Practicum II

Dauby, Ray

11-06-24

Problem 1: Classification with Naive Bayes, Decision Trees, and Logistic Regression

Question 1: Load Data

```
# File path of each provided data file
adult_data_path <- "adult/adult.data"
adult_test_path <- "adult/adult.test"
adult_names_path <- "adult/adult.names"

# Load the data
adult.data <- read.csv(adult_data_path, sep = ",", header = F, stringsAsFactors = T)
adult.test <- read.csv(adult_test_path, sep = ",", header = F, stringsAsFactors = T)
adult.names <- read.csv(adult_names_path, sep = ",", header = F, stringsAsFactors = T)
```

Question 2: Combine Data

```
# List of column names
column_names <- c('age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status',
                  'occupation', 'relationship', 'race', 'sex', 'capital_gain', 'capital_loss',
                  'hours_per_week', 'native_country', 'income')

# Merge dfs vertically
combined_df <- rbind(adult.data, adult.test)

# Assign column names
colnames(combined_df) <- column_names

# Format income column
combined_df$income <- gsub("\\.", "", combined_df$income)
combined_df <- combined_df[combined_df$income != "", ]
combined_df$income <- factor(combined_df$income)
combined_df <- na.omit(combined_df)

table(combined_df$income)
```

```
##
##  <=50K   >50K
##  37155  11687
```

```r
# Show required rows and columns
head(combined_df[c(11, 112, 199, 203), 1:4])
```

```
##     age workclass fnlwgt     education
## 11   37   Private 280464  Some-college
## 112  38   Private  65324   Prof-school
## 199  35   Private 138992       Masters
## 203  51   Private 259323     Bachelors
```

Question 3: Split Data

```r
# Set the seed for reproducibility
set.seed(33452)

# Split the data into 75% training and 25% validation
partition <- createDataPartition(combined_df$income, p = 0.75, list = FALSE)
```

Question 4: Naive Bayes Model

```r
# Subset df and remove rows with NAs
subset_combined_df <- na.omit(combined_df[, c("age", "workclass", "education_num", "race", "sex",
                                              "hours_per_week", "native_country", "income")])

# Remove rows with any empty strings in any factor columns
subset_combined_df <- subset_combined_df[!apply(subset_combined_df == "", 1, any), ]
subset_combined_df <- droplevels(subset_combined_df)

# Binning function (using fixed bin breaks)
# Asked ChatGPT for help standardizing the bin size
binning_function <- function(x, bin_edges) {
  cut(x, breaks = bin_edges, include.lowest = TRUE, labels = FALSE)
}

# Number of bins =~ sqrt(nrow(subset_combined_df))
num_bins <- 200

# Calculate bin edges for 3 continuous variables
age_breaks <- seq(min(subset_combined_df$age, na.rm = TRUE), max(subset_combined_df$age, na.rm = TRUE),
                  length.out = num_bins + 1)

education_num_breaks <- seq(min(subset_combined_df$education_num, na.rm = TRUE),
                            max(subset_combined_df$education_num, na.rm = TRUE),
                            length.out = num_bins + 1)

hours_per_week_breaks <- seq(min(subset_combined_df$hours_per_week, na.rm = TRUE),
                             max(subset_combined_df$hours_per_week, na.rm = TRUE),
                             length.out = num_bins + 1)

# Apply the binning to the entire dataset
```

```r
subset_combined_df$age <- binning_function(as.numeric(subset_combined_df$age), age_breaks)
subset_combined_df$education_num <- binning_function(as.numeric(subset_combined_df$education_num),
                                                     education_num_breaks)
subset_combined_df$hours_per_week <- binning_function(as.numeric(subset_combined_df$hours_per_week),
                                                      hours_per_week_breaks)

# Create the training set (75%) & validation set (25%)
train_data <- subset_combined_df[partition, ]
validation_data <- subset_combined_df[-partition, ]

# Train Naive Bayes model
nb_model <- naiveBayes(income ~ age + workclass + education_num + race + sex + hours_per_week +
                         native_country, data = train_data)
```

Question 5: Naive Bayes Confusion Matrix

```r
# Make Predictions on validation data
predictions <- predict(nb_model, validation_data)

# Confusion matrix
bayes_matrix <- confusionMatrix(predictions, validation_data$income)
print(bayes_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   8517  1615
##      >50K     771  1306
##
##                Accuracy : 0.8046
##                  95% CI : (0.7974, 0.8116)
##     No Information Rate : 0.7608
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4041
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9170
##             Specificity : 0.4471
##          Pos Pred Value : 0.8406
##          Neg Pred Value : 0.6288
##              Prevalence : 0.7608
##          Detection Rate : 0.6976
##    Detection Prevalence : 0.8299
##       Balanced Accuracy : 0.6820
##
##        'Positive' Class :  <=50K
##
```

The Naive Bayes model is performing quite well in terms of sensitivity (TPR = 91.7%), meaning it is highly effective at identifying the <=50K class. However, it struggles with specificity (TNR = 44.71%), meaning it has difficulty correctly identifying the minority class. The high prevalence of the <=50K class (76.08%) likely contributes to the model's bias towards predicting this class. While the model does a good job of predicting <=50K (with a high PPV of 84.06%), the model still misclassifies a non-negligible number of >50K instances as <=50K, contributing to the low specificity.

Question 6: Naive Bayes Model Statistics

```
# Extract confusion matrix values
bm <- bayes_matrix$table
TP <- bm[2, 2]
FP <- bm[1, 2]
TN <- bm[1, 1]
FN <- bm[2, 1]

# Manually Ccalculate accuracy
accuracy <- (TP + TN) / (TP + TN + FP + FN)

# Manually calculate precision
precision <- TP / (TP + FP)

# Print the metrics
cat("Overall Accuracy: ", accuracy, "\n")
```

```
## Overall Accuracy:  0.8045704
```

```
cat("Precision: ", precision, "\n")
```

```
## Precision:  0.4471072
```

