- sum((testing.data$mpg - multi.var.predictions)^2)

> sprintf("SSL/SSR/SSE: %f", test.multi.var.ssl)

[1] "SSL/SSR/SSE: 2601.049585"

> test.multi.var.mse <- test.multi.var.ssl / nrow(testing.data)

> sprintf("MSE: %f", test.multi.var.mse)

[1] "MSE: 13.070601"

> test.multi.var.rmse <- sqrt(test.multi.var.mse)

> sprintf("RMSE: %f", test.multi.var.rmse)

[1] "RMSE: 3.615329"

> scatter.smooth(x= testing.data$weight, y= testing.data$mpg, main="mpg ~ weight")

> scatter.smooth(x= testing.data$model.year, y= testing.data$mpg, main="mpg ~ model.year")

> full.model <- lm(mpg ~ cylinders + displacement + weight + acceleration + model.year, data = auto.mpg)

> print(summary(full.model))

Call:

lm(formula = mpg ~ cylinders + displacement + weight + acceleration +

model.year, data = auto.mpg)

Residuals:

Min 1Q Median 3Q Max

-8.6747 -2.3625 -0.1178 2.0375 14.3300

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.457e+01 4.138e+00 -3.521 0.00048 \*\*\*

cylinders -2.586e-01 3.286e-01 -0.787 0.43177

displacement 7.268e-03 7.146e-03 1.017 0.30977

weight -6.926e-03 5.963e-04 -11.614 < 2e-16 \*\*\*

acceleration 8.035e-02 7.839e-02 1.025 0.30604

model.year 7.553e-01 5.078e-02 14.875 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘’ 0.001 ‘’ 0.01 ‘’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.44 on 392 degrees of freedom

Multiple R-squared: 0.8087, Adjusted R-squared: 0.8062

F-statistic: 331.4 on 5 and 392 DF, p-value: < 2.2e-16

> mse <- mean(residuals(signif.model)^2)

Error: object 'signif.model' not found

> mse

[1] 10.47481

> rmse <- sqrt(mse)

> rmse

[1] 3.23648

> rss <- sum(residuals(signif.model)^2)

Error: object 'signif.model' not found

> rss

[1] 3707.696

> signif.model <- lm(mpg ~ weight + model.year, data = training.data)

> print(summary(signif.model))

Call:

lm(formula = mpg ~ weight + model.year, data = training.data)

Residuals:

Min 1Q Median 3Q Max

-8.2527 -2.1734 -0.1156 2.0752 14.1721

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -17.122226 5.231852 -3.273 0.00126 \*\*

weight -0.006783 0.000289 -23.474 < 2e-16 \*\*\*

model.year 0.798293 0.065136 12.256 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘’ 0.001 ‘’ 0.01 ‘’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.273 on 196 degrees of freedom

Multiple R-squared: 0.8212, Adjusted R-squared: 0.8194

F-statistic: 450.1 on 2 and 196 DF, p-value: < 2.2e-16

> mse <- mean(residuals(signif.model)^2)

> mse

[1] 10.54815

> rmse <- sqrt(mse)

> rmse

[1] 3.247792

> rss <- sum(residuals(signif.model)^2)

> rss

[1] 2099.082

> signif.predictions <- predict(signif.model, testing.data)

> test.signif.ssl <- sum((testing.data$mpg - signif.predictions)^2)

> sprintf("SSL/SSR/SSE: %f", test.signif.ssl)

[1] "SSL/SSR/SSE: 2582.768959"

> test.signif.mse <- test.signif.ssl / nrow(testing.data)

> sprintf("MSE: %f", test.signif.mse)

[1] "MSE: 12.978738"

> test.signif.rmse <- sqrt(test.signif.mse)

> sprintf("RMSE: %f", test.signif.rmse)

[1] "RMSE: 3.602602"

> full.signif.model <- lm(mpg ~ weight + model.year, data = auto.mpg)

>

> print(summary(full.signif.model))

Call:

lm(formula = mpg ~ weight + model.year, data = auto.mpg)

Residuals:

Min 1Q Median 3Q Max

-8.8777 -2.3140 -0.1211 2.0591 14.3330

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) -1.420e+01 3.968e+00 -3.578 0.000389 \*\*\*

weight -6.664e-03 2.139e-04 -31.161 < 2e-16 \*\*\*

model.year 7.566e-01 4.898e-02 15.447 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘’ 0.001 ‘’ 0.01 ‘’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.435 on 395 degrees of freedom

Multiple R-squared: 0.8079, Adjusted R-squared: 0.8069

F-statistic: 830.4 on 2 and 395 DF, p-value: < 2.2e-16

> mse <- mean(residuals(full.signif.model)^2)

> mse

[1] 11.70814

> rmse <- sqrt(mse)

> rmse

[1] 3.421715

> rss <- sum(residuals(full.signif.model)^2)

> rss

[1] 4659.838