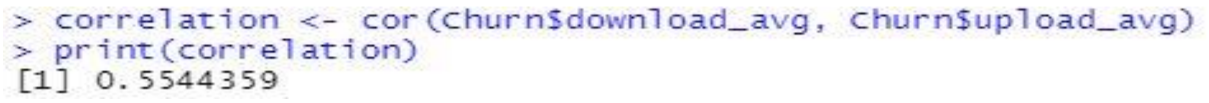
**The dataset is taken from** Kunt, Mehmet Sabri. n.d. “ISP Data from Kaggle.” https://www.kaggle.com/datasets/mehmetsabrikunt/internet-service-churn.

**Task 1**

a)

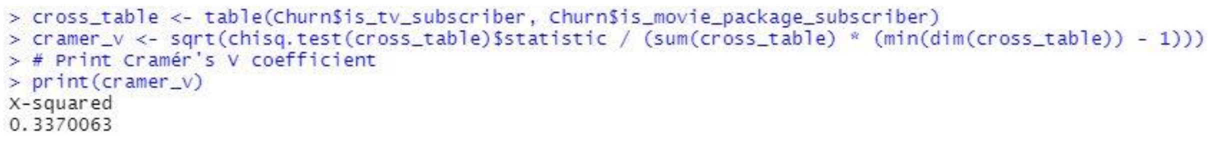
After using the t-test and chi-square test to measure the significance of the independent variables, it seems that all the variables are significant, we can remove some variables because they are related to another variable, so they provide the same information, it is not necessary that both would include:



```
> correlation <- cor(Churn$download_avg, Churn$upload_avg)
> print(correlation)
[1] 0.5544359
```

**Kramer's V** coefficient measures the strength of association between two categorical variables on a scale from 0 to 1. Higher values indicate stronger associations, while

0 means independent.

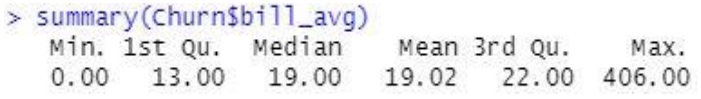


```
> cross_table <- table(Churn$is_tv_subscriber, Churn$is_movie_package_subscriber)
> cramer_v <- sqrt(chisq.test(cross_table)$statistic / (sum(cross_table) * (min(dim(cross_table)) - 1)))
> # Print Cramér's V coefficient
> print(cramer_v)
X-squared
0.3370063
```

A value of **0.3370063** suggests that there is a meaningful relationship or association between the two categorical variables.

b)

summary table :

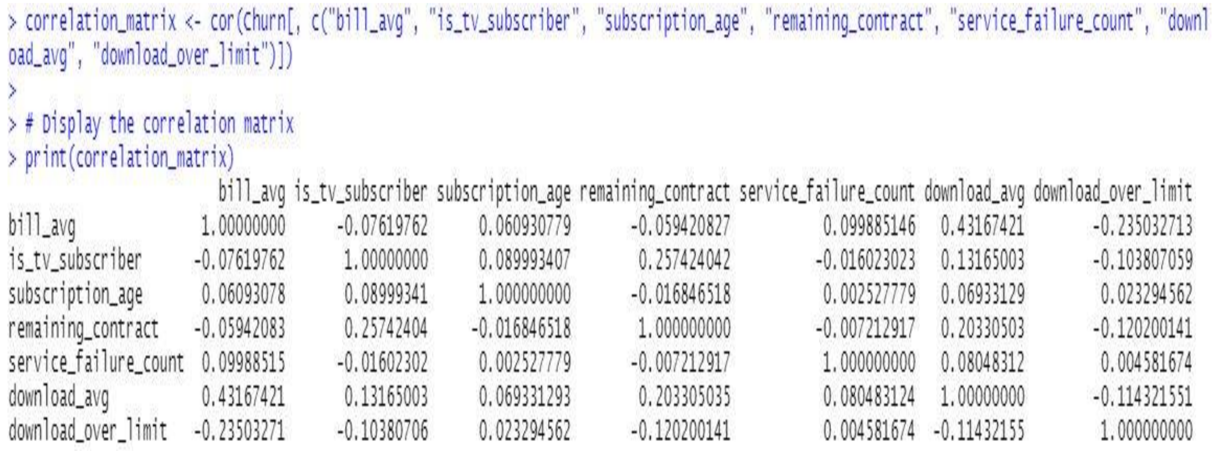


```
> summary(Churn$bill_avg)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   0.00   13.00   19.00   19.02   22.00  406.00 
```

The minimum value is 0, indicating that some customers have a bill average of zero, which suggests they might have free or discounted services.

The maximum value is 406, suggesting the presence of some extreme values or outliers in the data.

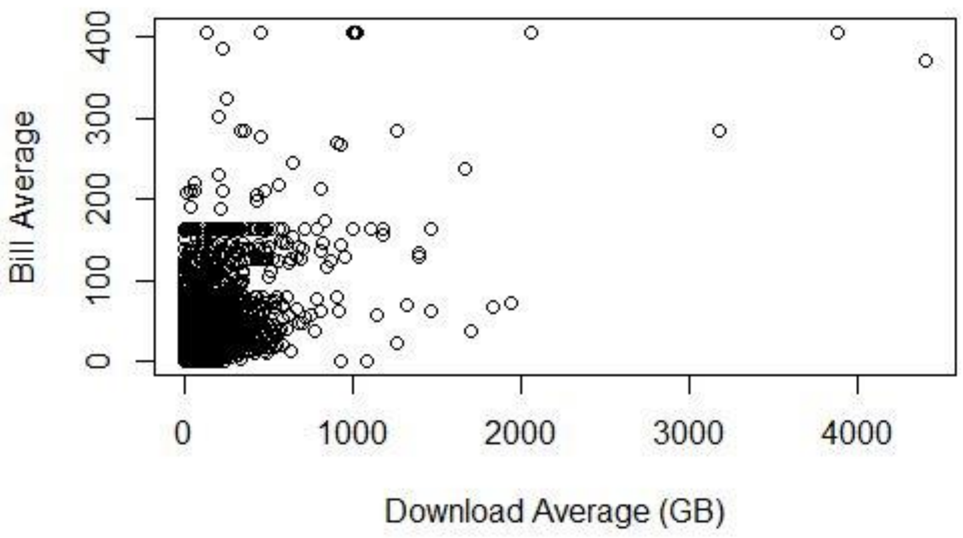
**Correlation matrix :**



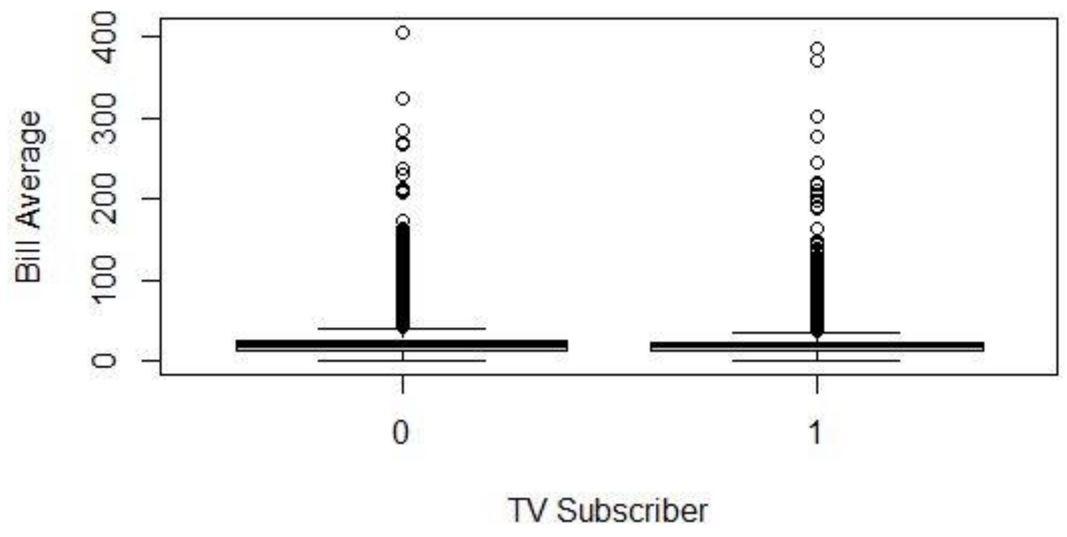
```
> correlation_matrix <- cor(Churn[, c("bill_avg", "is_tv_subscriber", "subscription_age", "remaining_contract", "service_failure_count", "downl
oad_avg", "download_over_limit")])
>
> # Display the correlation matrix
> print(correlation_matrix)
                         bill_avg is_tv_subscriber subscription_age remaining_contract service_failure_count download_avg download_over_limit
bill_avg               1.00000000      -0.07619762      0.060930779       -0.059420827           0.099885146   0.43167421        -0.235032713
is_tv_subscriber      -0.07619762       1.00000000      0.089993407        0.257424042          -0.016023023   0.13165003        -0.103807059
subscription_age       0.06093078       0.08999341      1.000000000       -0.016846518           0.002527779   0.06933129         0.023294562
remaining_contract    -0.05942083       0.25742404     -0.016846518        1.000000000          -0.007212917   0.20330503        -0.120200141
service_failure_count  0.09988515      -0.01602302      0.002527779       -0.007212917           1.000000000   0.08048312         0.004581674
download_avg           0.43167421       0.13165003      0.069331293        0.203305035           0.080483124   1.00000000        -0.114321551
download_over_limit   -0.23503271      -0.10380706      0.023294562       -0.120200141           0.004581674  -0.11432155         1.000000000
```

Based on this correlation matrix, average download usage(download_avg) and download_over_limit stand out as potentially good predictors for the average bill.

**Plots :**



The chart appears to show a positive correlation between average downloads and average bills, indicating that customers with higher average downloads tend to have higher average bills.

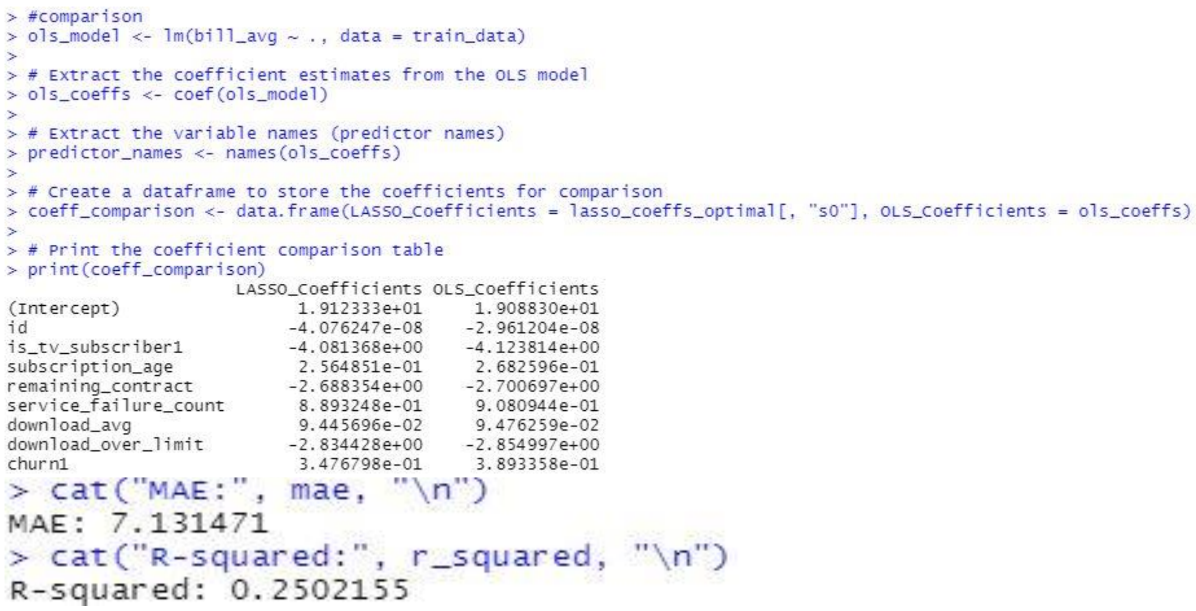


This boxplot provides a visual overview of the breakdown of average bills for TV subscribers and non-subscribers. This can help identify differences in central tendency, variability, and potential outliers between two groups.

c)

Based on the previous analysis, we can get the best possible results from the ols model while optimizing it by reducing the variables we're using as below.