Question 7: Logistic Model

```
# Fit the logistic regression model
logistic_model <- glm(income ~ age + workclass + education_num + race + sex + hours_per_week +
                        native_country,
                      data = train_data,
                      family = binomial)

# Print the summary
summary(logistic_model)
```

```
##
## Call:
## glm(formula = income ~ age + workclass + education_num + race +
##     sex + hours_per_week + native_country, family = binomial,
##     data = train_data)
##
## Coefficients:
```

```
##                                          Estimate Std. Error z value
## (Intercept)                             -9.268e+00  2.399e-01 -38.627
## age                                      1.671e-02  4.253e-04  39.279
## workclass Federal-gov                    1.312e+00  1.112e-01  11.806
## workclass Local-gov                      8.441e-01  9.954e-02   8.481
## workclass Never-worked                  -9.201e+00  1.658e+02  -0.055
## workclass Private                        8.932e-01  8.743e-02  10.217
## workclass Self-emp-inc                   1.470e+00  1.083e-01  13.577
## workclass Self-emp-not-inc               4.342e-01  9.784e-02   4.438
## workclass State-gov                      7.001e-01  1.095e-01   6.396
## workclass Without-pay                   -1.644e-01  8.526e-01  -0.193
## education_num                            2.628e-02  4.890e-04  53.735
## race Asian-Pac-Islander                  2.752e-01  2.094e-01   1.314
## race Black                              -6.910e-02  1.796e-01  -0.385
## race Other                               6.665e-02  2.727e-01   0.244
## race White                               3.774e-01  1.710e-01   2.207
## sex Male                                 1.143e+00  3.602e-02  31.724
## hours_per_week                           1.586e-02  6.103e-04  25.983
## native_country Cambodia                  7.309e-01  5.541e-01   1.319
## native_country Canada                    6.136e-01  2.354e-01   2.607
## native_country China                     2.443e-02  3.128e-01   0.078
## native_country Columbia                 -1.632e+00  6.073e-01  -2.687
## native_country Cuba                      4.314e-02  3.085e-01   0.140
## native_country Dominican-Republic       -5.526e-01  5.620e-01  -0.983
## native_country Ecuador                  -6.193e-01  6.096e-01  -1.016
## native_country El-Salvador              -1.235e-01  4.226e-01  -0.292
## native_country England                   4.157e-01  2.791e-01   1.490
## native_country France                    5.426e-01  4.438e-01   1.222
## native_country Germany                   4.129e-01  2.301e-01   1.794
## native_country Greece                    3.550e-01  4.287e-01   0.828
## native_country Guatemala                -3.522e-01  6.234e-01  -0.565
## native_country Haiti                    -5.527e-01  6.720e-01  -0.822
## native_country Holand-Netherlands       -1.069e+01  5.354e+02  -0.020
## native_country Honduras                 -1.464e-02  8.420e-01  -0.017
## native_country Hong                      9.272e-01  5.788e-01   1.602
## native_country Hungary                   1.199e-01  7.187e-01   0.167
## native_country India                     3.167e-01  2.593e-01   1.221
## native_country Iran                      3.751e-01  3.721e-01   1.008
## native_country Ireland                   1.030e+00  4.669e-01   2.206
## native_country Italy                     9.224e-01  3.024e-01   3.050
## native_country Jamaica                   1.791e-01  4.061e-01   0.441
## native_country Japan                     5.820e-01  3.168e-01   1.837
## native_country Laos                     -4.100e-01  8.619e-01  -0.476
## native_country Mexico                   -5.399e-01  2.226e-01  -2.426
## native_country Nicaragua                -3.516e-01  6.502e-01  -0.541
## native_country Outlying-US(Guam-USVI-etc) -1.193e+01  1.339e+02  -0.089
## native_country Peru                     -6.635e-01  6.497e-01  -1.021
## native_country Philippines               5.315e-01  2.229e-01   2.385
## native_country Poland                   -2.226e-02  3.476e-01  -0.064
## native_country Portugal                  8.748e-01  4.198e-01   2.084
## native_country Puerto-Rico               4.253e-02  3.088e-01   0.138
## native_country Scotland                 -8.319e-01  8.368e-01  -0.994
## native_country South                    -5.867e-01  3.789e-01  -1.549
## native_country Taiwan                    1.218e-01  3.686e-01   0.330
```

```
## native_country Thailand                  -5.281e-01  6.925e-01  -0.763
## native_country Trinadad&Tobago            -5.743e-01  1.111e+00  -0.517
## native_country United-States               2.894e-01  1.104e-01   2.621
## native_country Vietnam                    -7.789e-01  5.106e-01  -1.526
## native_country Yugoslavia                  8.841e-01  5.883e-01   1.503
##                                          Pr(>|z|)
## (Intercept)                               < 2e-16 ***
## age                                       < 2e-16 ***
## workclass Federal-gov                     < 2e-16 ***
## workclass Local-gov                       < 2e-16 ***
## workclass Never-worked                    0.95575
## workclass Private                         < 2e-16 ***
## workclass Self-emp-inc                    < 2e-16 ***
## workclass Self-emp-not-inc               9.07e-06 ***
## workclass State-gov                      1.60e-10 ***
## workclass Without-pay                     0.84714
## education_num                             < 2e-16 ***
## race Asian-Pac-Islander                   0.18871
## race Black                                0.70044
## race Other                                0.80693
## race White                                0.02734 *
## sex Male                                  < 2e-16 ***
## hours_per_week                            < 2e-16 ***
## native_country Cambodia                   0.18715
## native_country Canada                     0.00913 **
## native_country China                      0.93774
## native_country Columbia                   0.00721 **
## native_country Cuba                       0.88880
## native_country Dominican-Republic         0.32550
## native_country Ecuador                    0.30965
## native_country El-Salvador                0.77015
## native_country England                    0.13633
## native_country France                     0.22152
## native_country Germany                    0.07276 .
## native_country Greece                     0.40763
## native_country Guatemala                  0.57215
## native_country Haiti                      0.41082
## native_country Holand-Netherlands         0.98407
## native_country Honduras                   0.98613
## native_country Hong                       0.10916
## native_country Hungary                    0.86751
## native_country India                      0.22193
## native_country Iran                       0.31341
## native_country Ireland                    0.02737 *
## native_country Italy                      0.00229 **
## native_country Jamaica                    0.65913
## native_country Japan                      0.06617 .
## native_country Laos                       0.63425
## native_country Mexico                     0.01528 *
## native_country Nicaragua                  0.58869
## native_country Outlying-US(Guam-USVI-etc) 0.92902
## native_country Peru                       0.30718
## native_country Philippines                0.01709 *
## native_country Poland                     0.94894
```

```
## native_country Portugal                    0.03717 *
## native_country Puerto-Rico                 0.89044
## native_country Scotland                    0.32013
## native_country South                       0.12148
## native_country Taiwan                      0.74106
## native_country Thailand                    0.44569
## native_country Trinadad&Tobago             0.60529
## native_country United-States               0.00876 **
## native_country Vietnam                     0.12711
## native_country Yugoslavia                  0.13292
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 40316  on 36632  degrees of freedom
## Residual deviance: 30915  on 36575  degrees of freedom
## AIC: 31031
##
## Number of Fisher Scoring iterations: 12
```

