Feature Engineering for Churn Prediction

Andres Perez

Feature Engineering for Churn Prediction

This document outlines the feature engineering process for the churn prediction model. We'll explore various transformations and create new features to improve model performance.

Data Loading and Initial Setup

```
# Load the data
data <- read.csv("data/ChurnData.csv")

# Display initial structure
str(data)
```

```
## 'data.frame':    5713 obs. of  13 variables:
##  $ ID                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Customer.Months      : int  67 67 55 63 57 58 57 46 56 56 ...
##  $ Churn                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ CHI.Score.Mon0       : int  0 62 0 231 43 138 180 116 78 78 ...
##  $ CHI.Score            : int  0 4 0 1 -1 -10 -5 -11 -7 -37 ...
##  $ Support.Cases.Mon0   : int  0 0 0 1 0 0 1 0 1 0 ...
##  $ Support.Cases        : int  0 0 0 -1 0 0 1 0 -2 0 ...
##  $ SP.Mon0              : num  0 0 0 3 0 0 3 0 3 0 ...
##  $ SP                   : num  0 0 0 0 0 0 3 0 0 0 ...
##  $ Logins               : int  0 0 0 167 0 43 13 0 -9 -7 ...
##  $ Blog.Articles        : int  0 0 0 -8 0 0 -1 0 1 0 ...
##  $ Views                : int  0 -16 0 21996 9 -33 907 38 0 30 ...
##  $ Days.Since.Last.Login: int  31 31 31 0 31 0 0 6 7 14 ...
```

```
# Display first few rows
head(data)
```

```
##   ID Customer.Months Churn CHI.Score.Mon0 CHI.Score Support.Cases.Mon0
## 1  1              67     0              0         0                   0
## 2  2              67     0             62         4                   0
## 3  3              55     0              0         0                   0
## 4  4              63     0            231         1                   1
## 5  5              57     0             43        -1                   0
## 6  6              58     0            138       -10                   0
##   Support.Cases SP.Mon0 SP Logins Blog.Articles Views Days.Since.Last.Login
## 1             0       0  0      0             0     0                   31
## 2             0       0  0      0             0   -16                   31
## 3             0       0  0      0             0     0                   31
## 4            -1       3  0    167            -8 21996                    0
## 5             0       0  0      0             0     9                   31
## 6             0       0  0     43             0   -33                    0
```

```
# Summary statistics
summary(data)
```

```
##        ID       Customer.Months     Churn         CHI.Score.Mon0
##  Min.   :   1   Min.   : 1.00   Min.   :0.00000   Min.   :  0.00
##  1st Qu.:1561   1st Qu.: 5.00   1st Qu.:0.00000   1st Qu.: 25.00
##  Median :3172   Median :11.00   Median :0.00000   Median : 88.00
##  Mean   :3166   Mean   :13.91   Mean   :0.05094   Mean   : 87.45
##  3rd Qu.:4761   3rd Qu.:20.00   3rd Qu.:0.00000   3rd Qu.:139.00
##  Max.   :6347   Max.   :67.00   Max.   :1.00000   Max.   :298.00
##    CHI.Score       Support.Cases.Mon0 Support.Cases          SP.Mon0
##  Min.   :-125.00   Min.   : 0.0000    Min.   :-17.00000   Min.   :0.0000
##  1st Qu.:  -8.00   1st Qu.: 0.0000    1st Qu.:  0.00000   1st Qu.:0.0000
##  Median :   0.00   Median : 0.0000    Median :  0.00000   Median :0.0000
##  Mean   :   5.06   Mean   : 0.7098    Mean   : -0.01243   Mean   :0.8123
##  3rd Qu.:  15.00   3rd Qu.: 1.0000    3rd Qu.:  0.00000   3rd Qu.:2.6667
##  Max.   : 208.00   Max.   :32.0000    Max.   : 31.00000   Max.   :4.0000
##        SP               Logins        Blog.Articles          Views
##  Min.   :-4.00000   Min.   :-293.00   Min.   :-75.0000   Min.   :-28322.00
##  1st Qu.: 0.00000   1st Qu.:  -1.00   1st Qu.:  0.0000   1st Qu.:   -12.00
##  Median : 0.00000   Median :   2.00   Median :  0.0000   Median :     0.00
##  Mean   : 0.02592   Mean   :  15.68   Mean   :  0.1679   Mean   :    96.49
##  3rd Qu.: 0.00000   3rd Qu.:  23.00   3rd Qu.:  0.0000   3rd Qu.:    27.00
##  Max.   : 4.00000   Max.   : 865.00   Max.   :217.0000   Max.   :230414.00
##  Days.Since.Last.Login
##  Min.   :-648.000
##  1st Qu.:   0.000
##  Median :   0.000
##  Mean   :   1.876
##  3rd Qu.:   3.000
##  Max.   :  61.000
```