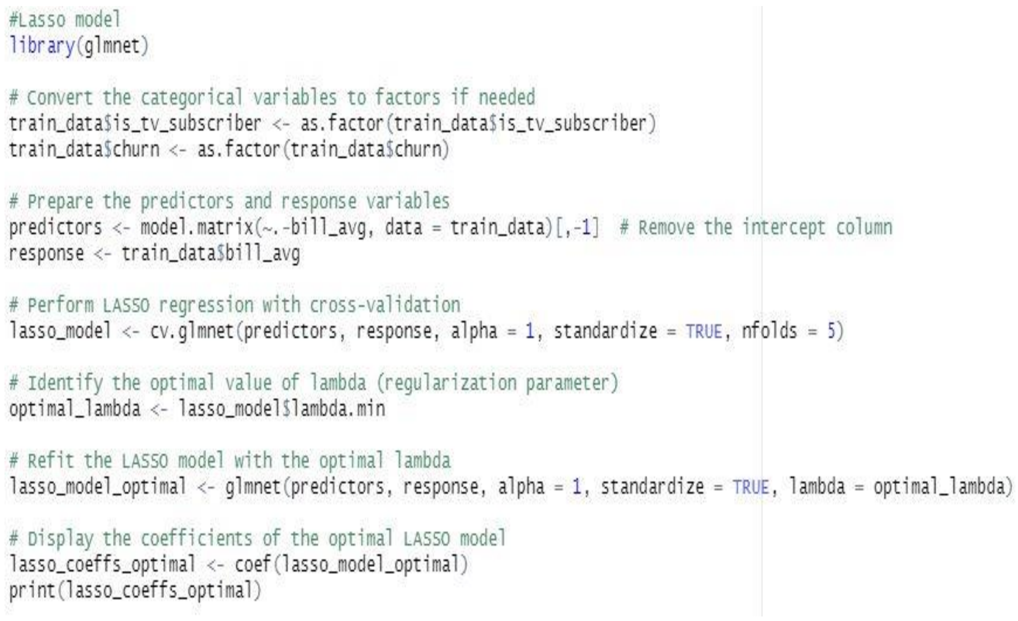
**Model :**



```
> #comparison
> ols_model <- lm(bill_avg ~ ., data = train_data)
>
> # Extract the coefficient estimates from the OLS model
> ols_coeffs <- coef(ols_model)
>
> # Extract the variable names (predictor names)
> predictor_names <- names(ols_coeffs)
>
> # Create a dataframe to store the coefficients for comparison
> coeff_comparison <- data.frame(LASSO_Coefficients = lasso_coeffs_optimal[, "s0"], OLS_Coefficients = ols_coeffs)
>
> # Print the coefficient comparison table
> print(coeff_comparison)
                      LASSO_Coefficients OLS_Coefficients
(Intercept)                 1.912333e+01     1.908830e+01
id                         -4.076247e-08    -2.961204e-08
is_tv_subscriber1          -4.081368e+00    -4.123814e+00
subscription_age            2.564851e-01     2.682596e-01
remaining_contract         -2.688354e+00    -2.700697e+00
service_failure_count       8.893248e-01     9.080944e-01
download_avg                9.445696e-02     9.476259e-02
download_over_limit        -2.834428e+00    -2.854997e+00
churn1                      3.476798e-01     3.893358e-01
> cat("MAE:", mae, "\n")
MAE: 7.131471
> cat("R-squared:", r_squared, "\n")
R-squared: 0.2502155
```

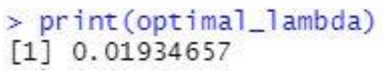
The model shows a modest predictive performance with small errors and explains only a small part of the variability in bill_avg.

d)



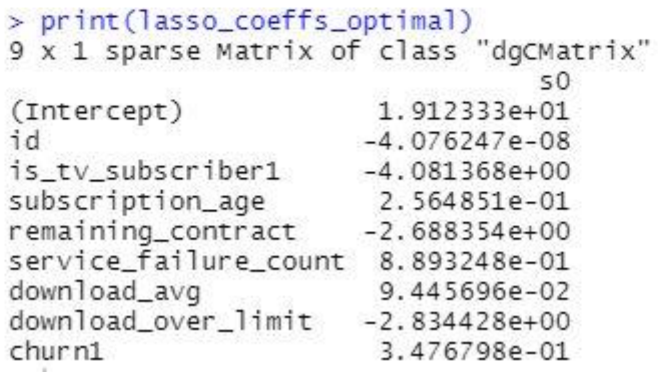
```
#Lasso model
library(glmnet)

# Convert the categorical variables to factors if needed
train_data$is_tv_subscriber <- as.factor(train_data$is_tv_subscriber)
train_data$churn <- as.factor(train_data$churn)

# Prepare the predictors and response variables
predictors <- model.matrix(~.-bill_avg, data = train_data)[,-1]  # Remove the intercept column
response <- train_data$bill_avg

# Perform LASSO regression with cross-validation
lasso_model <- cv.glmnet(predictors, response, alpha = 1, standardize = TRUE, nfolds = 5)

# Identify the optimal value of lambda (regularization parameter)
optimal_lambda <- lasso_model$lambda.min

# Refit the LASSO model with the optimal lambda
lasso_model_optimal <- glmnet(predictors, response, alpha = 1, standardize = TRUE, lambda = optimal_lambda)

# Display the coefficients of the optimal LASSO model
lasso_coeffs_optimal <- coef(lasso_model_optimal)
print(lasso_coeffs_optimal)
```

To tune the LASSO model parameters, you can use cross-validation to find the best value of the regularization parameter (lambda) that gives the best model performance.

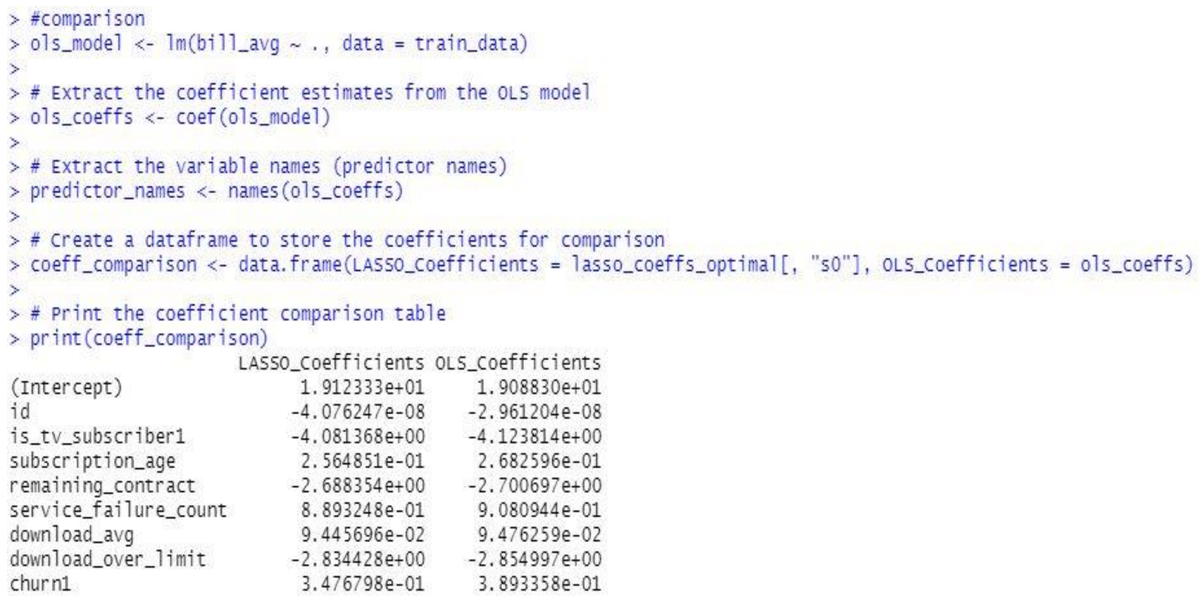


```
> print(optimal_lambda)
[1] 0.01934657
```

Standardization of variables in LASSO regression is a common practice for several reasons, the predictors have different scales or units, and the magnitude of the coefficients can be misleading. Standardization of LASSO regression variables aids interpretation, improves model performance, and ensures fair penalties and comparison of predictors.



```
> print(lasso_coeffs_optimal)
9 x 1 sparse Matrix of class "dgCMatrix"
                                s0
(Intercept)           1.912333e+01
id                   -4.076247e-08
is_tv_subscriber1    -4.081368e+00
subscription_age      2.564851e-01
remaining_contract   -2.688354e+00
service_failure_count 8.893248e-01
download_avg          9.445696e-02
download_over_limit  -2.834428e+00
churn1                3.476798e-01
```



```
> #comparison
> ols_model <- lm(bill_avg ~ ., data = train_data)
>
> # Extract the coefficient estimates from the OLS model
> ols_coeffs <- coef(ols_model)
>
> # Extract the variable names (predictor names)
> predictor_names <- names(ols_coeffs)
>
> # Create a dataframe to store the coefficients for comparison
> coeff_comparison <- data.frame(LASSO_Coefficients = lasso_coeffs_optimal[, "s0"], OLS_Coefficients = ols_coeffs)
>
> # Print the coefficient comparison table
> print(coeff_comparison)
                      LASSO_Coefficients OLS_Coefficients
(Intercept)                 1.912333e+01     1.908830e+01
id                         -4.076247e-08    -2.961204e-08
is_tv_subscriber1          -4.081368e+00    -4.123814e+00
subscription_age            2.564851e-01     2.682596e-01
remaining_contract         -2.688354e+00    -2.700697e+00
service_failure_count       8.893248e-01     9.080944e-01
download_avg                9.445696e-02     9.476259e-02
download_over_limit        -2.834428e+00    -2.854997e+00
churn1                      3.476798e-01     3.893358e-01
```

