Download two datasets: JDT.csv and PDE.csv. They are bug datasets of two projects JDT and PDE. Two datasets having the same column names:

**file**: name of the class in this project. This column should be used as row names in R.

**bug**: number of post-release bugs found in that class. If bug = 0 then the class is considered defect free. Otherwise, it is defect prone.

**bf**: number of previous bug fixes on this class.

**loc**: number of lines of code of this class.

**hcm**: a measurement of the complexity of the changes applied to this class in its history.

To answer the below questions, we need to set the directory where the “JDT.csv”, and “PDE.csv” files are located.

In my local machine, the code to set the working directory is

setwd(“C:/Users/cscha/OneDrive/Desktop/Software analytics”)

Now read the “JDT.csv” file by typing the following code

> data1 <- read.csv(“JDT.csv”)

Here, “data1” works as a variable to perform operations on the “JDT.csv” file

Now read the “PDE.csv” file by typing the following code

> data2 <- read.csv(“PDE.csv”)

Here, “data2” works as a variable to perform operations on the “PDE.csv” file

Q1. For each dataset, select randomly 50% classes for training and 50% for testing and repeat the process 30 times. Set the random seed as 3 last digits of your R#.

Build linear regression models to predict **bug**:

M1: using only **loc**

M2: using only **bf**

M3: using both **loc** and **bf**

Compare *R2* (goodness of fit) of those 3 models. Note: because we do cross-validation, *R2* is calculated for each training dataset 30 times. You will compare them using paired t.test. For linear model m1, call summary(m1)$r.squared to get its *R2*.

We measure the prediction effectiveness of a model by calculating the total number of actual bugs discovered in top-20% classes ranked based on the predicted number of bugs by that model on testing data. Compare 3 models based on that prediction effectiveness.

The ‘R’ code for data set JDT.csv to select randomly 50% classes for training and 50% for testing and to repeat the process 30 times and

Build linear regression models to predict **bug**:

M1: using only **loc**

M2: using only **bf**

M3: using both **loc** and **bf**

And also for comparing Compare *R2* (goodness of fit) of those 3 models using paired t-test is as follows:

> library(readr)

> set.seed(450)

> no\_of\_repetitions <- 30

> data1\_M1\_r\_squared <- numeric(no\_of\_repetitions)

> data1\_M2\_r\_squared <- numeric(no\_of\_repetitions)

> data1\_M3\_r\_squared <- numeric(no\_of\_repetitions)

> for ( i in 1:no\_of\_repetitions){

+ training\_indices <- sample(1:nrow(data1), nrow(data1) / 2)

+ training\_data <- data1[training\_indices, ]

+ testing\_data <- data1[-training\_indices, ]

+ data1\_M1\_model <- lm (bug ~ loc , data=training\_data)

+ data1\_M1\_r\_squared[i] <- summary(data1\_M1\_model)$r.squared

+ data1\_M2\_model <- lm (bug ~ bf , data=training\_data)

+ data1\_M2\_r\_squared[i] <- summary(data1\_M2\_model)$r.squared

+ data1\_M3\_model <- lm (bug ~ loc + bf , data=training\_data)

+ data1\_M3\_r\_squared[i] <- summary(data1\_M3\_model)$r.squared

+ }

> t\_test\_data1\_M1\_model\_M2\_model <- t.test(data1\_M1\_r\_squared, data1\_M2\_r\_squared, paired=TRUE)

> t\_test\_data1\_M1\_model\_M3\_model <- t.test(data1\_M1\_r\_squared, data1\_M3\_r\_squared, paired=TRUE)

> t\_test\_data1\_M2\_model\_M3\_model <- t.test(data1\_M2\_r\_squared, data1\_M3\_r\_squared, paired=TRUE)

> print(t\_test\_data1\_M1\_model\_M2\_model)

> print(t\_test\_data1\_M1\_model\_M3\_model)

> print(t\_test\_data1\_M2\_model\_M3\_model)

In the above line of code

At first, we considered the seed, which was set to 450(based on my last 3 digits of R#11800450) and we also considered the readr library for better operations on data present in .csv files.