Feature Engineering Process

1. Numeric Feature Transformations

The goal of numeric feature transformations is to improve the predictive power of our model by: 1. Handling skewed distributions that could bias our model 2. Creating meaningful ratios that capture customer engagement patterns 3. Identifying interaction effects between different customer behaviors 4. Normalizing metrics by customer tenure to enable fair comparisons

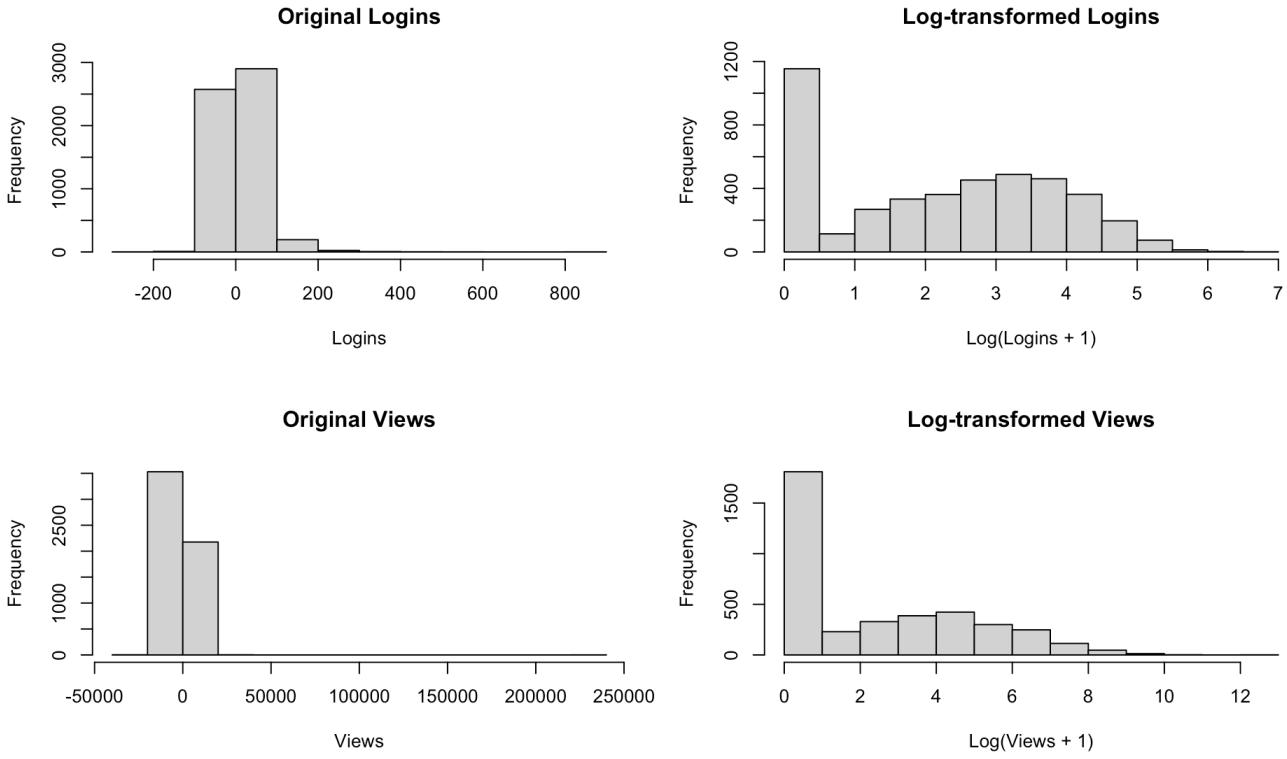
```
# Create a copy of the data for transformations
data_transformed <- data

# Log transformations for skewed numeric variables
# These transformations help normalize right-skewed distributions of engagement met
        rics
# and make the relationships more linear, which is beneficial for many modeling tec
        hniques
data_transformed$Logins_log <- log1p(data_transformed$Logins)
data_transformed$Views_log <- log1p(data_transformed$Views)
data_transformed$Blog.Articles_log <- log1p(data_transformed$Blog.Articles)

# Create ratio features
# These ratios help identify customers who are more engaged relative to their login
        frequency
# A high ratio might indicate more valuable customers who make the most of each log
        in
data_transformed$Views_per_Login <- ifelse(data_transformed$Logins > 0,
                                           data_transformed$Views / data_transformed
        $Logins,
                                           0)

data_transformed$Blog_per_Login <- ifelse(data_transformed$Logins > 0,
                                          data_transformed$Blog.Articles / data_trans
        formed$Logins,
                                          0)

# Create interaction features
# These interactions help capture combined effects of different customer behaviors
# For example, the relationship between support cases and CHI score might be differ
        ent
# for customers with different engagement levels
data_transformed$Support_Score_Interaction <- data_transformed$Support.Cases * data
        _transformed$CHI.Score
data_transformed$Login_View_Interaction <- data_transformed$Logins * data_transform
        ed$Views

# Create time-based features
# This normalizes engagement metrics by customer tenure to enable fair comparisons
# between customers who have been with the service for different lengths of time
data_transformed$Activity_Score <- (data_transformed$Logins + data_transformed$View
        s +
                                    data_transformed$Blog.Articles) / data_transforme
        d$Customer.Months

# Display summary of transformed features
print("Summary of original and transformed features:")
```

```
## [1] "Summary of original and transformed features:"
```

```
summary(data_transformed)
```

```