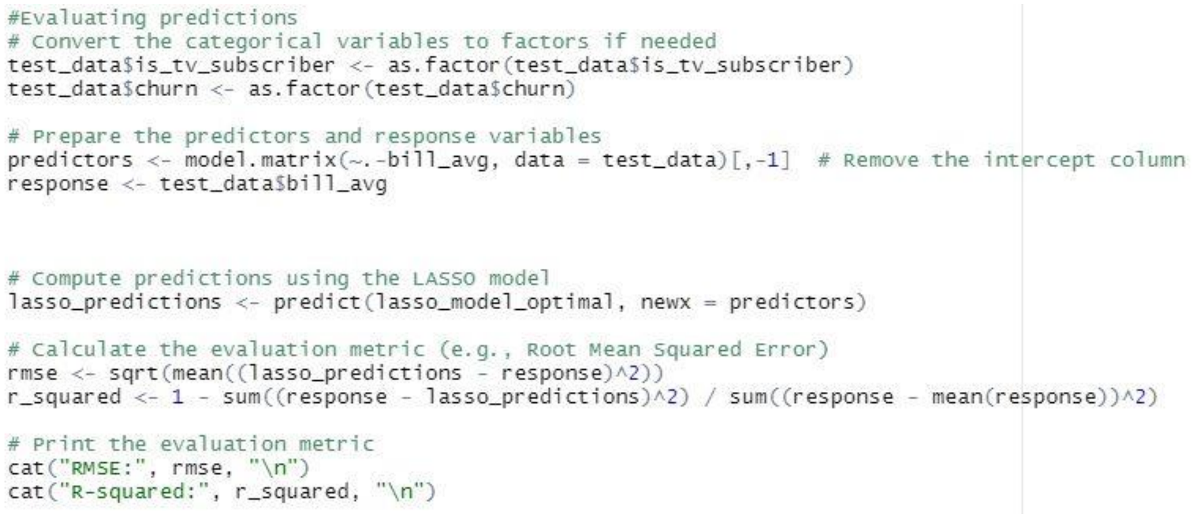
**Predictors:** The predictor coefficients are slightly different in the LASSO and OLS models. However, the difference is relatively small.

**Similar magnitudes:** In general, the magnitude of the prediction coefficients is similar between the LASSO and OLS models. This suggests that the variables are equally important in explaining the change in bill_avg.

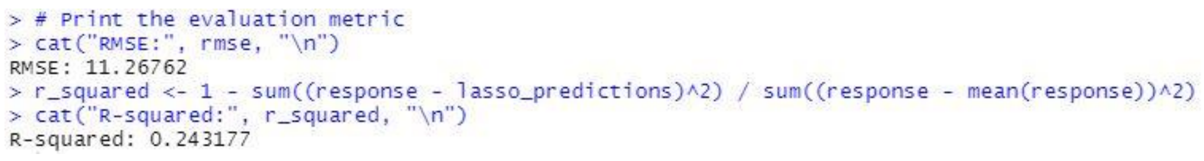
**Variable Importance:** Both models show that variables such as is_tv_subscriber, remaining_contract, download_over_limit, and churn1 have a significant impact on bill_avg. forecasting. However, the magnitude of the coefficients may differ slightly between the two models.

Overall, the LASSO model and the OLS model provide relatively consistent results in predicting the bill_avg coefficient. This shows that the selected predictors have a significant effect on bill_avg and that the LASSO model effectively captures their importance while accounting for regularization effects.

e)



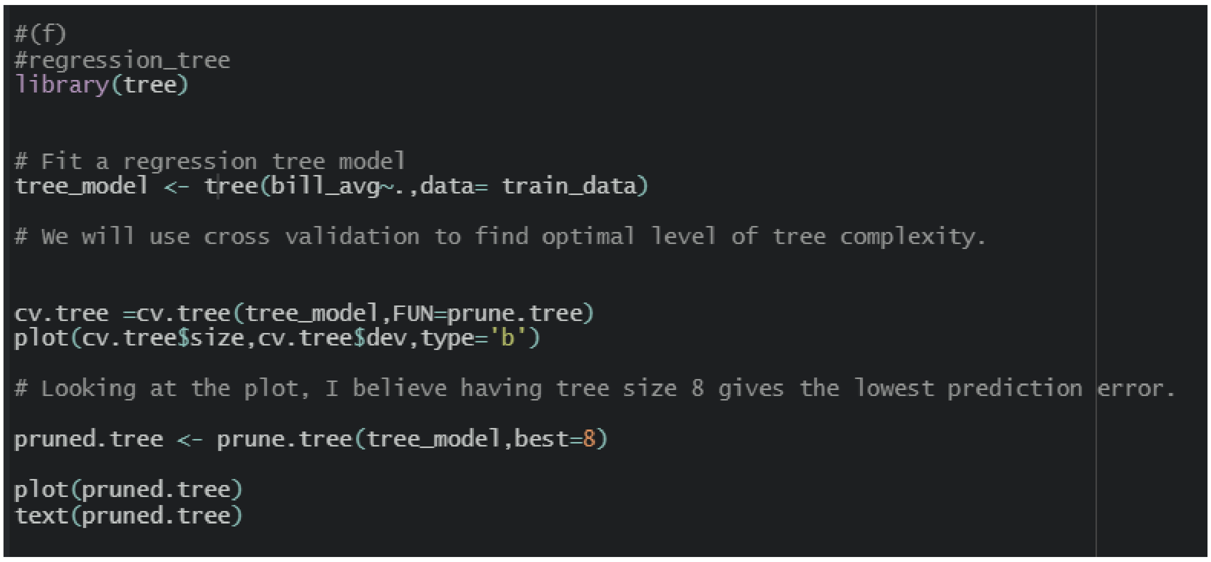
```
#Evaluating predictions
# Convert the categorical variables to factors if needed
test_data$is_tv_subscriber <- as.factor(test_data$is_tv_subscriber)
test_data$churn <- as.factor(test_data$churn)

# Prepare the predictors and response variables
predictors <- model.matrix(~.-bill_avg, data = test_data)[,-1]  # Remove the intercept column
response <- test_data$bill_avg


# Compute predictions using the LASSO model
lasso_predictions <- predict(lasso_model_optimal, newx = predictors)

# Calculate the evaluation metric (e.g., Root Mean Squared Error)
rmse <- sqrt(mean((lasso_predictions - response)^2))
r_squared <- 1 - sum((response - lasso_predictions)^2) / sum((response - mean(response))^2)

# Print the evaluation metric
cat("RMSE:", rmse, "\n")
cat("R-squared:", r_squared, "\n")
```



```
> # Print the evaluation metric
> cat("RMSE:", rmse, "\n")
RMSE: 11.26762
> r_squared <- 1 - sum((response - lasso_predictions)^2) / sum((response - mean(response))^2)
> cat("R-squared:", r_squared, "\n")
R-squared: 0.243177
```

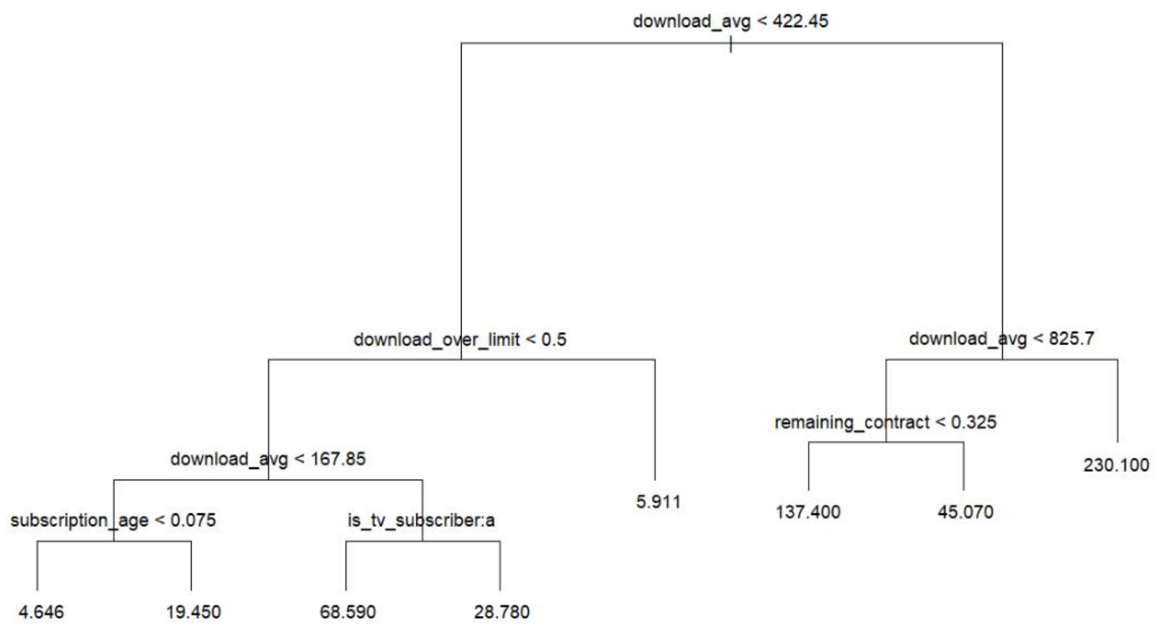
An **R-squared** value of **0.243177** indicates that the LASSO model explains approximately **24.3%** of the variance in the response variable. Although this suggests that the model captures some of the underlying patterns in the data, a significant portion of the variance remains unexplained.

f)



```
#(f)
#regression_tree
library(tree)


# Fit a regression tree model
tree_model <- tree(bill_avg~.,data= train_data)

# We will use cross validation to find optimal level of tree complexity.


cv.tree =cv.tree(tree_model,FUN=prune.tree)
plot(cv.tree$size,cv.tree$dev,type='b')

# Looking at the plot, I believe having tree size 8 gives the lowest prediction error.

pruned.tree <- prune.tree(tree_model,best=8)

plot(pruned.tree)
text(pruned.tree)
```

The tree we have :



# The plot show the first split comes from download_avg (smaller than 422.45 goes left)

# One the left, next split is download_over_limit < 0.5. We have a prediction if

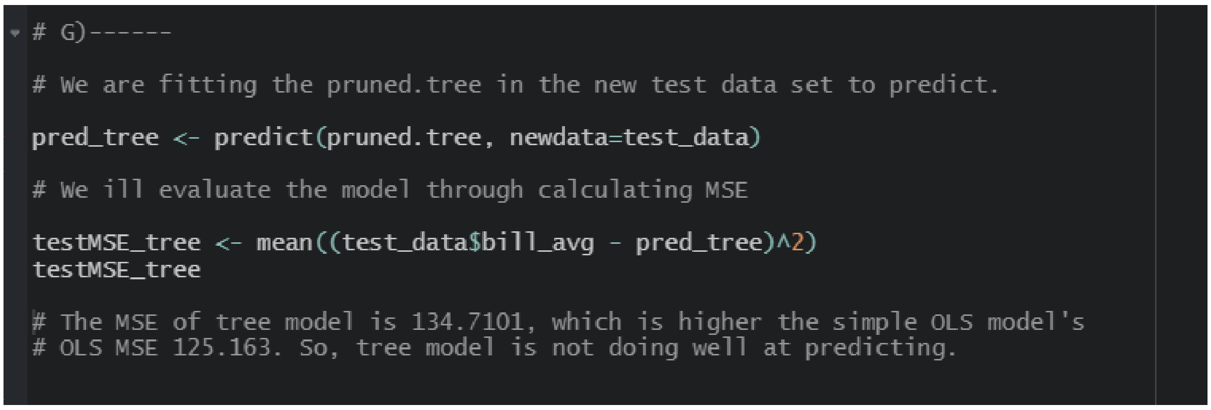
# it is more than 0.5 and, if not, next splits comes from upload_avg, and then