Later we defined the number of repetitions to 30 which was considered according to the given question.

Now we initialize the r-squared values vector for three respective models, and we named them by following the code.

data1\_M1\_r\_squared <- numeric(no\_of\_repetitions)

data1\_M2\_r\_squared <- numeric(no\_of\_repetitions)

data1\_M3\_r\_squared <- numeric(no\_of\_repetitions), each vector size is equivalent to no\_of\_repetitions.

Now we go through a for loop to generate the linear regression model based on the given conditions.

To perform cross-validation, a for loop iterates 30 times Data is randomly divided into training and testing sets for each iteration with the same 50% classes in each by performing following code

training\_indices <- sample(1:nrow(data1), nrow(data1) / 2)

training\_data <- data1[training\_indices, ]

testing\_data <- data1[-training\_indices, ].

Later based on the given condition we generate the three models, and the training data are fitted to the data1\_M1\_model, data1\_M2\_model, and data1\_M3\_model linear regression models, and the resulting R-squared values are stored in the corresponding vectors. Now we perform the paired t-test between all three models to compare the goodness of fit

The output of (t\_test\_data1\_M1\_model\_M2\_model) is:

Paired t-test

data: data1\_M1\_r\_squared and data1\_M2\_r\_squared

t = -11.433, df = 29, p-value = 2.908e-12

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

-0.12794343 -0.08911504

sample estimates:

mean difference

-0.1085292

In conclusion, the above results say that the means of data1\_M1\_r\_squared and data1\_M2\_r\_squared significantly differ from one another. Data1\_M1\_r\_squared is generally less than data1\_M2\_r\_squared, by considering the negative mean difference of about -0.1085 (negative). It is clear from the extremely low p-value that this difference is not the result of chance, providing strong evidence against the null hypothesis.

> print(summary(data1\_M1\_r\_squared))

The above code gives the summary of Model 1 which is M1\_r\_squared vector of data1. The output is:

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.1846 0.2867 0.3446 0.3337 0.3660 0.4556

> print(summary(data1\_M1\_model)$r.squared)

This code gives us the r-squared value for the model M1 of data1.

0.2769639

The output of (t\_test\_data1\_M1\_model\_M3\_model) is:

Paired t-test

data: data1\_M1\_r\_squared and data1\_M3\_r\_squared

t = -19.307, df = 29, p-value < 2.2e-16

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

-0.1470974 -0.1189180

sample estimates:

mean difference

-0.1330077

It can be concluded from the above result that there is a very significant difference between the means of data1\_M1\_r\_squared and data1\_M3\_r\_squared, with a negative mean difference, indicating that data1\_M1\_r\_squared is, on average, lower than data1\_M3\_r\_squared. From the above results, it is also clear that there is a low p value which is almost equal to zero.

> print(summary(data1\_M3\_r\_squared))

The above code gives the summary of Model 3 which is M3\_r\_squared vector of data1. The output is:

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.3535 0.4258 0.4596 0.4667 0.4914 0.6266

> print(summary(data1\_M3\_model)$r.squared)

This code gives us the r-squared value for the model M3 of data1.

[1] 0.3716608

The output of (t\_test\_data1\_M2\_model\_M3\_model) is:

Paired t-test

data: data1\_M2\_r\_squared and data1\_M3\_r\_squared

t = -7.4995, df = 29, p-value = 2.888e-08

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

-0.03115410 -0.01780286

sample estimates:

mean difference

-0.02447848

From the above output, we can note that the t-value is -7.4 approximately and the p-value is extremely low which is almost close to zero. From the values above it can be concluded that there is a difference between the means of data1\_M2\_r\_squared and data1\_M3\_r\_squared, with a negative mean difference, indicating that data1\_M2\_r\_squared is, on average, lower than data1\_M3\_r\_squared. It is also clear that there is a low p-value which is equal to zero from the above results.