##        ID       Customer.Months     Churn         CHI.Score.Mon0
##  Min.   :   1   Min.   : 1.00   Min.   :0.00000   Min.   :  0.00
##  1st Qu.:1561   1st Qu.: 5.00   1st Qu.:0.00000   1st Qu.: 25.00
##  Median :3172   Median :11.00   Median :0.00000   Median : 88.00
##  Mean   :3166   Mean   :13.91   Mean   :0.05094   Mean   : 87.45
##  3rd Qu.:4761   3rd Qu.:20.00   3rd Qu.:0.00000   3rd Qu.:139.00
##  Max.   :6347   Max.   :67.00   Max.   :1.00000   Max.   :298.00
##
##    CHI.Score       Support.Cases.Mon0 Support.Cases          SP.Mon0
##  Min.   :-125.00   Min.   : 0.0000    Min.   :-17.00000   Min.   :0.0000
##  1st Qu.:  -8.00   1st Qu.: 0.0000    1st Qu.:  0.00000   1st Qu.:0.0000
##  Median :   0.00   Median : 0.0000    Median :  0.00000   Median :0.0000
##  Mean   :   5.06   Mean   : 0.7098    Mean   : -0.01243   Mean   :0.8123
##  3rd Qu.:  15.00   3rd Qu.: 1.0000    3rd Qu.:  0.00000   3rd Qu.:2.6667
##  Max.   : 208.00   Max.   :32.0000    Max.   : 31.00000   Max.   :4.0000
##
##        SP               Logins        Blog.Articles          Views
##  Min.   :-4.00000   Min.   :-293.00   Min.   :-75.0000   Min.   :-28322.00
##  1st Qu.: 0.00000   1st Qu.:  -1.00   1st Qu.:  0.0000   1st Qu.:   -12.00
##  Median : 0.00000   Median :   2.00   Median :  0.0000   Median :     0.00
##  Mean   : 0.02592   Mean   :  15.68   Mean   :  0.1679   Mean   :    96.49
##  3rd Qu.: 0.00000   3rd Qu.:  23.00   3rd Qu.:  0.0000   3rd Qu.:    27.00
##  Max.   : 4.00000   Max.   : 865.00   Max.   :217.0000   Max.   :230414.00
##
##  Days.Since.Last.Login   Logins_log      Views_log      Blog.Articles_log
##  Min.   :-648.000      Min.   : -Inf   Min.   :  -Inf   Min.   :  -Inf
##  1st Qu.:   0.000      1st Qu.:0.000   1st Qu.: 0.000   1st Qu.:0.0000
##  Median :   0.000      Median :2.303   Median : 1.386   Median :0.0000
##  Mean   :   1.876      Mean   : -Inf   Mean   :  -Inf   Mean   :  -Inf
##  3rd Qu.:   3.000      3rd Qu.:3.526   3rd Qu.: 4.382   3rd Qu.:0.6931
##  Max.   :  61.000      Max.   :6.764   Max.   :12.348   Max.   :5.3845
##                        NA's   :1301    NA's   :1743     NA's   :621
##  Views_per_Login     Blog_per_Login      Support_Score_Interaction
##  Min.   :-2943.857   Min.   :-75.00000   Min.   :-1005.00
##  1st Qu.:    0.000   1st Qu.:  0.00000   1st Qu.:    0.00
##  Median :    0.000   Median :  0.00000   Median :    0.00
##  Mean   :    4.083   Mean   : -0.00486   Mean   :   16.22
##  3rd Qu.:    0.000   3rd Qu.:  0.00000   3rd Qu.:    0.00
##  Max.   : 6863.000   Max.   :  6.50000   Max.   : 3131.00
##
##  Login_View_Interaction Activity_Score
##  Min.   :-3146642       Min.   :-2353.000
##  1st Qu.:    -122       1st Qu.:   -0.773
##  Median :       0       Median :    0.167
##  Mean   :    5356       Mean   :   13.331
##  3rd Qu.:     126       3rd Qu.:    6.960
##  Max.   :23041400       Max.   :23051.600
##
```

```
# Visualize the impact of transformations
par(mfrow = c(2, 2))
hist(data_transformed$Logins, main = "Original Logins", xlab = "Logins")
hist(data_transformed$Logins_log, main = "Log-transformed Logins", xlab = "Log(Logi
        ns + 1)")
hist(data_transformed$Views, main = "Original Views", xlab = "Views")
hist(data_transformed$Views_log, main = "Log-transformed Views", xlab = "Log(Views
        + 1)")
```