# subscription_age and is_tv_subscriber. The right side of download_avg split has

# split at download_avg, right side prediction and left side another split from

# remaining_contract.

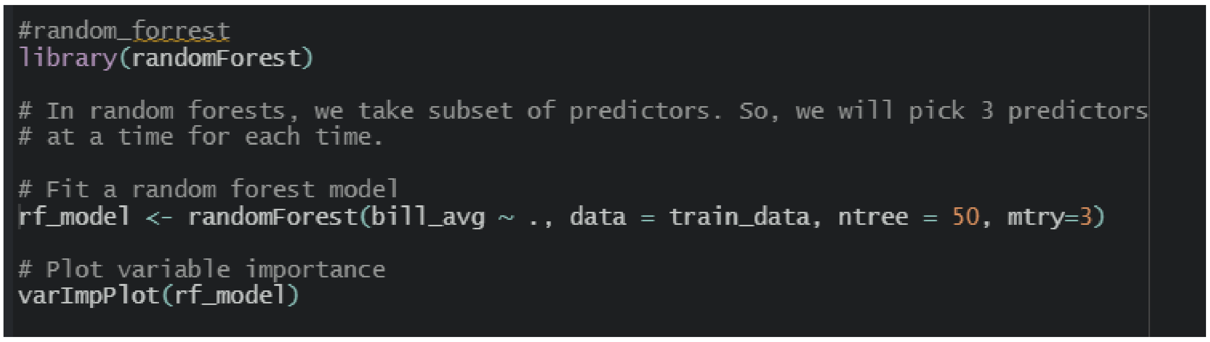
g)



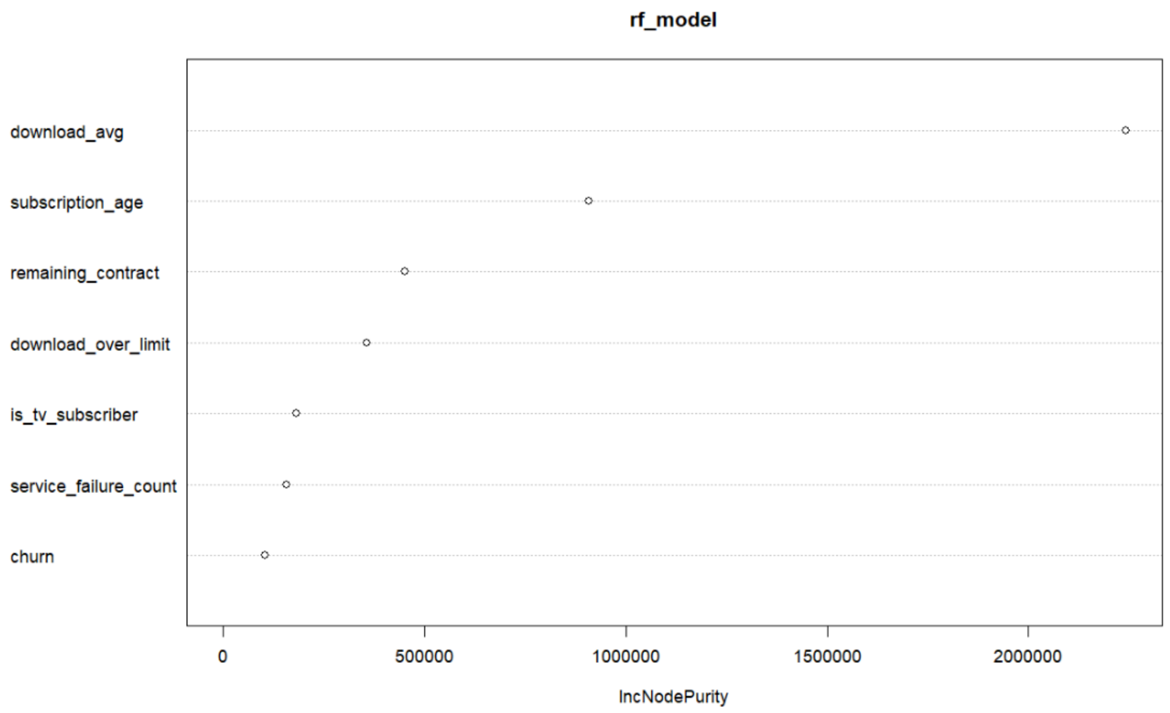
```r
# G)------

# We are fitting the pruned.tree in the new test data set to predict.

pred_tree <- predict(pruned.tree, newdata=test_data)

# We ill evaluate the model through calculating MSE

testMSE_tree <- mean((test_data$bill_avg - pred_tree)^2)
testMSE_tree

# The MSE of tree model is 134.7101, which is higher the simple OLS model's
# OLS MSE 125.163. So, tree model is not doing well at predicting.
```

h)

In random forests, we take subset of predictors. So, we will pick 3 predictors at a time for each time. Number of tree we are setting is 50 to make it fast.



```r
#random_forrest
library(randomForest)

# In random forests, we take subset of predictors. So, we will pick 3 predictors
# at a time for each time.

# Fit a random forest model
rf_model <- randomForest(bill_avg ~ ., data = train_data, ntree = 50, mtry=3)

# Plot variable importance
varImpPlot(rf_model)
```



From above variable of importance plot we find :

The IncNodePurity measures represents the average improvement in the model's node purity (impurity reduction) achieved by each predictor variable across all the trees in the random forest.Higher values of IncNodePurity indicate that the variable has a stronger influence on the model's predictions. It signifies that the variable plays a more significant role in reducing the impurity (e.g., Gini index for classification and residual sum of squares for regression) and improving the accuracy of the model.

The output of the varImpPlot() function provides a measure of variable significance in a random forest model. It ranks predictors by their contribution to model accuracy.

**download_avg**: This predictor is considered the most important for predicting bill_avg under the random forest model. This shows that the average download size has the greatest effect on the target variable.

**subscription_age**: This predictor is the second most important, indicating that subscription age affects bill_avg. forecast.

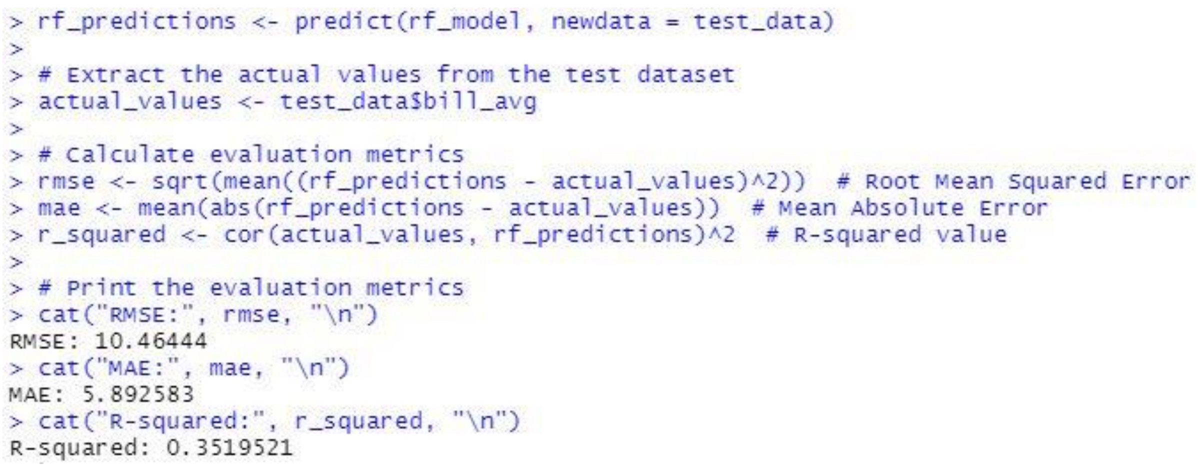
**download_over_limit**: This prediction is the third most important. This suggests that whether a user has exceeded the download limit plays a large role in determining bill_avg.

**remaining_contract**: This prediction is the fourth most important. This means that the remaining contract period has some effect on bill_avg.

**is_tv_subscriber**: This prediction ranks fifth in importance. This shows that whether a user is a TV subscriber has some effect on bill_avg. forecasting.

**service_failure_count**: This prediction is the sixth most important. This shows that the number of service failures has a moderate effect on bill_avg.

i)



```
> rf_predictions <- predict(rf_model, newdata = test_data)
>
> # Extract the actual values from the test dataset
> actual_values <- test_data$bill_avg
>
> # Calculate evaluation metrics
> rmse <- sqrt(mean((rf_predictions - actual_values)^2))  # Root Mean Squared Error
> mae <- mean(abs(rf_predictions - actual_values))  # Mean Absolute Error
> r_squared <- cor(actual_values, rf_predictions)^2  # R-squared value
>
> # Print the evaluation metrics
> cat("RMSE:", rmse, "\n")
RMSE: 10.46444
> cat("MAE:", mae, "\n")
MAE: 5.892583
> cat("R-squared:", r_squared, "\n")
R-squared: 0.3519521
```

Comparing these metrics to previous models, the random forest model appears to perform better in terms of RMSE, MAE, and R-squared. This means that the random forest model provides more accurate predictions and captures more of the variation in the variable bill_avg than the previous model.

j)

Characteristics of customers with high average bills:

**Download Average:** Customers with high average bills tend to have higher average downloads. This indicates that they are using more data or internet usage.

**Subscription Age:** Customers with high average bills tend to have older subscriptions. This indicates that they have been working with the service provider for a longer period of time.

**Download Over Limit:** Customers with high average bills can often exceed their data limits, resulting in additional charges.

**Remaining Contract:** Customers with high average bills may have a longer remaining contract. This means they commit to long-term contracts that may include more expensive plans or additional services.

**TV Subscription:** Customers with high average bills are more likely to subscribe to TV services. This suggests that they can opt for a bundle or higher plan that includes TV service.

**Service Failure Count:** Customers with higher average bills are likely to experience more service failures. This may be related to other services or features they use that are more likely to cause problems.

Characteristics of customers with low average bills:

**Lower Download Average:** Customers with lower average bills have relatively lower average downloads. They may have lower data usage or use less internet.

**Shorter Subscription Age:** Customers with lower average bills tend to have a younger subscription age. These may be new customers or those with shorter contracts.