> print(summary(data1\_M2\_r\_squared))

The above code gives the summary of Model 2 which is M2\_r\_squared vector of data1. The output is:

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.3239 0.3938 0.4418 0.4422 0.4811 0.6043

> print(summary(data1\_M2\_model)$r.squared). This code gives us the r-squared value for the model M2 of data1.

[1] 0.3459744

The ‘R’ code measures the prediction effectiveness of a model by calculating the total number of actual bugs discovered in the top 20% classes ranked based on the predicted number of bugs by that model on testing data and comparing 3 models based on that prediction effectiveness is as follows:

> data1\_M1\_predictions <- predict(data1\_M1\_model, newdata = testing\_data)

> data1\_M2\_predictions <- predict(data1\_M2\_model, newdata = testing\_data)

> data1\_M3\_predictions <- predict(data1\_M3\_model, newdata = testing\_data)

> output\_data <- data.frame(

+ real\_bugs <- testing\_data$bug,

+ M1\_prediction = data1\_M1\_predictions,

+ M2\_prediction = data1\_M2\_predictions,

+ M3\_prediction = data1\_M3\_predictions

+ )

> output\_data <- output\_data[order(-output\_data$M1\_prediction),]

> data1\_M1\_top20 <- round(0.2\*nrow(output\_data))

> data1\_M1\_total\_bugs <- sum(output\_data$real\_bugs[1:data1\_M1\_top20])

> output\_data <- output\_data[order(-output\_data$M2\_prediction),]

> data1\_M2\_top20 <- round(0.2\*nrow(output\_data))

> data1\_M2\_total\_bugs <- sum(output\_data$real\_bugs[1:data1\_M2\_top20])

> output\_data <- output\_data[order(-output\_data$M3\_prediction),]

> data1\_M3\_top20 <- round(0.2\*nrow(output\_data))

> data1\_M3\_total\_bugs <- sum(output\_data$real\_bugs[1:data1\_M3\_top20])

In the above code at first three predictive models are considered to make the predictions on testing\_data by implementing the predict () function. The individual predictive models are assigned to the variables such as data1\_M1\_predictions, data1\_M2\_predictioins, and data1\_M3\_predcitions.

Later we created a new data frame and named it output\_data, where the data frame combines the real bug values from the testing data and the three models' predictions and the three model predictions are named M1\_predictions, M2\_predicitons, and M3\_predicitons respectively.

Now we perform the individual model operation to get the bugs in the respective model.

First, we sort the output data according to the M1\_prediction and in descending order by referencing the order() function. The next line of code helps us to get data1\_M1\_top20 which calculates the top 20 % of data when the output data is sorted by M1\_prediction. The next line of code is data1\_M1\_total\_bugs <- sum(output\_data$real\_bugs[1:data1\_M1\_top20]), gives the number of bugs in the top 20% for M1\_predicitons. The bugs are obtained by adding the actual bugs from the top 20% of classes according to M1\_prediction.

In the same way, we sort the output data according to the M2\_prediction and in descending order by referencing the order() function. The next line of code helps us to get data1\_M2\_top20 which calculates the top 20 % of data when the output data is sorted by M2\_prediction. The next line of code is data1\_M2\_total\_bugs <- sum(output\_data$real\_bugs[1:data1\_M2\_top20]), which gives the number of bugs in the top 20% for M2\_predicitons. The bugs are obtained by adding the actual bugs from the top 20% of classes according to M2\_prediction.

At last, we sort the output data according to the M3\_prediction and in descending order by referencing the order() function. The next line of code helps us to get data1\_M3\_top20 which calculates the top 20 % of data when the output data is sorted by M3\_prediction. The next line of code is data1\_M3\_total\_bugs <- sum(output\_data$real\_bugs[1:data1\_M3\_top20]), which gives the number of bugs in the top 20% for M3\_predicitons. The bugs are obtained by adding the actual bugs from the top 20% of classes according to M3\_prediction.