2. Feature Selection

```
# Data validation
print("Checking for missing values:")
```

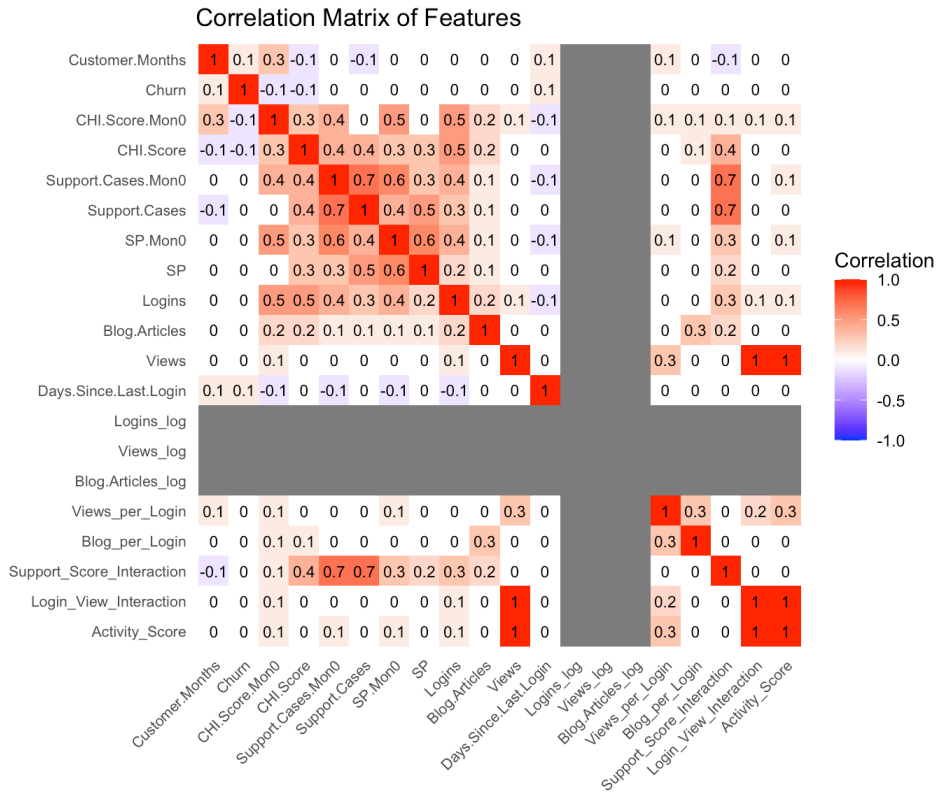
```
## [1] "Checking for missing values:"
```

```
print(colSums(is.na(data_transformed)))
```

```
##                         ID           Customer.Months                    Churn
##                          0                         0                        0
##             CHI.Score.Mon0               CHI.Score        Support.Cases.Mon0
##                          0                         0                        0
##              Support.Cases                 SP.Mon0                       SP
##                          0                         0                        0
##                     Logins             Blog.Articles                    Views
##                          0                         0                        0
##     Days.Since.Last.Login                Logins_log                Views_log
##                          0                      1301                     1743
##          Blog.Articles_log           Views_per_Login            Blog_per_Login
##                        621                         0                        0
## Support_Score_Interaction    Login_View_Interaction            Activity_Score
##                          0                         0                        0
```