Question 8: Logistic Model Confusion Matrix

```r
# Make predictions on the validation_data
predictions <- predict(logistic_model, validation_data, type = "response")

# Convert predictions to binary outcome
predicted_class <- ifelse(predictions > 0.5, " >50K", " <=50K")
predicted_class <- factor(predicted_class, levels = levels(validation_data$income))

# Create confusion matrix
log_matrix <- confusionMatrix(predicted_class, validation_data$income)
print(log_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   8720  1805
##      >50K     568  1116
##
##                Accuracy : 0.8056
##                  95% CI : (0.7985, 0.8126)
##     No Information Rate : 0.7608
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3754
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9388
##             Specificity : 0.3821
```

```
##          Pos Pred Value : 0.8285
##          Neg Pred Value : 0.6627
##              Prevalence : 0.7608
##          Detection Rate : 0.7142
##    Detection Prevalence : 0.8621
##       Balanced Accuracy : 0.6605
##
##        'Positive' Class :  <=50K
##
```

The logistic regression model has high sensitivity (TPR = 93.88%) and relatively low specificity (TNR = 38.21%), meaning it has trouble correctly identifying the >50K class. The model's performance is again affected by the class imbalance — this is likely why the model tends to predict the majority class more often. The model achieves a good positive predictive value (PPV = 82.85%) for predicting <=50K, but the negative predictive value for >50K (NPV = 66.27%) is lower. The relatively low balanced accuracy (66.05%) shows that while the model performs well for the dominant <=50K class, it is less effective at predicting the minority class (>50K).

The Kappa value for the logistic regression model (0.375396) is slightly lower than the C5.0 model's Kappa (0.4041192), suggesting slightly worse agreement between the predicted and actual classes, though both models have moderate agreement.

Question 9: Decision Tree Model

```r
# Train the C5.0 model excluding the specified columns from the training data
dt_model <- C5.0(income ~ age+workclass+education_num+race+sex+hours_per_week+native_country,
                 data = train_data, rules=F)

# Print the summary of the model to see the tree structure
summary(dt_model)
```

```
##
## Call:
## C5.0.formula(formula = income ~ age + workclass + education_num + race + sex
##  + hours_per_week + native_country, data = train_data, rules = F)
##
##
## C5.0 [Release 2.07 GPL Edition]      Tue Nov 12 20:45:24 2024
## -------------------------------
##
## Class specified by attribute 'outcome'
##
## Read 36633 cases (8 attributes) from undefined.data
##
## Decision tree:
##
## age <= 28: <=50K (9065/299)
## age > 28:
## :...education_num > 147:
##     :...sex = Female:
##     :   :...hours_per_week <= 90: <=50K (1562/455)
##     :   :   hours_per_week > 90:
```

```
##     :   :   :...education_num <= 174: <=50K (398/174)
##     :   :       education_num > 174:
##     :   :       :...hours_per_week <= 141: >50K (59/11)
##     :   :           hours_per_week > 141: <=50K (10/1)
##     :   sex = Male:
##     :   :...age <= 44:
##     :       :...hours_per_week <= 82: <=50K (522/178)
##     :       :   hours_per_week > 82:
##     :       :   :...age <= 31: <=50K (102/39)
##     :       :       age > 31:
##     :       :       :...workclass in {?,Federal-gov,Never-worked,Private,
##     :       :           :             Self-emp-inc,Self-emp-not-inc,
##     :       :           :             Without-pay}: >50K (383.2/155.4)
##     :       :           workclass in {Local-gov,State-gov}: <=50K (54.8/22.1)
##     :       age > 44:
##     :       :...hours_per_week <= 62:
##     :           :...education_num <= 174: <=50K (258/74)
##     :           :   education_num > 174:
##     :           :   :...hours_per_week <= 45: <=50K (41/16)
##     :           :       hours_per_week > 45: >50K (25/7)
##     :           hours_per_week > 62:
##     :           :...workclass in {?,Federal-gov,Never-worked,Private,
##     :               :             Self-emp-inc,
##     :               :             Without-pay}: >50K (3184.3/892.2)
##     :               workclass in {Local-gov,Self-emp-not-inc,State-gov}:
##     :               :...education_num > 160: >50K (604.5/160.4)
##     :                   education_num <= 160:
##     :                   :...race in {Amer-Indian-Eskimo,
##     :                       :        Other}: >50K (4/1)
##     :                       race = Asian-Pac-Islander: <=50K (28/12)
##     :                       race = Black:
##     :                       :...hours_per_week <= 84: <=50K (21.3/6)
##     :                       :   hours_per_week > 84: >50K (10.5/3)
##     :                       race = White:
##     :                       :...age <= 61: <=50K (125.5/48)
##     :                           age > 61: >50K (413.8/183.6)
##     education_num <= 147:
##     :...sex = Female: <=50K (6207/576)
##         sex = Male:
##         :...hours_per_week <= 70: <=50K (1529/170)
##             hours_per_week > 70:
##             :...education_num <= 94: <=50K (1906/208)
##                 education_num > 94:
##                 :...education_num <= 107: <=50K (5574/1499)
##                     education_num > 107:
##                     :...hours_per_week <= 86:
##                         :...age <= 72: <=50K (1530/411)
##                         :   age > 72:
##                         :   :...hours_per_week > 76: <=50K (1021/452)
##                         :       hours_per_week <= 76:
##                         :       :...age <= 102: >50K (29/7)
##                         :           age > 102: <=50K (9/2)
##                         hours_per_week > 86:
##                         :...age <= 53: <=50K (664/192)
```

```
##                                age > 53:
##                                :...workclass in {Federal-gov,
##                                    :             Self-emp-inc}: >50K (168.5/60.2)
##                                    workclass in {?,Never-worked,Self-emp-not-inc,
##                                    :             State-gov,
##                                    :             Without-pay}: <=50K (229.8/73.8)
##                                    workclass = Local-gov:
##                                    :...hours_per_week <= 111: <=50K (41.9/18.3)
##                                    :   hours_per_week > 111: >50K (21.4/6.2)
##                                    workclass = Private:
##                                    :...education_num > 134: >50K (101.6/45.6)
##                                        education_num <= 134:
##                                        :...education_num <= 120:
##                                            :...age <= 58: <=50K (74.6/26.6)
##                                            :   age > 58: >50K (534.6/255.4)
##                                            education_num > 120:
##                                            :...hours_per_week <= 111: >50K (98.9/43.3)
##                                                hours_per_week > 111: <=50K (20.6/6)
##
##
## Evaluation on training data (36633 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##      38 6786(18.5%)   <<
##
##
##     (a)   (b)    <-classified as
##    ----  ----
##   26041  1826    (a): class <=50K
##    4960  3806    (b): class >50K
##
##
##  Attribute usage:
##
##  100.00% age
##   75.25% education_num
##   75.25% sex
##   58.31% hours_per_week
##   16.44% workclass
##    1.72% race
##
##
## Time: 0.1 secs
```