Based on the given condition the total number of actual bugs discovered in the top-20% classes ranked based on the predicted number of bugs by the data1\_M1\_model on testing data are calculated by

> print(data1\_M1\_total\_bugs)

The output is:

[1] 136

Based on the given condition the total number of actual bugs discovered in the top-20% classes ranked based on the predicted number of bugs by the data1\_M2\_model on testing data are calculated by

> print(data1\_M2\_total\_bugs)

The output is:

[1] 138

Based on the given condition the total number of actual bugs discovered in the top-20% classes ranked based on the predicted number of bugs by the data1\_M3\_model on testing data are calculated by

> print(data1\_M3\_total\_bugs)

The output is:

[1] 140

The ‘R’ code for data set CSV.csv to select randomly 50% classes for training and 50% for testing and to repeat the process 30 times and

Build linear regression models to predict **bug**:

M1: using only **loc**

M2: using only **bf**

M3: using both **loc** and **bf**

And also for comparing Compare *R2* (goodness of fit) of those 3 models using paired t-test is as follows:

> library(readr)

> set.seed(450)

> no\_of\_repetitions <- 30

> data2\_M1\_r\_squared <- numeric(no\_of\_repetitions)

> data2\_M2\_r\_squared <- numeric(no\_of\_repetitions)

> data2\_M3\_r\_squared <- numeric(no\_of\_repetitions)

> for ( i in 1:no\_of\_repetitions){

+ training\_indices <- sample(1:nrow(data2), nrow(data2) / 2)

+ training\_data <- data2[training\_indices, ]

+ testing\_data <- data2[-training\_indices, ]

+ data2\_M1\_model <- lm (bug ~ loc , data=training\_data)

+ data2\_M1\_r\_squared[i] <- summary(data2\_M1\_model)$r.squared

+ data2\_M2\_model <- lm (bug ~ bf , data=training\_data)

+ data2\_M2\_r\_squared[i] <- summary(data2\_M2\_model)$r.squared

+ data2\_M3\_model <- lm (bug ~ loc + bf , data=training\_data)

+ data2\_M3\_r\_squared[i] <- summary(data2\_M3\_model)$r.squared

+ }

> t\_test\_data2\_M1\_model\_M2\_model <- t.test(data2\_M1\_r\_squared, data2\_M2\_r\_squared, paired=TRUE)

> t\_test\_data2\_M2\_model\_M3\_model <- t.test(data2\_M2\_r\_squared, data2\_M3\_r\_squared, paired=TRUE)

> t\_test\_data2\_M1\_model\_M3\_model <- t.test(data2\_M1\_r\_squared, data2\_M3\_r\_squared, paired=TRUE)

In the above line of code

At first, we considered the seed, which was set to 450(based on my last 3 digits of R#11800450) and we also considered the readr library for better operations on data present in .csv files.

Later we defined the number of repetitions to 30 which was considered according to the given question.

Now we initialize the r-squared values vector for three respective models, and we named them by following the code.

data2\_M1\_r\_squared <- numeric(no\_of\_repetitions)

data2\_M2\_r\_squared <- numeric(no\_of\_repetitions)

data2\_M3\_r\_squared <- numeric(no\_of\_repetitions), each vector size is equivalent to no\_of\_repetitions.

Now we go through a for loop to generate the linear regression model based on the given conditions.

To perform cross-validation, a for loop iterates 30 times Data is randomly divided into training and testing sets for each iteration with the same 50% classes in each by performing following code

training\_indices <- sample(1:nrow(data2), nrow(data2) / 2)

training\_data <- data2[training\_indices,]

testing\_data <- data2[-training\_indices,].