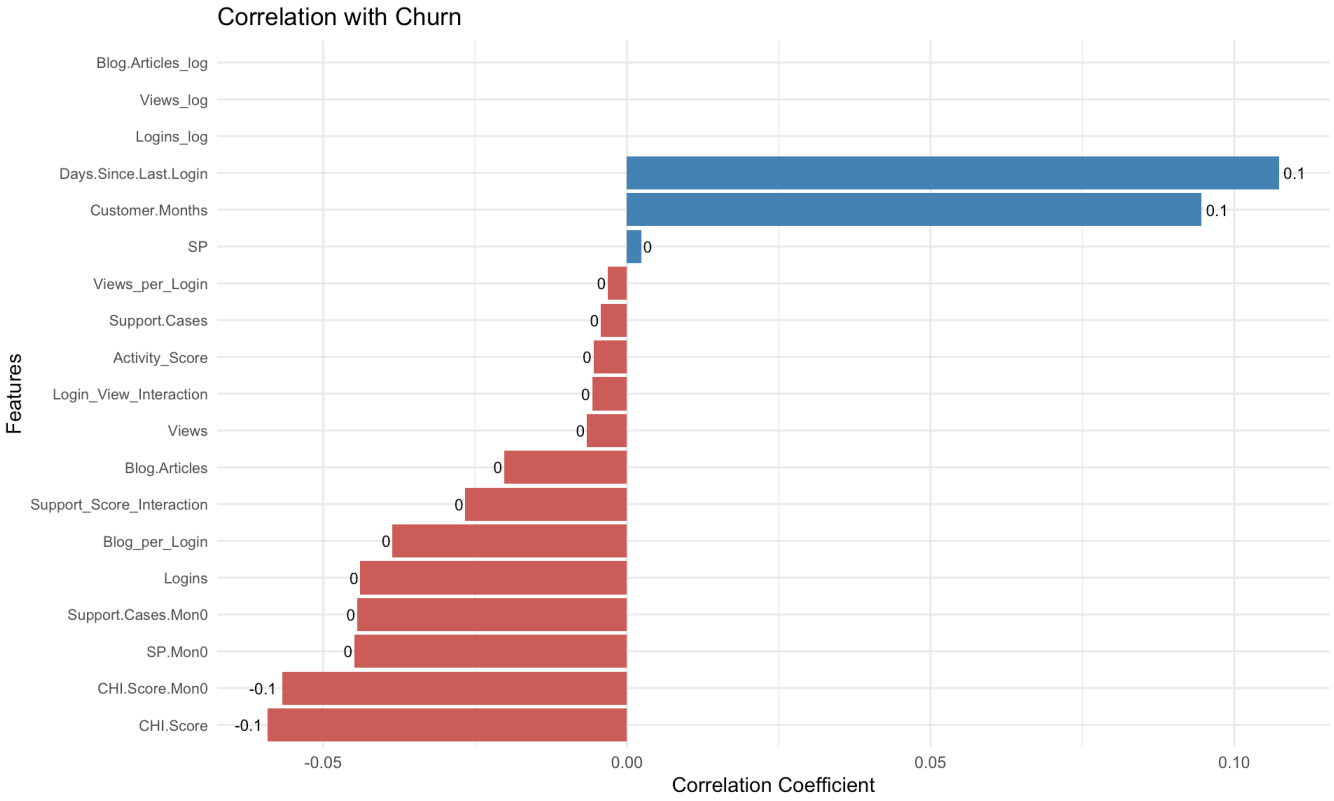
```r
# Remove any rows with missing values for correlation analysis
data_for_cor <- data_transformed %>%
  select(-ID) %>%
  select_if(is.numeric) %>%
  na.omit()

# Remove columns with all NA before correlation
cor_data <- data_for_cor[, colSums(is.na(data_for_cor)) < nrow(data_for_cor)]

if(ncol(cor_data) < 2) {
  cat("\nNot enough valid features to compute a correlation matrix.\n")
} else {
  cor_matrix <- cor(cor_data, use = "pairwise.complete.obs")
  cor_matrix_rounded <- round(cor_matrix, 1)

  # Convert correlation matrix to long format for ggplot
  cor_long <- as.data.frame(cor_matrix_rounded) %>%
    rownames_to_column("Var1") %>%
    pivot_longer(-Var1, names_to = "Var2", values_to = "Correlation") %>%
    mutate(Var1 = factor(Var1, levels = rev(rownames(cor_matrix))),
           Var2 = factor(Var2, levels = colnames(cor_matrix)))

  # Create correlation heatmap using ggplot2 (red=positive, blue=negative, white=0)
  print(
    ggplot(cor_long, aes(x = Var2, y = Var1, fill = Correlation)) +
      geom_tile() +
      geom_text(aes(label = Correlation), size = 3) +
      scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                           midpoint = 0, limit = c(-1, 1), space = "Lab",
                           name = "Correlation") +
      theme_minimal() +
      theme(axis.text.x = element_text(angle = 45, hjust = 1, size = 8),
            axis.text.y = element_text(size = 8),
            axis.title = element_blank(),
            panel.grid = element_blank()) +
      coord_fixed() +
      labs(title = "Correlation Matrix of Features")
  )

  # Identify highly correlated features
  cor_threshold <- 0.7
  high_cor_pairs <- which(abs(cor_matrix) > cor_threshold & upper.tri(cor_matrix),
        arr.ind = TRUE)

  if(nrow(high_cor_pairs) > 0) {
    print("Highly correlated feature pairs (correlation > 0.7):")
    for(i in 1:nrow(high_cor_pairs)) {
      row_name <- rownames(cor_matrix)[high_cor_pairs[i,1]]
      col_name <- colnames(cor_matrix)[high_cor_pairs[i,2]]
      cor_value <- round(cor_matrix[high_cor_pairs[i,1], high_cor_pairs[i,2]], 1)
      cat(sprintf("%s - %s: %.1f\n", row_name, col_name, cor_value))
    }
```

```
  } else {
    print("No highly correlated features found (correlation > 0.7)")
  }
}
```




```
## [1] "Highly correlated feature pairs (correlation > 0.7):"
## Support.Cases - Support_Score_Interaction: 0.7
## Views - Login_View_Interaction: 1.0
## Views - Activity_Score: 1.0
## Login_View_Interaction - Activity_Score: 1.0
```

```r
# Plot correlation with target variable
target_correlations <- cor_matrix[, "Churn"]
target_correlations <- target_correlations[order(abs(target_correlations), decreasi
        ng = TRUE)]

# Create a bar plot of correlations with target using ggplot2
target_cor_df <- data.frame(
  Feature = names(target_correlations),
  Correlation = target_correlations
) %>%
  filter(Feature != "Churn") %>%
  mutate(Feature = factor(Feature, levels = Feature[order(Correlation)]))

ggplot(target_cor_df, aes(x = Feature, y = Correlation, fill = Correlation > 0)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  scale_fill_manual(values = c("TRUE" = "steelblue", "FALSE" = "indianred")) +
  theme_minimal() +
  theme(axis.text.y = element_text(size = 8),
        legend.position = "none") +
  labs(title = "Correlation with Churn",
       x = "Features",
       y = "Correlation Coefficient") +
  geom_text(aes(label = round(Correlation, 1)),
            hjust = ifelse(target_cor_df$Correlation > 0, -0.2, 1.2),
            size = 3)
```