Question 10: Decision Tree Confusion Matrix

```r
# Make predictions on the validation data
tree_predictions <- predict(dt_model, validation_data[,-8])
```

```r
# Confusion matrix
tree_matrix <- confusionMatrix(tree_predictions, validation_data$income)
print(tree_matrix)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  <=50K  >50K
##      <=50K   8708  1681
##      >50K     580  1240
##
##                Accuracy : 0.8148
##                  95% CI : (0.8078, 0.8217)
##     No Information Rate : 0.7608
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4158
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9376
##             Specificity : 0.4245
##          Pos Pred Value : 0.8382
##          Neg Pred Value : 0.6813
##              Prevalence : 0.7608
##          Detection Rate : 0.7132
##    Detection Prevalence : 0.8509
##       Balanced Accuracy : 0.6810
##
##        'Positive' Class :  <=50K
##
```

The decision tree model has an accuracy of 81.48%, with high sensitivity (TPR = 93.76%), but low specificity (TNR = 42.45%) , indicating that it again struggles to identify the minority class. The positive predictive value is high (PPV = 83.82%), but the negative predictive value (NPV = 68.13%) is lower, reflecting the model's bias towards predicting the majority class. With a Kappa of 0.415782 and balanced accuracy (NA%), the model's performance is good for <=50K but less effective for >50K, likely due to class imbalance in the dataset. This suggests the model is biased toward the majority class and would benefit from adjustments for handling imbalanced data.

The decision tree model has the highest Kappa value thus far (0.415782), indicating the best agreement between predicted and actual classes of our three models.

Question 11: Ensemble Model

```r
# Function to predict earnings class using an ensemble
predictEarningsClass <- function(train_data, validation_data, nb_model, logistic_model, dt_model) {

  # Make predictions for each model

  # Naive Bayes
  nb_predictions <- predict(nb_model, validation_data)
```

```r
  # Logistic Regression
  logistic_predictions <- predict(logistic_model, validation_data, type = "response")
  logistic_predictions <- ifelse(logistic_predictions > 0.5, " >50K", " <=50K")
  logistic_predictions <- factor(logistic_predictions, levels = levels(validation_data$income))

  # Decision Tree
  dt_predictions <- predict(dt_model, validation_data[,-8])

  # Combine predictions using majority vote
  combined_predictions <- data.frame(
    nb = nb_predictions,
    logistic = logistic_predictions,
    dt = dt_predictions
  )

  # Determine final prediction
  # Asked ChatGPT to help format my function here
  final_predictions <- apply(combined_predictions, 1, function(x) {
    vote_count <- table(x)
    # Class with highest count
    return(names(vote_count)[which.max(vote_count)])
  })

  # Convert final prediction to factor
  final_predictions <- factor(final_predictions, levels = levels(validation_data$income))
  return(final_predictions)
}
```

Question 12: Ensemble Model Prediction

```r
# Create new data point
new_data <- data.frame(
  age = 35,
  workclass = factor("Private", levels = levels(subset_combined_df$workclass)),
  education_num = 7,
  race = factor("Black", levels = levels(subset_combined_df$race)),
  sex = factor("Male", levels = levels(subset_combined_df$sex)),
  hours_per_week = 40,
  native_country = factor("Brazil", levels = levels(subset_combined_df$native_country)),
  income = factor(" <=50K", levels = levels(subset_combined_df$income))  # Placeholder
)

# Apply binning to continuous variables (age, education_num, hours_per_week)
new_data$age <- binning_function(as.numeric(new_data$age), age_breaks)
new_data$education_num <- binning_function(as.numeric(new_data$education_num), education_num_breaks)
new_data$hours_per_week <- binning_function(as.numeric(new_data$hours_per_week), hours_per_week_breaks)

# Predict using the ensemble model
final_prediction <- predictEarningsClass(train_data, new_data, nb_model, logistic_model, dt_model)
print(final_prediction)
```

```
##      1
##  <=50K
## Levels:  <=50K  >50K
```

BONUS: Question 13: Calculate F1 Scores

Problem 2: Multiple Regression

Question 1: Load Data

```r
# Load Data
sales_path <- "HomeSalesUFFIData.csv"
sales.df <- read.csv(sales_path, sep = ",", header = T, stringsAsFactors = T)

# Pre-process SalesPrice data
sales.df$SalesPrice <- as.numeric(gsub("[\\$,]", "", as.character(sales.df$SalesPrice)))
```