Later based on the given condition we generate the three models, and the training data are fitted to the data2\_M1\_model, data2\_M2\_model, and data2\_M3\_model linear regression models, and the resulting R-squared values are stored in the corresponding vectors. Now we perform the paired t-test between all three models to compare the goodness of fit

The output of (t\_test\_data2\_M1\_model\_M2\_model) is:

Paired t-test

data: data2\_M1\_r\_squared and data2\_M2\_r\_squared

t = -6.2857, df = 29, p-value = 7.296e-07

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

-0.4855710 -0.2471574

sample estimates:

mean difference

-0.3663642

The results suggest that the means of data2\_M1\_r\_squared and data2\_M2\_r\_squared significantly differ from one another. Data2\_M1\_r\_squared is generally less than data2\_M2\_r\_squared, as indicated by the negative mean difference of approximately -0.3664 (negative). The extremely low p-value of 7.296e-07 demonstrates that this difference is not due to chance.

> print(summary(data2\_M1\_r\_squared))

The above code gives the summary of Model 1 which is M1\_r\_squared vector of data2. The output is:

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.01149 0.02533 0.04960 0.06797 0.11519 0.14983

> print(summary(data2\_M1\_model)$r.squared)

This code gives us the r-squared value for the model M1 of data2.

[1] 0.06561307

The output of (t\_test\_data2\_M2\_model\_M3\_model) is:

Paired t-test

data: data2\_M2\_r\_squared and data2\_M3\_r\_squared

t = -8.3045, df = 29, p-value = 3.73e-09

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

-0.014976794 -0.009057631

sample estimates:

mean difference

-0.01201721

Between data2\_M2\_r\_squared and data2\_M3\_r\_squared, the results show a considerable mean difference. With a mean difference estimated to be around -0.0120 (negative), Data2\_M2\_r\_squared values are less than Data2\_M3\_r\_squared. Strong proof that this difference is not a result of chance by considering extreme low p-value of 3.73e-09.

> print(summary(data2\_M2\_r\_squared))

The above code gives the summary of Model 2 which is M2\_r\_squared vector of data2. The output is:

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.1122 0.1723 0.4466 0.4343 0.7078 0.7363

> print(summary(data2\_M2\_model)$r.squared)

This code gives us the r-squared value for the model M2 of data2.

[1] 0.112176

The output of (t\_test\_data2\_M1\_model\_M3\_model) is:

Paired t-test

data: data2\_M1\_r\_squared and data2\_M3\_r\_squared

t = -6.4122, df = 29, p-value = 5.174e-07

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

-0.4990688 -0.2576940

sample estimates:

mean difference

-0.3783814

The results show a significant difference between the means of data2\_M1\_r\_squared and data2\_M3\_r\_squared. The estimated mean difference between Data2\_M1\_r\_squared and Data2\_M3\_r\_squared is approximately -0.3784 (negative). By the extremely low p-value of 5.174e-07, indicating that this difference is not the result of chance.

> print(summary(data2\_M3\_r\_squared))

The above code gives the summary of Model 3 which is M3\_r\_squared vector of data2. The output is:

Min. 1st Qu. Median Mean 3rd Qu. Max.

0.1126 0.1850 0.4599 0.4463 0.7229 0.7434

> print(summary(data2\_M3\_model)$r.squared)

This code gives us the r-squared value for the model M3 of data2.

[1] 0.1154984

The ‘R’ code measures the prediction effectiveness of a model by calculating the total number of actual bugs discovered in the top 20% of classes ranked based on the predicted number of bugs by that model on testing data and comparing 3 models based on that prediction effectiveness is as follows:

> data2\_M1\_predictions <- predict(data2\_M1\_model, newdata = testing\_data)

> data2\_M2\_predictions <- predict(data2\_M2\_model, newdata = testing\_data)

> data2\_M3\_predictions <- predict(data2\_M3\_model, newdata = testing\_data)

> output\_data <- data.frame(

+ real\_bugs <- testing\_data$bug,

+ M1\_prediction = data2\_M1\_predictions,

+ M2\_prediction = data2\_M2\_predictions,

+ M3\_prediction = data2\_M3\_predictions

+ )

> output\_data <- output\_data[order(-output\_data$M1\_prediction),]