There is visibly no strong correlation between any feature and churn, so we will proceed with non-linear models, as linear relationships are not present in the data.

3. Feature Importance

We will now train a simple random forest model to identify the most important features for predicting churn. Random forests can capture complex, non-linear relationships that correlation analysis may miss. This helps us discover which features are most useful for our predictive modeling, even if they don't have a strong linear relationship with churn.

```
# Prepare data for feature importance
set.seed(123)

# Remove rows with missing values and prepare data
data_for_importance <- data_transformed %>%
  select(-ID) %>%
  select_if(is.numeric) %>%
  na.omit()

# Print the number of rows after removing missing values
print(paste("Number of complete cases:", nrow(data_for_importance)))
```

```
## [1] "Number of complete cases: 2896"
```

```
# Check for infinite values
print("Checking for infinite values:")
```

```
## [1] "Checking for infinite values:"
```

```
print(colSums(is.infinite(as.matrix(data_for_importance))))
```

```
##           Customer.Months                     Churn            CHI.Score.Mon0
##                         0                         0                         0
##                 CHI.Score       Support.Cases.Mon0             Support.Cases
##                         0                         0                         0
##                   SP.Mon0                        SP                   Logins
##                         0                         0                         0
##             Blog.Articles                     Views     Days.Since.Last.Login
##                         0                         0                         0
##                Logins_log                Views_log         Blog.Articles_log
##                        84                        50                       223
##           Views_per_Login           Blog_per_Login Support_Score_Interaction
##                         0                         0                         0
##    Login_View_Interaction           Activity_Score
##                         0                         0
```

```
# Replace infinite values with NA and then remove them
data_for_importance <- data_for_importance %>%
  mutate(across(everything(), ~ifelse(is.infinite(.), NA, .))) %>%
  na.omit()

# Ensure Churn is a factor
data_for_importance$Churn <- as.factor(data_for_importance$Churn)

# Print data structure before modeling
print("Data structure before modeling:")
```