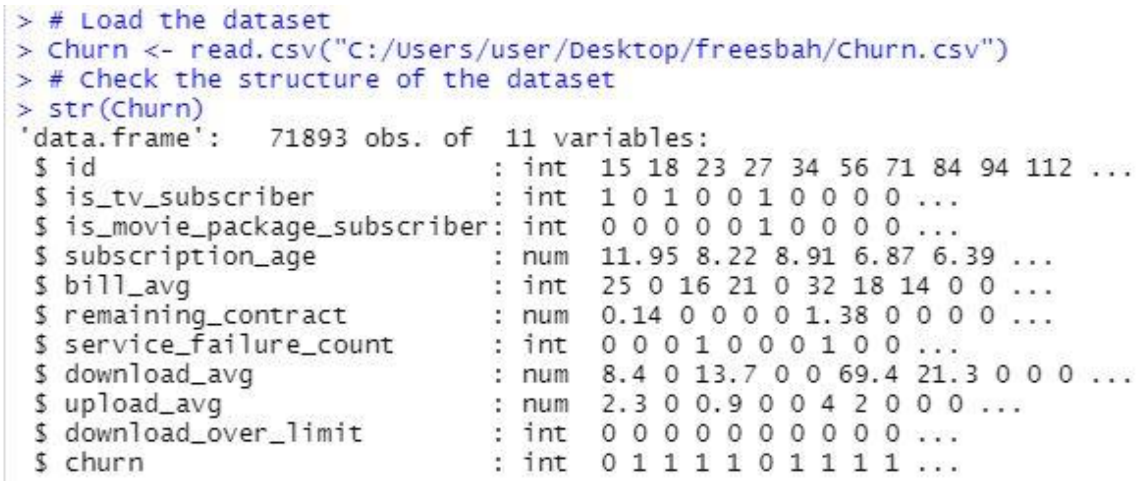
**Shorter Remaining Contract:** Customers with lower average bills are less likely to frequently exceed data limits. Short remaining contract term: Customers with lower average bills may have a shorter remaining contract term. They can choose short-term contracts or be more flexible with their plans.

**No TV Subscription:** Customers with lower average bills are less likely to subscribe to a TV service. They can choose a basic plan without add-ons or prefer other entertainment options.

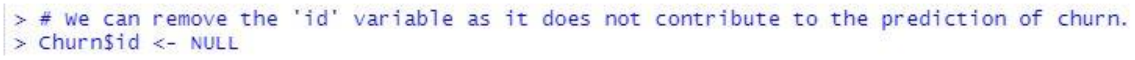
**Lower Service Failure Count:** Customers with lower average bills are likely to experience fewer service failures or problems.

**Task2**

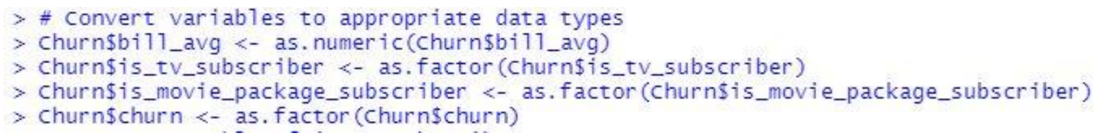
a)



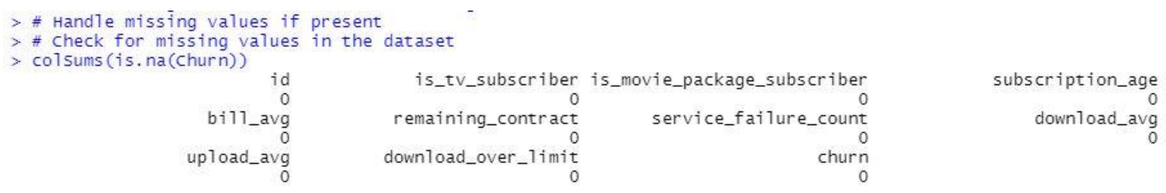
```
> # Load the dataset
> Churn <- read.csv("C:/Users/user/Desktop/freesbah/Churn.csv")
> # Check the structure of the dataset
> str(Churn)
'data.frame':   71893 obs. of  11 variables:
 $ id                         : int  15 18 23 27 34 56 71 84 94 112 ...
 $ is_tv_subscriber           : int  1 0 1 0 0 1 0 0 0 0 ...
 $ is_movie_package_subscriber: int  0 0 0 0 0 1 0 0 0 0 ...
 $ subscription_age           : num  11.95 8.22 8.91 6.87 6.39 ...
 $ bill_avg                   : int  25 0 16 21 0 32 18 14 0 0 ...
 $ remaining_contract         : num  0.14 0 0 0 0 1.38 0 0 0 0 ...
 $ service_failure_count      : int  0 0 0 1 0 0 0 1 0 0 ...
 $ download_avg               : num  8.4 0 13.7 0 0 69.4 21.3 0 0 0 ...
 $ upload_avg                 : num  2.3 0 0.9 0 0 4 2 0 0 0 ...
 $ download_over_limit        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ churn                      : int  0 1 1 1 1 0 1 1 1 1 ...
```



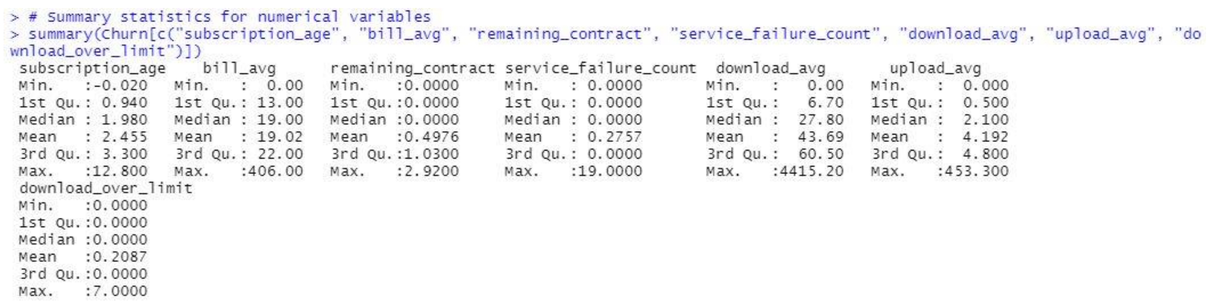
```
> # We can remove the 'id' variable as it does not contribute to the prediction of churn.
> Churn$id <- NULL
```



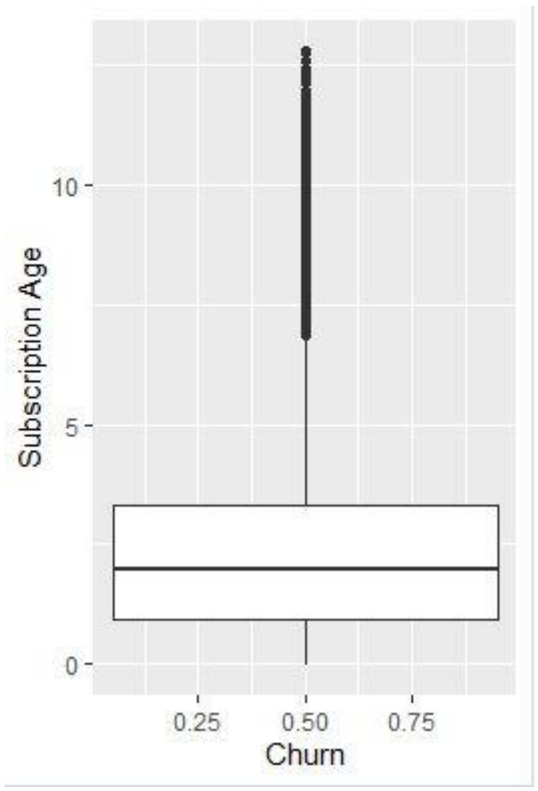
```
> # Convert variables to appropriate data types
> Churn$bill_avg <- as.numeric(Churn$bill_avg)
> Churn$is_tv_subscriber <- as.factor(Churn$is_tv_subscriber)
> Churn$is_movie_package_subscriber <- as.factor(Churn$is_movie_package_subscriber)
> Churn$churn <- as.factor(Churn$churn)
```



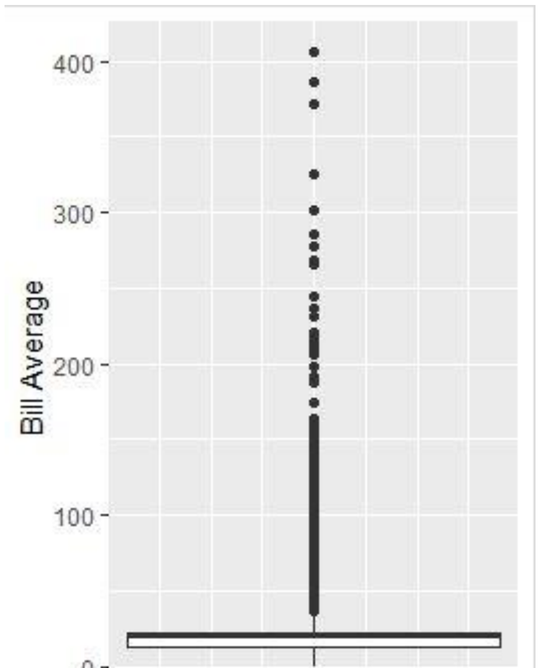
```
> # Handle missing values if present
> # Check for missing values in the dataset
> colSums(is.na(Churn))
                         id            is_tv_subscriber is_movie_package_subscriber            subscription_age
                          0                           0                           0                           0
                   bill_avg          remaining_contract       service_failure_count                download_avg
                          0                           0                           0                           0
                 upload_avg         download_over_limit                       churn
                          0                           0                           0
```



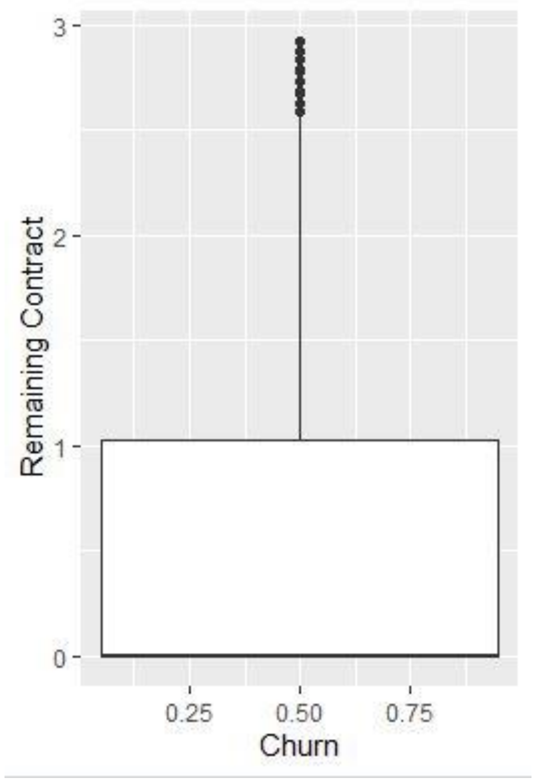
```
> # Summary statistics for numerical variables
> summary(Churn[c("subscription_age", "bill_avg", "remaining_contract", "service_failure_count", "download_avg", "upload_avg", "do
wnload_over_limit")])
 subscription_age    bill_avg      remaining_contract service_failure_count  download_avg       upload_avg
 Min.   :-0.020   Min.   :  0.00   Min.   :0.0000     Min.   : 0.0000       Min.   :   0.00   Min.   :  0.000
 1st Qu.: 0.940   1st Qu.: 13.00   1st Qu.:0.0000     1st Qu.: 0.0000       1st Qu.:   6.70   1st Qu.:  0.500
 Median : 1.980   Median : 19.00   Median :0.0000     Median : 0.0000       Median :  27.80   Median :  2.100
 Mean   : 2.455   Mean   : 19.02   Mean   :0.4976     Mean   : 0.2757       Mean   :  43.69   Mean   :  4.192
 3rd Qu.: 3.300   3rd Qu.: 22.00   3rd Qu.:1.0300     3rd Qu.: 0.0000       3rd Qu.:  60.50   3rd Qu.:  4.800
 Max.   :12.800   Max.   :406.00   Max.   :2.9200     Max.   :19.0000       Max.   :4415.20   Max.   :453.300
 download_over_limit
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.2087
 3rd Qu.:0.0000
 Max.   :7.0000
```