Question 2: Identify Outliers

```r
# Function to remove outliers based on Z-score and count outliers in each column
# Used a similar function to what I used in a past assignment
remove_outliers_zscore <- function(df, threshold = 3) {

  # Create empty data frame
  df_no_outliers <- df

  # Loop through each numeric column
  outlier_count <- list()
  for (colname in colnames(df)) {
    if (is.numeric(df[[colname]])) {

      # Calculate Z-scores for column
      mean_col <- mean(df[[colname]], na.rm = TRUE)
      sd_col <- sd(df[[colname]], na.rm = TRUE)
      z_scores <- (df[[colname]] - mean_col) / sd_col

      # Identify outliers
      outliers <- abs(z_scores) > threshold
      outlier_count[[colname]] <- sum(outliers, na.rm = TRUE)

      # Remove rows where the Z-score is greater than threshold
      df_no_outliers <- df_no_outliers[!outliers, ]
    }
  }

  # Print the number of outliers for each column
  cat("# of outliers:\n")
  for (colname in names(outlier_count)) {
    cat(colname, ": ", outlier_count[[colname]], "\n")
  }
```

```r
  return(df_no_outliers)
}

# Dataset without outliers
sales.no.df <- remove_outliers_zscore(sales.df)
```

```
## # of outliers:
## Observation :  0
## YearSold :  0
## SalesPrice :  3
## UFFI.Present :  0
## HasBrickExt :  0
## Gt45YrOld :  0
## Finished.Bsmnt :  0
## LotSizeSqFt :  0
## NumEncParkSpaces :  0
## LivingAreaSqFt :  2
## HasAC :  0
## HasPool :  3
```

There are outliers in the SalesPrice, LivingAreaSqFt, and HasPool feature columns. I used z-scoring to identify and remove outliers, where values more than 3 Standard Deviations away from the mean were removed.

Question 3: Test Feature Normality

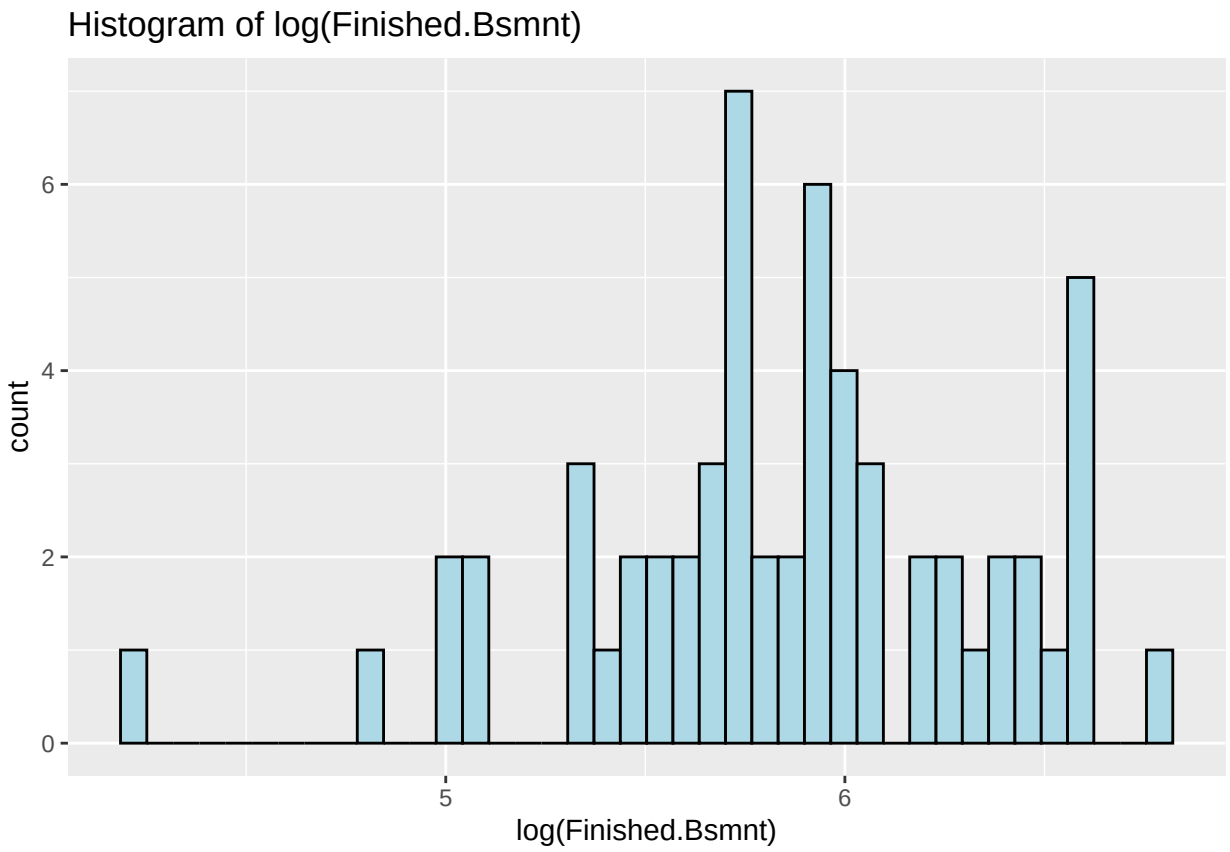
```r
# Set seed for reproducibility
set.seed(123)

# Function to test normality
test_normality <- function(df) {

  non_normal_columns <- c()
  continuous_numeric_cols <- c("Finished.Bsmnt", "LotSizeSqFt", "LivingAreaSqFt")

  # Loop through each column
  for (colname in continuous_numeric_cols) {

    # Shapiro-Wilk test
    shapiro_result <- shapiro.test(df[[colname]])

    # If p-value < 0.05, column not normally distributed
    if (shapiro_result$p.value < 0.05) {
      non_normal_columns <- c(non_normal_columns, colname)
    }
  }
  return(non_normal_columns)
}

# Run the function
non_normal_columns <- test_normality(sales.no.df)
print(non_normal_columns)
```

```
## [1] "Finished.Bsmnt" "LotSizeSqFt"    "LivingAreaSqFt"
```