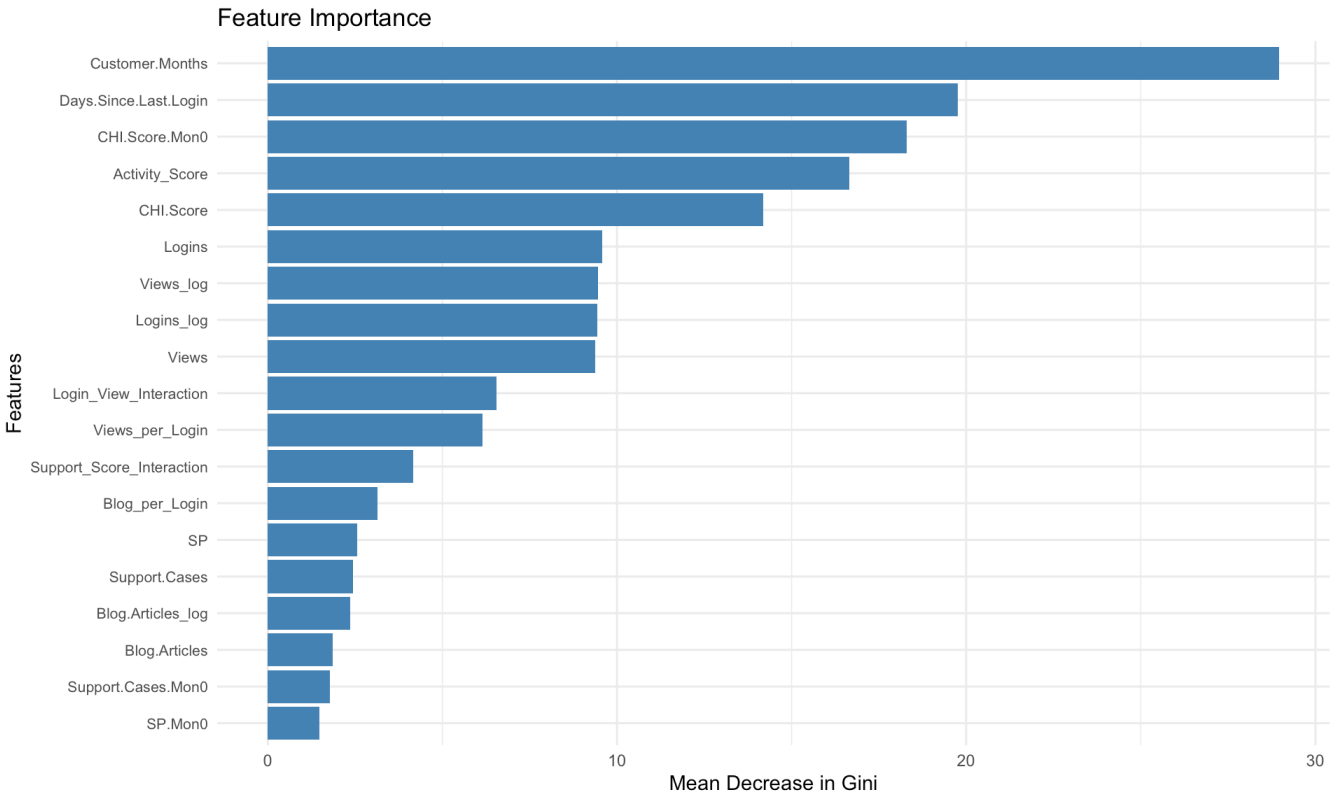
```
## [1] "Data structure before modeling:"
```

```
str(data_for_importance)
```

```
## 'data.frame':    2554 obs. of  20 variables:
##  $ Customer.Months          : int  67 55 57 46 53 56 57 57 55 56 ...
##  $ Churn                    : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1
...
##  $ CHI.Score.Mon0           : int  0 0 43 116 91 40 215 0 118 59 ...
##  $ CHI.Score                : int  0 0 -1 -11 -1 14 15 0 63 16 ...
##  $ Support.Cases.Mon0       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Support.Cases            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SP.Mon0                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ SP                       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Logins                   : int  0 0 0 0 14 0 71 0 5 0 ...
##  $ Blog.Articles            : int  0 0 0 0 3 0 9 0 1 0 ...
##  $ Views                    : int  0 0 9 38 0 15 8658 0 995 4 ...
##  $ Days.Since.Last.Login    : int  31 31 31 6 0 31 0 31 0 31 ...
##  $ Logins_log               : num  0 0 0 0 2.71 ...
##  $ Views_log                : num  0 0 2.3 3.66 0 ...
##  $ Blog.Articles_log        : num  0 0 0 0 1.39 ...
##  $ Views_per_Login          : num  0 0 0 0 0 ...
##  $ Blog_per_Login           : num  0 0 0 0 0.214 ...
##  $ Support_Score_Interaction: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Login_View_Interaction   : int  0 0 0 0 0 0 614718 0 4975 0 ...
##  $ Activity_Score           : num  0 0 0.158 0.826 0.321 ...
##  - attr(*, "na.action")= 'omit' Named int [1:342] 4 20 31 32 40 44 53 68 83 85
...
##   ..- attr(*, "names")= chr [1:342] "7" "34" "59" "61" ...
```

```r
# Train a simple random forest model with error handling
tryCatch({
  rf_model <- randomForest(Churn ~ .,
                           data = data_for_importance,
                           importance = TRUE,
                           ntree = 100)

  # Get feature importance
  importance_data <- as.data.frame(importance(rf_model))
  importance_data$Feature <- rownames(importance_data)
  importance_data <- importance_data[order(importance_data$MeanDecreaseGini, decrea
        sing = TRUE),]

  # Plot feature importance
  print(ggplot(importance_data, aes(x = reorder(Feature, MeanDecreaseGini), y = Mea
        nDecreaseGini)) +
    geom_bar(stat = "identity", fill = "steelblue") +
    coord_flip() +
    theme_minimal() +
    labs(title = "Feature Importance",
         x = "Features",
         y = "Mean Decrease in Gini") +
    theme(axis.text.y = element_text(size = 8)))

  # Print top 10 most important features
  print("Top 10 most important features:")
  print(head(importance_data[, c("Feature", "MeanDecreaseGini")], 10))
}, error = function(e) {
  print("Error in random forest model:")
  print(e)

  # Fallback to correlation-based importance
  print("Using correlation-based importance instead:")
  cor_importance <- abs(cor(data_for_importance %>% select(-Churn),
                            as.numeric(data_for_importance$Churn) - 1))
  cor_importance <- data.frame(
    Feature = rownames(cor_importance),
    Importance = cor_importance[,1]
  )
  cor_importance <- cor_importance[order(cor_importance$Importance, decreasing = TR
        UE),]

  print(ggplot(cor_importance, aes(x = reorder(Feature, Importance), y = Importanc
        e)) +
    geom_bar(stat = "identity", fill = "steelblue") +
    coord_flip() +
    theme_minimal() +
    labs(title = "Feature Importance (Correlation-based)",
         x = "Features",
         y = "Absolute Correlation with Churn") +
    theme(axis.text.y = element_text(size = 8)))

  print("Top 10 most important features (correlation-based):")
```

```
  print(head(cor_importance, 10))
})
```




```
## [1] "Top 10 most important features:"
##                                     Feature MeanDecreaseGini
## Customer.Months               Customer.Months        28.963866
## Days.Since.Last.Login   Days.Since.Last.Login        19.752431
## CHI.Score.Mon0                 CHI.Score.Mon0        18.306257
## Activity_Score                 Activity_Score        16.651143
## CHI.Score                           CHI.Score        14.177672
## Logins                                 Logins         9.577671
## Views_log                           Views_log         9.463294
## Logins_log                         Logins_log         9.434210
## Views                                   Views         9.376403
## Login_View_Interaction Login_View_Interaction         6.540693
```