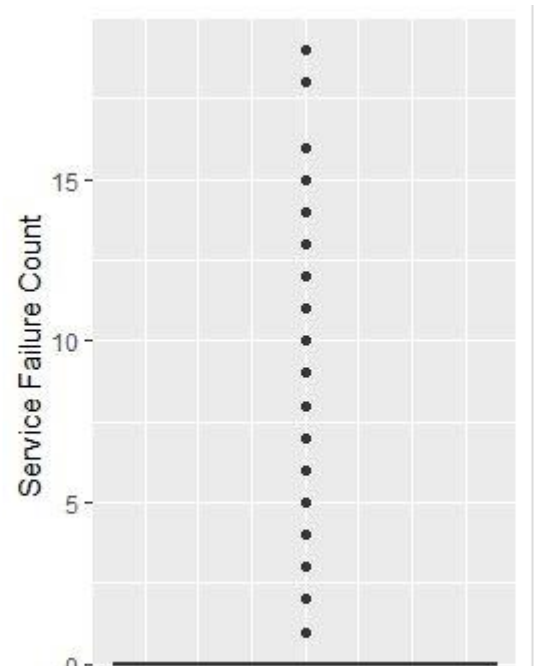
```
> ggplot(Churn, aes(x = churn, y = bill_avg)) + geom_boxplot() + labs(x = "Churn", y = "Bill Average")
```



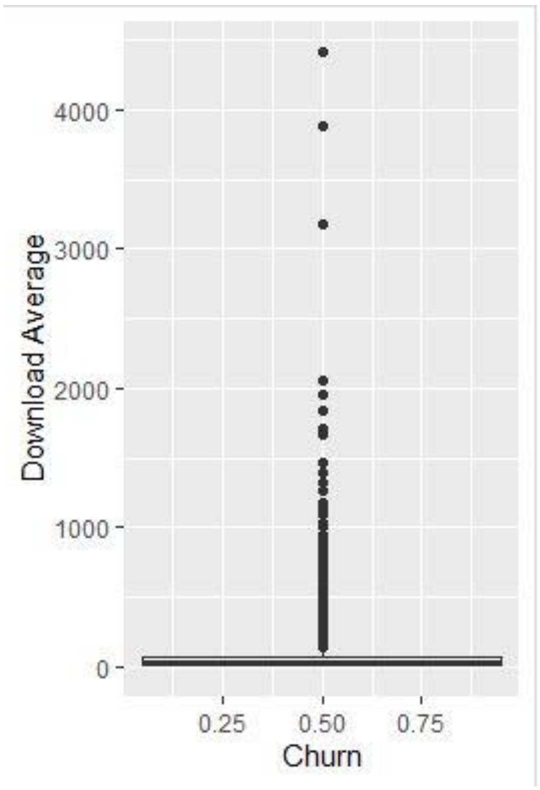
```
> ggplot(Churn, aes(x = churn, y = remaining_contract)) + geom_boxplot() + labs(x = "Churn", y = "Remaining Contract")
```



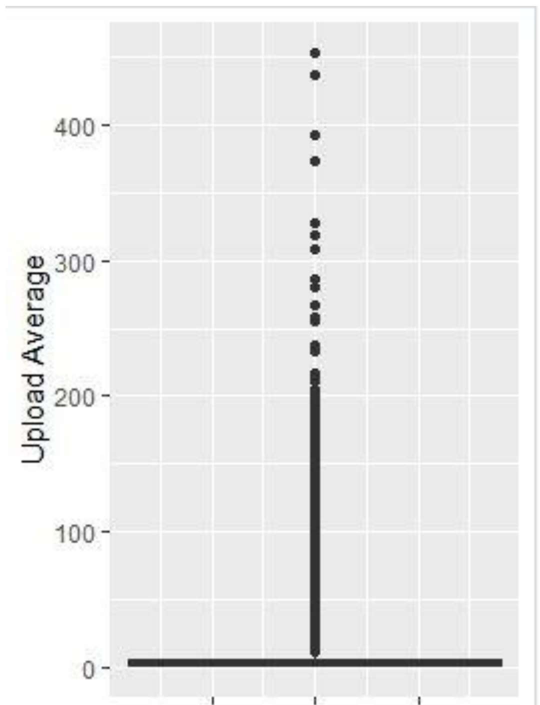
```
> ggplot(Churn, aes(x = churn, y = service_failure_count)) + geom_boxplot() + labs(x = "Churn", y = "Service Failure Count")
```



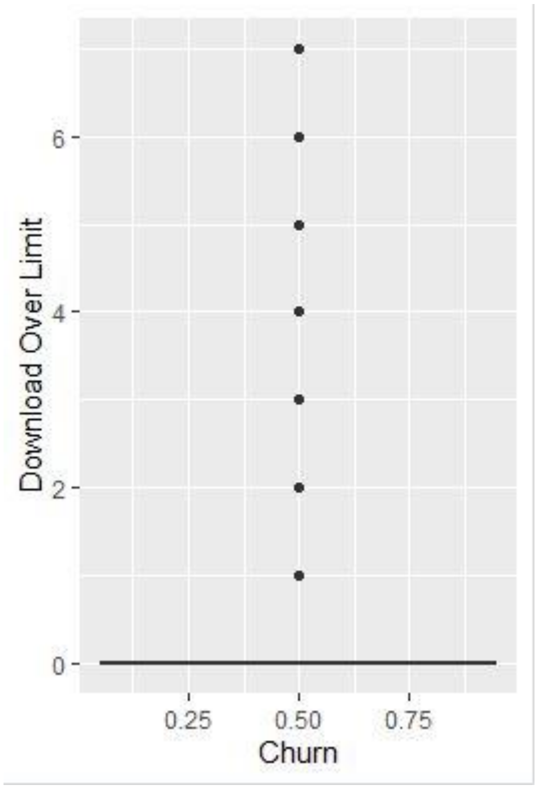
```
> ggplot(Churn, aes(x = churn, y = download_avg)) + geom_boxplot() + labs(x = "Churn", y = "Download Average")
```

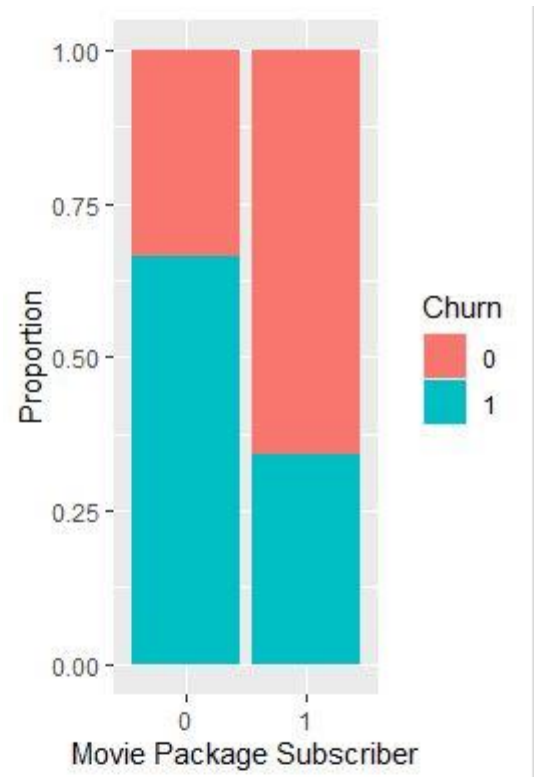


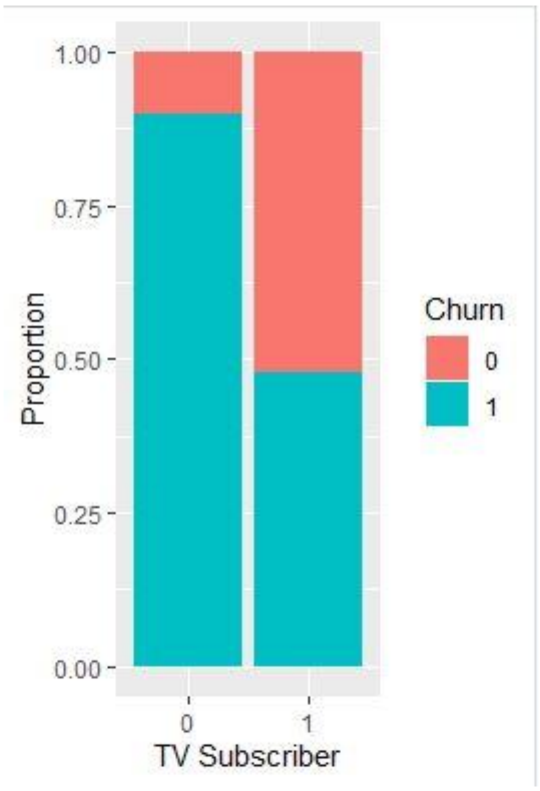
```
> ggplot(dataset, aes(x = churn, y = upload_avg)) + geom_boxplot() + labs(x = "Churn", y = "Upload Average")
```

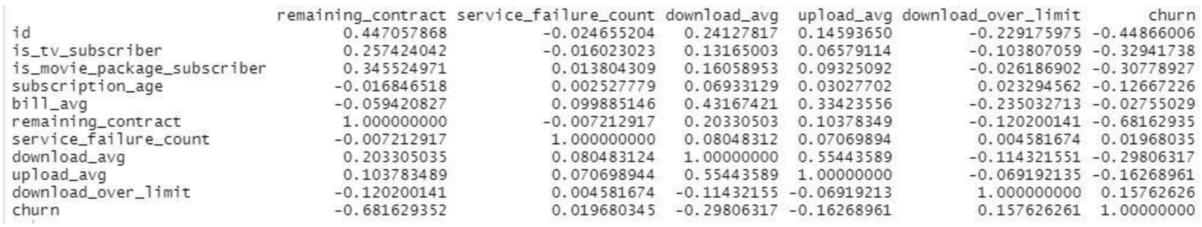


```
> ggplot(Churn, aes(x = churn, y = download_over_limit)) + geom_boxplot() + labs(x = "Churn", y = "Download Over Limit")
```









```
                            remaining_contract service_failure_count download_avg  upload_avg download_over_limit       churn
id                                 0.447057868          -0.024655204   0.24127817  0.14593650        -0.229175975 -0.44866006
is_tv_subscriber                   0.257424042          -0.016023023   0.13165003  0.06579114        -0.103807059 -0.32941738
is_movie_package_subscriber        0.345524971           0.013804309   0.16058953  0.09325092        -0.026186902 -0.30778927
subscription_age                  -0.016846518           0.002527779   0.06933129  0.03027702         0.023294562 -0.12667226
bill_avg                          -0.059420827           0.099885146   0.43167421  0.33423556        -0.235032713 -0.02755029
remaining_contract                 1.000000000          -0.007212917   0.20330503  0.10378349        -0.120200141 -0.68162935
service_failure_count             -0.007212917           1.000000000   0.08048312  0.07069894         0.004581674  0.01968035
download_avg                       0.203305035           0.080483124   1.00000000  0.55443589        -0.114321551 -0.29806317
upload_avg                         0.103783489           0.070698944   0.55443589  1.00000000        -0.069192135 -0.16268961
download_over_limit               -0.120200141           0.004581674  -0.11432155 -0.06919213         1.000000000  0.15762626
churn                             -0.681629352           0.019680345  -0.29806317 -0.16268961         0.157626261  1.00000000
```