> data2\_M1\_top20 <- round(0.2\*nrow(output\_data))

> data2\_M1\_total\_bugs <- sum(output\_data$real\_bugs[1:data2\_M1\_top20])

> output\_data <- output\_data[order(-output\_data$M2\_prediction),]

> data2\_M2\_top20 <- round(0.2\*nrow(output\_data))

> data2\_M2\_total\_bugs <- sum(output\_data$real\_bugs[1:data2\_M2\_top20])

> output\_data <- output\_data[order(-output\_data$M3\_prediction),]

> data2\_M3\_top20 <- round(0.2\*nrow(output\_data))

> data2\_M3\_total\_bugs <- sum(output\_data$real\_bugs[1:data2\_M3\_top20])

In the above code at first three predictive models are considered to make the predictions on testing\_data by implementing the predict () function. The individual predictive models are assigned to the variables such as data2\_M1\_predictions, data2\_M2\_predictioins, and data2\_M3\_predcitions.

Later we created a new data frame and named it output\_data, where the data frame combines the real bug values from the testing data and the three models' predictions and the three model predictions are named M1\_predictions, M2\_predicitons, and M3\_predicitons respectively.

Now we perform the individual model operation to get the bugs in the respective model.

First, we sort the output data according to the M1\_prediction and in descending order by referencing the order() function. The next line of code helps us to get data2\_M1\_top20 which calculates the top 20 % of data when the output data is sorted by M1\_prediction. The next line of code is data2\_M1\_total\_bugs <- sum(output\_data$real\_bugs[1:data2\_M1\_top20]), which gives the number of bugs in the top 20% for M1\_predicitons. The bugs are obtained by adding the actual bugs from the top 20% of classes according to M1\_prediction.

In the same way, we sort the output data according to the M2\_prediction and in descending order by referencing the order() function. The next line of code helps us to get data2\_M2\_top20 which calculates the top 20 % of data when the output data is sorted by M2\_prediction. The next line of code is data2\_M2\_total\_bugs <- sum(output\_data$real\_bugs[1:data2\_M2\_top20]), which gives the number of bugs in the top 20% for M2\_predicitons. The bugs are obtained by adding the actual bugs from the top 20% of classes according to M2\_prediction.

At last, we sort the output data according to the M3\_prediction and in descending order by referencing the order() function. The next line of code helps us to get data2\_M3\_top20 which calculates the top 20 % of data when the output data is sorted by M3\_prediction. The next line of code is data2\_M3\_total\_bugs <- sum(output\_data$real\_bugs[1:data2\_M3\_top20]), which gives the number of bugs in the top 20% for M3\_predicitons. The bugs are obtained by adding the actual bugs from the top 20% of classes according to M3\_prediction.

Based on the given condition the total number of actual bugs discovered in the top-20% classes ranked based on the predicted number of bugs by the data2\_M1\_model on testing data are calculated by

> print(data2\_M1\_total\_bugs)

The output is:

[1] 97

Based on the given condition the total number of actual bugs discovered in the top-20% classes ranked based on the predicted number of bugs by the data2\_M2\_model on testing data are calculated by

> print(data2\_M2\_total\_bugs)

The output is:

[1] 139

Based on the given condition the total number of actual bugs discovered in the top-20% classes ranked based on the predicted number of bugs by the data2\_M3\_model on testing data are calculated by

> print(data2\_M3\_total\_bugs)

The output is:

[1] 139

Q2. Train M3 on JDT and test on PDE. Then, train M3 on PDE and test on JDT. Compare the goodness of fit and prediction effectiveness in those cross-project prediction settings with cross-validation settings in Q1. To perform many experiments for statistical testing, you can repeat the training/testing process 30 times, each with a random sample from the training dataset (e.g., JDT) and a random sample from the testing dataset (e.g., PDE).