Question 4: Normalize Features

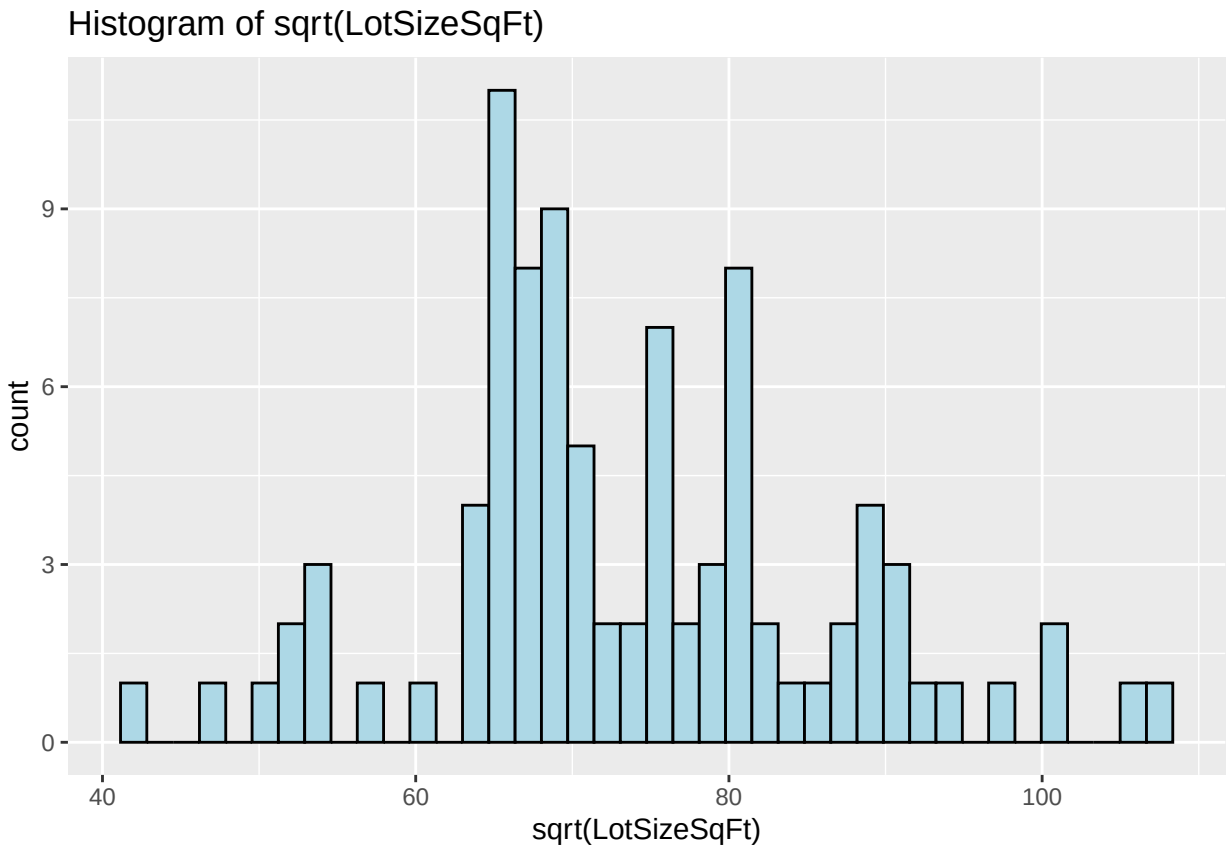
```
# Transform to approximately normal distributions

# Histogram of log(Finished.Bsmnt)
ggplot(sales.no.df, aes(x = log(Finished.Bsmnt))) +
  geom_histogram(bins = 40, fill = "lightblue", color = "black") +
  labs(title = "Histogram of log(Finished.Bsmnt)")
```

```
## Warning: Removed 38 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



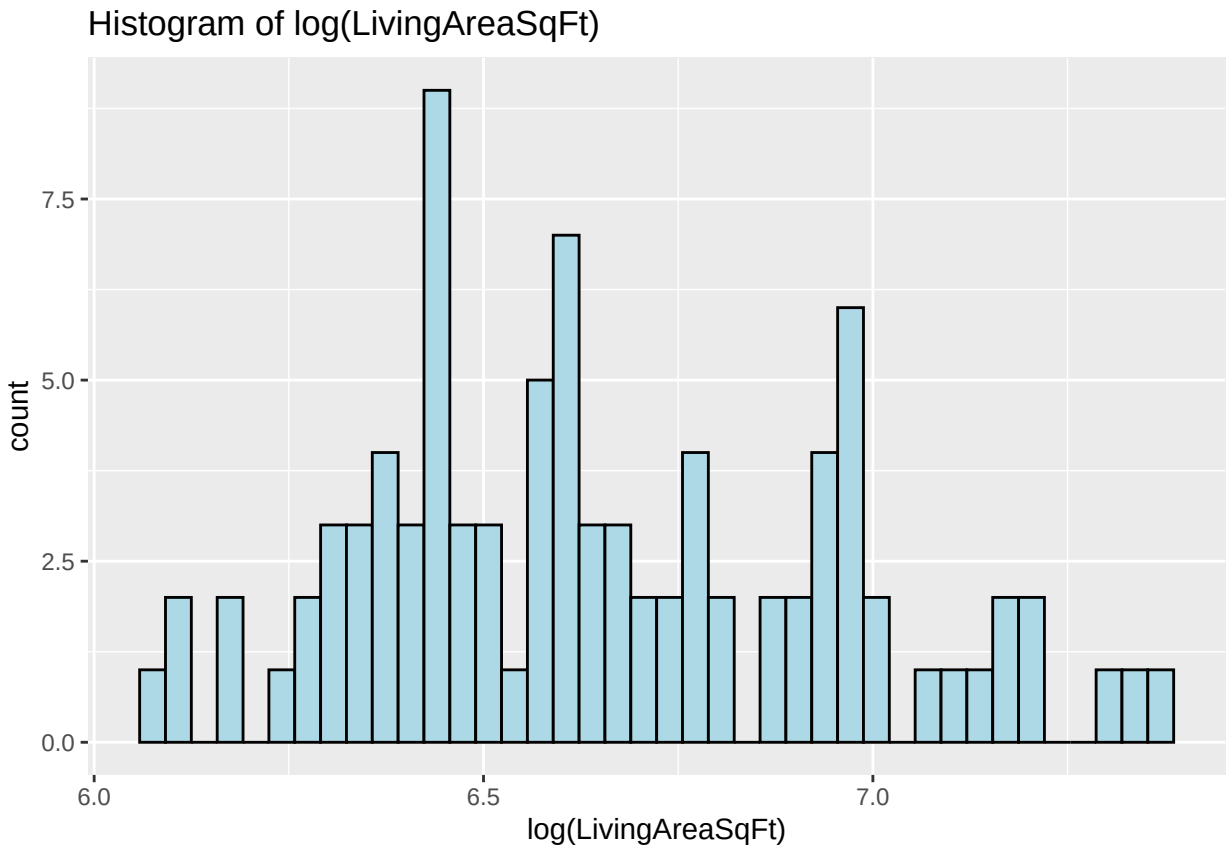