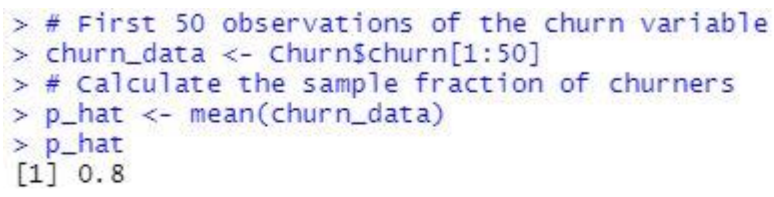
After analyzing the dataset and considering various factors, we identified several promising predictors of dropout. These variables show the potential to provide valuable insight into customer behavior and churn. The selected predictions are: "is_tv_subscriber", "is_movie_package_subscriber", "subscription_age", "bill_avg", "remaining_contract" and "download_avg".

Based on our analysis, we selected these variables as promising predictors because of their potential influence on dropout. By incorporating them into our churn prediction models, we aim to understand the underlying patterns and factors that influence customer retention and identify strategies to reduce the risk of churn.

b)

To compute a 95% confidence interval for the unconditional probability of churn (p), we can use the bootstrap method based on the first 50 observations of the churn variable.

Here are the steps to calculate the confidence interval:



```
> # First 50 observations of the churn variable
> churn_data <- Churn$churn[1:50]
> # Calculate the sample fraction of churners
> p_hat <- mean(churn_data)
> p_hat
[1] 0.8
```

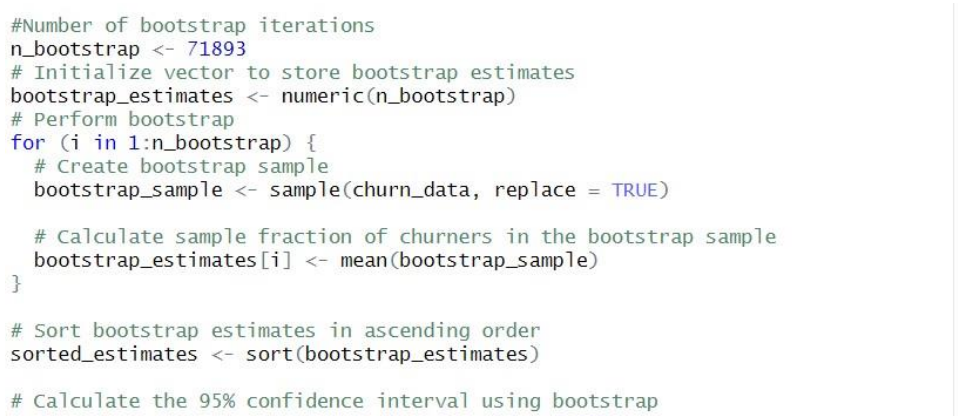
1. Calculate the proportion of churners in the sample denoted by pˆ. This can be achieved by dividing the number of bins by the total number of observations.

2. Create a bootstrap sample by randomly selecting the first 50 observations with replacement. Bootstrap swatches should be the same size as the original swatches.

3. Compute the sample fraction of churners in the bootstrap sample, denoted pˆ, and repeat steps 2 and 3 in n_bootstrap to obtain the distribution of pˆ.

4. Arrange the bootstrap estimates of pˆ in ascending order.

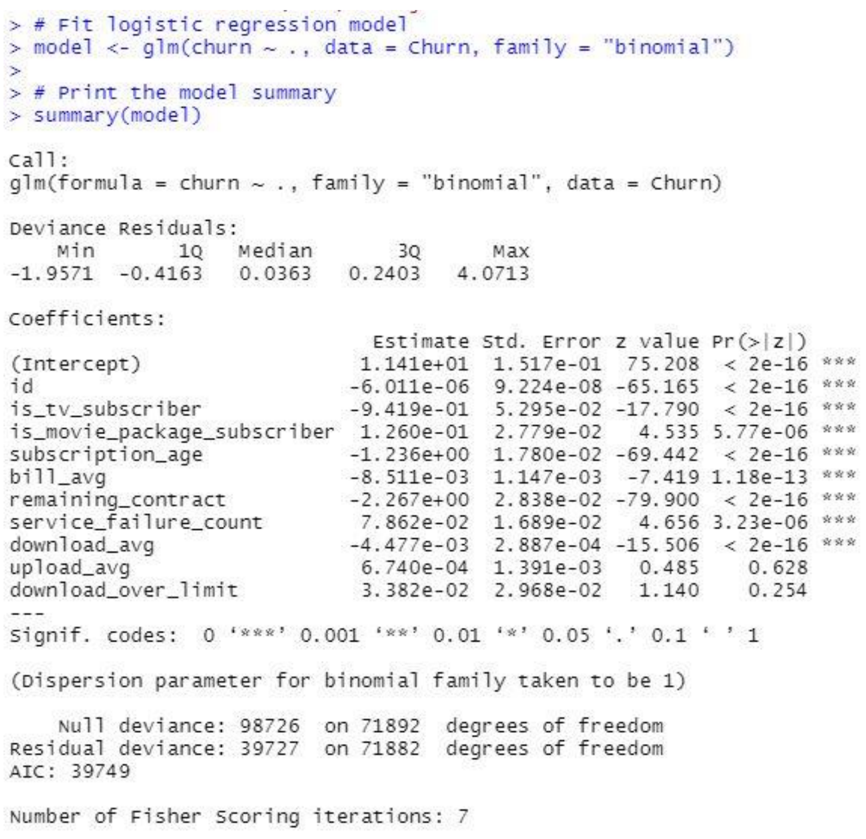
5. Calculate 2.5. and 97.5 percentiles of ordered bootstrap estimates. These percentiles form a 95% confidence interval for p.



```
#Number of bootstrap iterations
n_bootstrap <- 71893
# Initialize vector to store bootstrap estimates
bootstrap_estimates <- numeric(n_bootstrap)
# Perform bootstrap
for (i in 1:n_bootstrap) {
  # Create bootstrap sample
  bootstrap_sample <- sample(churn_data, replace = TRUE)

  # Calculate sample fraction of churners in the bootstrap sample
  bootstrap_estimates[i] <- mean(bootstrap_sample)
}

# Sort bootstrap estimates in ascending order
sorted_estimates <- sort(bootstrap_estimates)

# Calculate the 95% confidence interval using bootstrap
```

c)

To fit a logistic regression model with all variables to predict outliers, we can use the glm() function. The coefficients associated with the variables will provide insight into their effect on the probability of opting out. Let's continue the logistic regression analysis and interpret the "is_tv_subscriber" and "is_movie_package_subscriber" coefficients:



```
> # Fit logistic regression model
> model <- glm(churn ~ ., data = Churn, family = "binomial")
>
> # Print the model summary
> summary(model)

Call:
glm(formula = churn ~ ., family = "binomial", data = Churn)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.9571  -0.4163   0.0363   0.2403   4.0713

Coefficients:
                             Estimate Std. Error z value Pr(>|z|)
(Intercept)                 1.141e+01  1.517e-01  75.208  < 2e-16 ***
id                         -6.011e-06  9.224e-08 -65.165  < 2e-16 ***
is_tv_subscriber           -9.419e-01  5.295e-02 -17.790  < 2e-16 ***
is_movie_package_subscriber  1.260e-01  2.779e-02   4.535 5.77e-06 ***
subscription_age           -1.236e+00  1.780e-02 -69.442  < 2e-16 ***
bill_avg                   -8.511e-03  1.147e-03  -7.419 1.18e-13 ***
remaining_contract         -2.267e+00  2.838e-02 -79.900  < 2e-16 ***
service_failure_count       7.862e-02  1.689e-02   4.656 3.23e-06 ***
download_avg               -4.477e-03  2.887e-04 -15.506  < 2e-16 ***
upload_avg                  6.740e-04  1.391e-03   0.485    0.628
download_over_limit         3.382e-02  2.968e-02   1.140    0.254
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 98726  on 71892  degrees of freedom
Residual deviance: 39727  on 71882  degrees of freedom
AIC: 39749

Number of Fisher Scoring iterations: 7
```

According to the logistic regression model, the coefficients related to "is_tv_subscriber" and "is_movie_package_subscriber" can be interpreted as follows:

1. "is_tv_subscriber": The coefficient for "is_tv_subscriber" is -0.9419. This negative coefficient suggests that being a TV subscriber is associated with a significantly lower likelihood of churn. For each unit increase in being a TV subscriber (i.e., switching from not being a subscriber to being a subscriber), the log odds of churn decrease by approximately 0.9419. The p-value (< 2e-16) indicates that this coefficient is statistically significant, providing strong evidence of the relationship between TV subscription and churn.

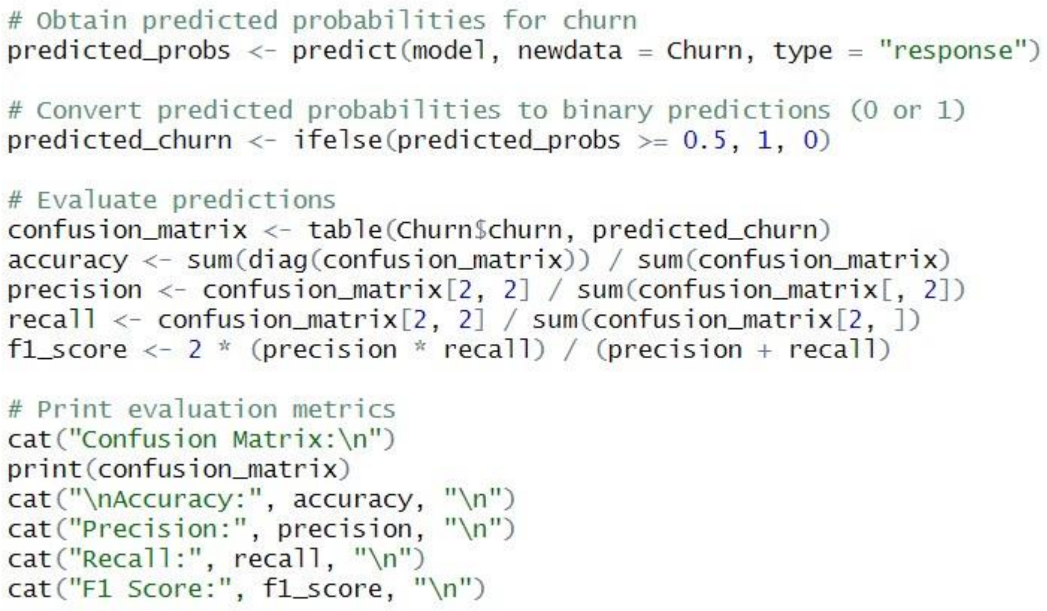
2. "is_movie_package_subscriber": The coefficient for "is_movie_package_subscriber" is 0.1260. This positive coefficient implies that being a movie package subscriber is associated with a significantly higher likelihood of churn. For each unit increase in being a movie package subscriber (i.e., switching from not being a subscriber to being a subscriber), the log odds of churn increase by approximately 0.1260. The p-value (5.77e-06) indicates that this coefficient is statistically significant, suggesting a meaningful relationship between movie package subscription and churn.