The ‘R’ code for the above question is as follows:

> library(readr)

> library(dplyr)

> set.seed(450)

> no\_of\_repetitions <- 30

>

> data1\_to\_data2\_r\_squared <- numeric(no\_of\_repetitions)

> data2\_to\_data1\_r\_squared <- numeric(no\_of\_repetitions)

> data1\_to\_data2\_predictions <- numeric(no\_of\_repetitions)

> data2\_to\_data1\_predictions <- numeric(no\_of\_repetitions)

>

> for (i in 1:no\_of\_repetitions) {

+ required\_training\_data <- sample\_n(data1, nrow(data1) / 2)

+ required\_testing\_data <- sample\_n(data2, nrow(data2) / 2)

+

+ M3\_model\_data1\_to\_data2 <- lm(bug ~ loc + bf, data = required\_training\_data)

+ predictions\_data1\_to\_data2 <- predict(M3\_model\_data1\_to\_data2, newdata = required\_testing\_data)

+ r\_squared\_data1\_to\_data2 <- 1 - sum((required\_testing\_data$bug - predictions\_data1\_to\_data2)^2) / sum((required\_testing\_data$bug - mean(required\_testing\_data$bug))^2)

+

+ required\_testing\_data <- required\_testing\_data %>% arrange(desc(predictions\_data1\_to\_data2))

+ index\_top\_20\_data2 <- round(0.2 \* nrow(required\_testing\_data))

+ data\_top\_20\_data2 <- head(required\_testing\_data, index\_top\_20\_data2)

+ total\_bugs\_data1\_to\_data2 <- sum(data\_top\_20\_data2$bug)

+

+ data1\_to\_data2\_r\_squared[i] <- r\_squared\_data1\_to\_data2

+ data1\_to\_data2\_predictions[i] <- total\_bugs\_data1\_to\_data2

+

+ resultant\_training\_data <- sample\_n(data2, nrow(data2) / 2)

+ resultant\_testing\_data <- sample\_n(data1, nrow(data1) / 2)

+

+ M3\_model\_data2\_to\_data1 <- lm(bug ~ loc + bf, data = resultant\_training\_data)

+ predictions\_data2\_to\_data1 <- predict(M3\_model\_data2\_to\_data1, newdata = resultant\_testing\_data)

+

+ r\_squared\_data2\_to\_data1 <- 1 - sum((resultant\_testing\_data$bug - predictions\_data2\_to\_data1)^2) / sum((resultant\_testing\_data$bug - mean(resultant\_testing\_data$bug))^2)

+

+ resultant\_testing\_data <- resultant\_testing\_data %>% arrange(desc(predictions\_data2\_to\_data1))

+ index\_top\_20\_data1 <- round(0.2 \* nrow(resultant\_testing\_data))

+ data\_top\_20\_data1 <- head(resultant\_testing\_data, index\_top\_20\_data1)

+ total\_bugs\_data2\_to\_data1 <- sum(data\_top\_20\_data1$bug)

+

+ data2\_to\_data1\_r\_squared[i] <- r\_squared\_data2\_to\_data1

+ data2\_to\_data1\_predictions[i] <- total\_bugs\_data2\_to\_data1

+ }

>

> mean\_data1\_to\_data2\_r\_squared <- mean(data1\_to\_data2\_r\_squared)

> mean\_data2\_to\_data1\_r\_squared <- mean(data2\_to\_data1\_r\_squared)

> mean\_data1\_to\_data2\_predictions <- mean(data1\_to\_data2\_predictions)

> mean\_data2\_to\_data1\_predictions <- mean(data2\_to\_data1\_predictions)

In the above code

At first, we considered the seed, which was set to 450(based on my last 3 digits of R#11800450).

Later we defined the number of repetitions to 30 which was considered according to the given question.

Now, we declare the variables which are the vectors that store data of r squared values, and the data for prediction effectiveness of linear regression model M3.