```
# Histogram of sqrt(LotSizeSqFt)
ggplot(sales.no.df, aes(x = sqrt(LotSizeSqFt))) +
  geom_histogram(bins = 40, fill = "lightblue", color = "black") +
  labs(title = "Histogram of sqrt(LotSizeSqFt)")
```

```
## Warning: Removed 6 rows containing non-finite outside the scale range
## (`stat_bin()`).
```




```
# Histogram of log(LivingAreaSqFt)
ggplot(sales.no.df, aes(x = log(LivingAreaSqFt))) +
  geom_histogram(bins = 40, fill = "lightblue", color = "black") +
  labs(title = "Histogram of log(LivingAreaSqFt)")
```

```
## Warning: Removed 6 rows containing non-finite outside the scale range
## (`stat_bin()`).
```




```
# Update data to reflect transformations
sales.tx <- sales.no.df %>%
  mutate(
    Finished.Bsmnt = log(Finished.Bsmnt +1),
    LotSizeSqFt = sqrt(LotSizeSqFt),
    LivingAreaSqFt = log(LivingAreaSqFt +1)
  )

# Remove any NAs produced by normalization
sales.tx <- na.omit(sales.tx)
```

Question 5: Variable Correlations

```
# Calculate correlations for all numeric variables
cor(sales.tx[sapply(sales.tx, is.numeric)], use = "complete.obs")
```

```
##                Observation    YearSold  SalesPrice UFFI.Present HasBrickExt
## Observation     1.00000000 -0.03048400  0.02809944   0.06424995  0.25135276
## YearSold       -0.03048400  1.00000000  0.68474772  -0.19472463  0.20519567
## SalesPrice      0.02809944  0.68474772  1.00000000  -0.19522779  0.11980013
## UFFI.Present    0.06424995 -0.19472463 -0.19522779   1.00000000 -0.01166050
## HasBrickExt     0.25135276  0.20519567  0.11980013  -0.01166050  1.00000000
## Gt45YrOld      -0.63557495 -0.13129397 -0.17080074   0.04332979 -0.26911052
```

```
## Finished.Bsmnt    0.23824580  0.08221408  0.22211954  -0.11758654 -0.13728475
## LotSizeSqFt       0.15968114  0.24986253  0.33856213   0.13305481 -0.10209693
## NumEncParkSpaces -0.05433124  0.22920374  0.37812742  -0.12430927 -0.10434667
## LivingAreaSqFt   -0.03121250  0.32817106  0.62770377   0.04850651  0.11523667
## HasAC             0.21489457  0.05823253  0.22993698  -0.02963695 -0.06557412
## HasPool           0.16496384 -0.11357827  0.01308617  -0.05952036 -0.08141057
##                    Gt45YrOld Finished.Bsmnt LotSizeSqFt NumEncParkSpaces
## Observation      -0.63557495     0.23824580  0.15968114      -0.05433124
## YearSold         -0.13129397     0.08221408  0.24986253       0.22920374
## SalesPrice       -0.17080074     0.22211954  0.33856213       0.37812742
## UFFI.Present      0.04332979    -0.11758654  0.13305481      -0.12430927
## HasBrickExt      -0.26911052    -0.13728475 -0.10209693      -0.10434667
## Gt45YrOld         1.00000000    -0.31178826 -0.41086464       0.04682929
## Finished.Bsmnt   -0.31178826     1.00000000  0.22291777       0.01275459
## LotSizeSqFt      -0.41086464     0.22291777  1.00000000       0.19996399
## NumEncParkSpaces  0.04682929     0.01275459  0.19996399       1.00000000
## LivingAreaSqFt    0.03418261    -0.12779887  0.18959712       0.16745839
## HasAC            -0.20519567     0.36317192  0.27250109       0.12491918
## HasPool          -0.23726841     0.09160513  0.05759632      -0.11754386
##                  LivingAreaSqFt       HasAC     HasPool
## Observation         -0.03121250  0.21489457  0.16496384
## YearSold             0.32817106  0.05823253 -0.11357827
## SalesPrice           0.62770377  0.22993698  0.01308617
## UFFI.Present         0.04850651 -0.02963695 -0.05952036
## HasBrickExt          0.11523667 -0.06557412 -0.08141057
## Gt45YrOld            0.03418261 -0.20519567 -0.23726841
## Finished.Bsmnt      -0.12779887  0.36317192  0.09160513
## LotSizeSqFt          0.18959712  0.27250109  0.05759632
## NumEncParkSpaces     0.16745839  0.12491918 -0.11754386
## LivingAreaSqFt       1.00000000  0.16928889 -0.10050347
## HasAC                0.16928889  1.00000000  0.09128709
## HasPool             -0.10050347  0.09128709  1.00000000
```

The correlations of each feature to the target variable can be found in the target variable column (SalesPrice) of the correlation matrix above. The only feature that exhibits a correlation above 0.6 is YearSold.

Question 6: Partition 3 Datasets

```
# Clean the data
sales.df <- na.omit(sales.df)
sales.no.df <- na.omit(sales.no.df)

# Partition sales.no.df, sales.df, and sales.tx
sales.indices <- createDataPartition(sales.df$SalesPrice, p = 0.85, list = FALSE)
sales.no.indices <- createDataPartition(sales.no.df$SalesPrice, p = 0.85, list = FALSE)
sales.tx.indices <- createDataPartition(sales.tx$SalesPrice, p = 0.85, list = FALSE)

# Subset the data into training and testing sets
sales.training <- sales.df[sales.indices, ]
sales.testing <- sales.df[-sales.indices, ]

sales.no.training <- sales.no.df[sales.no.indices, ]
```

```r
sales.no.testing <- sales.no.df[-sales.no.indices, ]

sales.tx.training <- sales.tx[sales.tx.indices, ]
sales.tx.testing <- sales.tx[-sales.tx.indices, ]
```

Question 7: Multiple Regression Models