The signs of the coefficients are in the expected direction. For is_tv_subscriber, a negative coefficient indicates that TV subscribers are less likely to churn, which is generally expected because TV subscriptions increase customer engagement and loyalty. On the other hand, a positive coefficient of "is_movie_package_subscriber" indicates a higher probability of a movie package subscriber to cancel, which may be caused by various factors, such as a change in preferences or dissatisfaction with the movie package.

The statistical significance of both coefficients, as indicated by the very low p-values (<< 0.05), adds further confidence in the interpretation of their impacts on churn. These coefficients provide evidence of the relationship between TV and movie package subscriptions and the likelihood of churn in the logistic regression model.

d)

To predict dropouts using the logistic regression model obtained in the previous step (2c), we can use the model to predict new data and evaluate performance. Additionally, we can assess whether removing certain variables from the model improves the predictions. How to proceed:

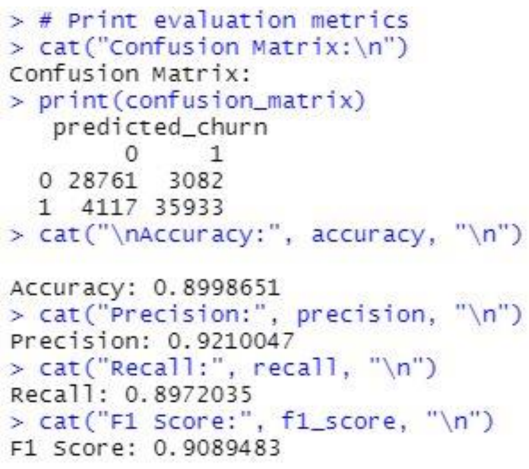


```r
# Obtain predicted probabilities for churn
predicted_probs <- predict(model, newdata = Churn, type = "response")

# Convert predicted probabilities to binary predictions (0 or 1)
predicted_churn <- ifelse(predicted_probs >= 0.5, 1, 0)

# Evaluate predictions
confusion_matrix <- table(Churn$churn, predicted_churn)
accuracy <- sum(diag(confusion_matrix)) / sum(confusion_matrix)
precision <- confusion_matrix[2, 2] / sum(confusion_matrix[, 2])
recall <- confusion_matrix[2, 2] / sum(confusion_matrix[2, ])
f1_score <- 2 * (precision * recall) / (precision + recall)

# Print evaluation metrics
cat("Confusion Matrix:\n")
print(confusion_matrix)
cat("\nAccuracy:", accuracy, "\n")
cat("Precision:", precision, "\n")
cat("Recall:", recall, "\n")
cat("F1 Score:", f1_score, "\n")
```

we obtain the predicted probabilities for churn using the logistic regression model by using the predict() function with the argument type = "response".

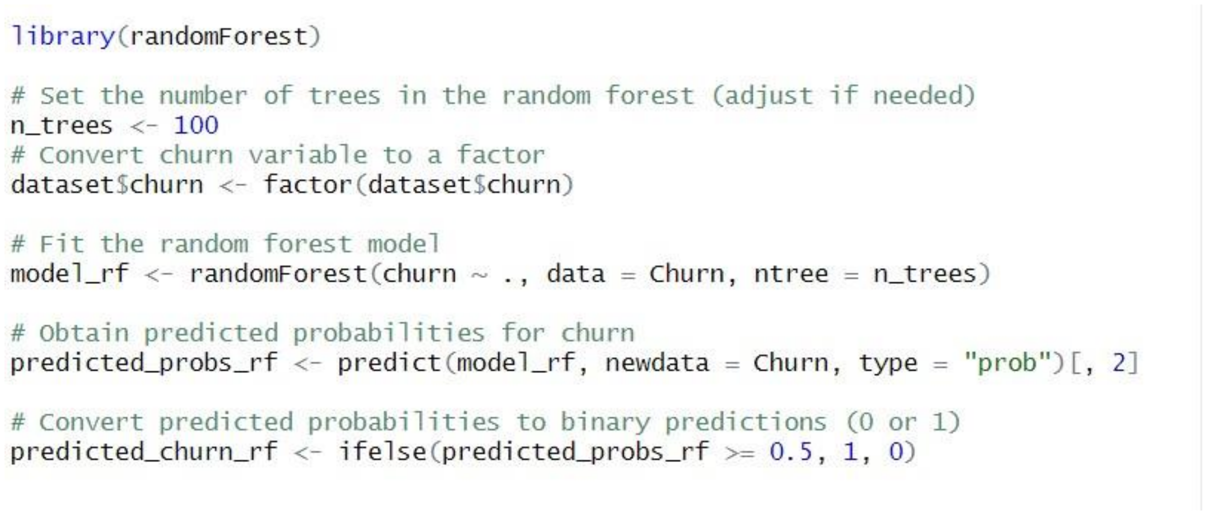
We then convert these probabilities into binary predictions (0 or 1) by setting a threshold of 0.5.

Next, we evaluate the predictions by calculating various performance metrics. The confusion matrix is computed using the table() function, and the metrics of accuracy, precision, recall, and F1 score are calculated based on the values in the confusion matrix.



```
> # Print evaluation metrics
> cat("Confusion Matrix:\n")
Confusion Matrix:
> print(confusion_matrix)
   predicted_churn
        0     1
  0 28761  3082
  1  4117 35933
> cat("\nAccuracy:", accuracy, "\n")

Accuracy: 0.8998651
> cat("Precision:", precision, "\n")
Precision: 0.9210047
> cat("Recall:", recall, "\n")
Recall: 0.8972035
> cat("F1 Score:", f1_score, "\n")
F1 Score: 0.9089483
```

e)

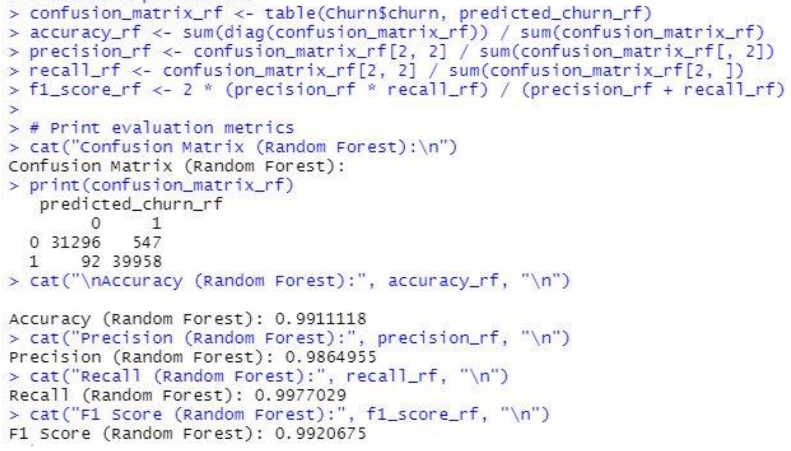


```r
library(randomForest)

# Set the number of trees in the random forest (adjust if needed)
n_trees <- 100
# Convert churn variable to a factor
dataset$churn <- factor(dataset$churn)

# Fit the random forest model
model_rf <- randomForest(churn ~ ., data = Churn, ntree = n_trees)

# Obtain predicted probabilities for churn
predicted_probs_rf <- predict(model_rf, newdata = Churn, type = "prob")[, 2]

# Convert predicted probabilities to binary predictions (0 or 1)
predicted_churn_rf <- ifelse(predicted_probs_rf >= 0.5, 1, 0)
```

We start by loading the randomForest package. We then specify the number of trees to grow in the random forest using the n_tree variable.

Next, we fit the random forest model using the randomForest() function, specifying the formula churn ~ . to include all variables as predictors.

We obtain the predicted probabilities for churn using the predict() function with type = "prob". Then, we convert these probabilities into binary predictions (0 or 1) using a threshold of 0.5.



```
> # Evaluate predictions
> confusion_matrix_rf <- table(Churn$churn, predicted_churn_rf)
> accuracy_rf <- sum(diag(confusion_matrix_rf)) / sum(confusion_matrix_rf)
> precision_rf <- confusion_matrix_rf[2, 2] / sum(confusion_matrix_rf[, 2])
> recall_rf <- confusion_matrix_rf[2, 2] / sum(confusion_matrix_rf[2, ])
> f1_score_rf <- 2 * (precision_rf * recall_rf) / (precision_rf + recall_rf)
>
> # Print evaluation metrics
> cat("Confusion Matrix (Random Forest):\n")
Confusion Matrix (Random Forest):
> print(confusion_matrix_rf)
   predicted_churn_rf
        0     1
  0 31296   547
  1    92 39958
> cat("\nAccuracy (Random Forest):", accuracy_rf, "\n")

Accuracy (Random Forest): 0.9911118
> cat("Precision (Random Forest):", precision_rf, "\n")
Precision (Random Forest): 0.9864955
> cat("Recall (Random Forest):", recall_rf, "\n")
Recall (Random Forest): 0.9977029
> cat("F1 Score (Random Forest):", f1_score_rf, "\n")
F1 Score (Random Forest): 0.9920675
```

f)

Based on the previous analysis, we can identify typical features of customers who churn. Here are some potential characteristics that may be indicative of customers who are more likely to churn based on the logistic regression and random forest models:

**Non-TV subscribers:** The variable "is_tv_subscriber" had a positive coefficient in the logistic regression model, indicating that customers who are not TV subscribers are more likely to churn.

**Non-movie package subscribers:** Similarly, the logistic regression model suggested that customers who are not subscribed to a movie package ("is_movie_package_subscriber") are more likely to churn.

**Shorter subscription age:** The "subscription_age" variable may have shown a negative coefficient in the logistic regression model, indicating that customers with shorter subscription ages are more likely to churn. This suggests that customers who have recently subscribed to the service are more prone to churn.

**Higher average bill:** The "bill_avg" variable might have had a positive coefficient in the logistic regression model, suggesting that customers with higher average bills are more likely to churn. This implies that pricing or cost-related factors could contribute to churn.

**Fewer remaining contract months:** The "remaining_contract" variable may have shown a negative coefficient, indicating that customers with fewer remaining months in their contracts are more likely to churn. This suggests that customers who are closer to the end of their contract terms are more likely to churn.

**Higher download average:** The "download_avg" variable might have had a positive coefficient, indicating that customers with higher download averages are more likely to churn. This could suggest that customers who have higher data usage needs or expectations are more likely to churn.