Now we iterate through a for loop based on a given condition and at first, we generate the training data and testing data where each of them contains a random of 50% classes. In the first case, the training data was selected randomly from the JDT.csv, and testing data was selected randomly from PDE.csv datasets. In the next case, it is vice versa. Focusing on the first case after having training data and testing data we generate a model M3 which is a linear regression model using both loc and bf as predictors. These models are trained on the training data, and predictions are made on the testing data. The predictions are made by referencing the predict () function. Later we calculate the r-square values by summing the squared differences between actual bug values and predicted values and dividing the whole respective sum by the sum of squared differences between actual bug values and their mean and this tells us how the model M3 details the variance in the dependent variables. The condition to calculate r-square values was given accordingly, Now we sort the testing data according to M3 model prediction, and top 20 % of data is selected for further process. This is done by the code required\_testing\_data <- required\_testing\_data %>% arrange(desc(predictions\_data1\_to\_data2)).The overall number of real bugs discovered in the top 20% of cases is later calculated. This gauges how well predictions work. We store the values of the resultant r -square in data1\_to\_data2\_r\_squared in case 1 and the values of prediction effectiveness in data1\_to\_data2\_predictions variables respectively.

In the same manner, we generate the r-square values and values of prediction effectiveness for case 2 by considering the training data as a PDE.csv dataset which is data2, and testing as a JDT.csv dataset which is data1. Here we follow the same pattern in determining r square values and values of prediction effectiveness and store them in variables called data2\_to\_data1\_r\_squared and data2\_to\_data1\_predictions respectively,

After the for loop iteration was completed we calculated the mean of r squares and prediction effectiveness which are obtained in both cases.

The output of case 1:

> print(mean\_data1\_to\_data2\_r\_squared)

[1] 0.1961011

> print(mean\_data1\_to\_data2\_predictions)

[1] 110.7667

The output of case 2:

> print(mean\_data2\_to\_data1\_r\_squared)

[1] -0.7270016

> print(mean\_data2\_to\_data1\_predictions)

[1] 128.4

From the above outputs, we can state that,

The mean r-squared value in the case where M3 is trained on data1 and tested on data2 is roughly 0.1961, indicating a fair fit. The mean r-squared value in the reverse scenario, when M3 trained on data2 and tested on data1, is roughly -0.7270, showing a less good fit and possibly indicating that the model doesn't generalize well across datasets.

For the scenario where M3 is trained on data1 and tested on data2, the mean prediction is roughly 110.77, while the mean prediction is roughly 128.4 for the flip scenario where M3 is trained on data2 and tested on data1.

According to these findings, training M3 on JDT.csv data and testing it on PDE.csv data produces better goodness of fit than the opposite scenario.

Now we calculate the t-test between r-squared values of data1\_to\_data2\_r\_squared and data2\_data1\_r\_squared and also for predictions such as data1\_to\_data2\_predictions and data2\_to\_data1\_predictions as follows:

>t\_test\_r\_sqaured <- t.test(data1\_to\_data2\_r\_squared, data2\_to\_data1\_r\_squared, paired = TRUE)

>t\_test\_predictions <- t.test (data1\_to\_data2\_predictions, data2\_to\_data1\_predictions, paired = TRUE)

The output of r\_squared t-test results is as follows:

Paired t-test

data: data1\_to\_data2\_r\_squared and data2\_to\_data1\_r\_squared

t = 5.2012, df = 29, p-value = 1.45e-05

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

0.5601179 1.2860875

sample estimates:

mean difference

0.9231027

The output of predictions t-test results is as follows:

Paired t-test

data: data1\_to\_data2\_predictions and data2\_to\_data1\_predictions

t = -4.6555, df = 29, p-value = 6.602e-05

alternative hypothesis: true mean difference is not equal to 0

95 percent confidence interval:

-25.379852 -9.886815

sample estimates:

mean difference

-17.63333

Note:

* <https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/sample>

I have referred to this website to understand how to split the data randomly from a given data set.

* <https://dplyr.tidyverse.org/articles/dplyr.html>

I have referred to this website to get to know how to sort the data in required conditions such as sorting the dataset, arranging datasets, and selecting the top 20% of data.