```r
# Original Full Dataset
# Removed Features: Gt45YrOld, Observation, Finished.Bsmnt, UFFI.Present, HasAC
sales.df.model <- lm(SalesPrice ~ YearSold+HasBrickExt+LotSizeSqFt+NumEncParkSpaces+
                       LivingAreaSqFt+HasPool, data = sales.training)
summary.df.model <- summary(sales.df.model)

# Outliers Removed Dataset
# Removed Features:  Observation, Gt45YrOld, LotSizeSqFt, HasAC, HasBrickExt, UFFI.Present, HasPool
sales.no.model <- lm(SalesPrice ~ YearSold+Finished.Bsmnt+NumEncParkSpaces+
                       LivingAreaSqFt, data = sales.no.training)
summary.no.model <- summary(sales.no.model)

# Normalized and Outliers Removed dataset
# Removed Features: HasAC, Gt45YrOld, Observation, LotSizeSqFt, HasBrickExt, UFFI.Present
sales.tx.model <- lm(SalesPrice ~ YearSold+Finished.Bsmnt+NumEncParkSpaces+
                       LivingAreaSqFt+HasPool, data = sales.tx.training)
summary.tx.model <- summary(sales.tx.model)
```

Question 8: Analyze Models

```r
# Extract Residual Standard Error (RSE)
RSE.df.model <- summary.df.model$sigma
RSE.no.model <- summary.no.model$sigma
RSE.tx.model <- summary.tx.model$sigma

# Extract Adjusted R-squared (ARS)
ARS.df.model <- summary.df.model$adj.r.squared
ARS.no.model <- summary.no.model$adj.r.squared
ARS.tx.model <- summary.tx.model$adj.r.squared

# Predict on testing data
pred.df.model <- predict(sales.df.model, newdata = sales.testing)
pred.no.model <- predict(sales.no.model, newdata = sales.no.testing)
pred.tx.model <- predict(sales.tx.model, newdata = sales.tx.testing)

# Calculate RMSE for each model
RMSE.df.model <- sqrt(mean((pred.df.model - sales.testing$SalesPrice)^2))
RMSE.no.model <- sqrt(mean((pred.no.model - sales.no.testing$SalesPrice)^2))
RMSE.tx.model <- sqrt(mean((pred.tx.model - sales.tx.testing$SalesPrice)^2))

# Output Adjusted R-Squared and RMSE
cat("Adjusted R-Squared for sales.df.model:", ARS.df.model, "\n")
```

```
## Adjusted R-Squared for sales.df.model: 0.7436453
```

```
cat("Adjusted R-Squared for sales.no.model:", ARS.no.model, "\n")
```

```
## Adjusted R-Squared for sales.no.model: 0.7036586
```

```
cat("Adjusted R-Squared for sales.tx.model:", ARS.tx.model, "\n")
```

```
## Adjusted R-Squared for sales.tx.model: 0.7493693
```

```
cat("RMSE for sales.df.model:", RMSE.df.model, "\n")
```

```
## RMSE for sales.df.model: 22989.93
```

```
cat("RMSE for sales.no.model:", RMSE.no.model, "\n")
```

```
## RMSE for sales.no.model: 17533.75
```

```
cat("RMSE for sales.tx.model:", RMSE.tx.model, "\n")
```

```
## RMSE for sales.tx.model: 23266.54
```

Based on the RMSE alone, sales.no.model (outliers removed) seems to perform the best as it has by far the lowest RMSE (1.7533751×10^4). However, it also has the lowest Adjusted R-Squared, which suggests that it may not explain as much variance in the data as the other models. This indicates a trade-off between predictive accuracy (RMSE) and explanatory power (ARS). If the goal is to maximize the amount of variance explained by the model and understand the relationships between various predictors and SalesPrice, the Outliers-Removed and Features-Normalized Dataset with an Adjusted R-Squared of 0.749 is the best. It explains the most variance in the target variable, though at the cost of increased residual error.

BONUS: Question 9: Confidence Intervals

```
# Prediction intervals for the original model (sales.df.model)
pred_df_with_intervals <- predict(sales.df.model, newdata = sales.testing, interval = "prediction",
                                  level = 0.95)

# Prediction intervals for the outliers removed model (sales.no.model)
pred_no_with_intervals <- predict(sales.no.model, newdata = sales.no.testing, interval = "prediction",
                                  level = 0.95)

# Prediction intervals for the normalized model (sales.tx.model)
pred_tx_with_intervals <- predict(sales.tx.model, newdata = sales.tx.testing, interval = "prediction",
                                  level = 0.95)

# Combine the predictions with intervals into dataframes for easier review
pred_df_intervals <- as.data.frame(pred_df_with_intervals)
pred_no_intervals <- as.data.frame(pred_no_with_intervals)
pred_tx_intervals <- as.data.frame(pred_tx_with_intervals)
```

```r
# Add column names for clarity
colnames(pred_df_intervals) <- c("Prediction", "Lower_95", "Upper_95")
colnames(pred_no_intervals) <- c("Prediction", "Lower_95", "Upper_95")
colnames(pred_tx_intervals) <- c("Prediction", "Lower_95", "Upper_95")

# View the first few rows of the prediction intervals for each model
head(pred_df_intervals)
```

```
##    Prediction Lower_95 Upper_95
## 2    246054.0 168586.1 323522.0
## 7    228568.9 151195.2 305942.7
## 23   183276.3 106173.4 260379.2
## 31   179446.4 101392.9 257499.9
## 42   253491.0 175059.0 331923.0
## 59   210150.5 131786.0 288514.9
```

```r
head(pred_no_intervals)
```

```
##    Prediction Lower_95 Upper_95
## 2    247009.8 192002.7 302017.0
## 9    222596.7 166643.2 278550.2
## 30   158522.2 103187.2 213857.2
## 31   206403.2 150669.8 262136.7
## 54   181743.4 124661.0 238825.7
## 57   209932.8 152683.7 267182.0
```

```r
head(pred_tx_intervals)
```

```
##    Prediction Lower_95 Upper_95
## 5    210583.5 158397.9 262769.1
## 9    221673.9 168541.5 274806.2
## 28   188560.9 134409.7 242712.0
## 38   225444.3 173240.6 277648.0
## 45   244006.2 191596.6 296415.9
## 48   255145.2 202507.1 307783.3
```