Next Steps

Now that we have identified the most important features using a random forest model, the next steps are:

1. Use these top predictors to build and tune more advanced machine learning models (such as random forests, gradient boosting, or neural networks) to improve churn prediction.
2. Perform cross-validation and hyperparameter tuning to optimize model performance.
3. Evaluate the models using appropriate metrics (such as accuracy, precision, recall, and ROC-AUC).
4. Interpret the results and, if necessary, iterate on feature engineering or try additional modeling approaches.

This process will help us develop a robust model for predicting customer churn based on the most informative features.

Feature Engineering Decisions

Rationale for Transformations

1. **Log Transformations**
   - Applied to `Logins`, `Views`, and `Blog.Articles` to handle right-skewed distributions
   - Used `log1p()` to handle zero values appropriately
2. **Ratio Features**
   - Created `Views_per_Login` and `Blog_per_Login` to capture engagement efficiency
   - These ratios help identify users who are more engaged relative to their login frequency
3. **Interaction Features**
   - Created `Support_Score_Interaction` to capture the relationship between support cases and CHI score
   - Added `Login_View_Interaction` to identify patterns in user engagement
4. **Time-Based Features**
   - Created `Activity_Score` to normalize engagement metrics by customer tenure
   - This helps compare engagement levels across customers with different subscription lengths

Next Steps

1. **Feature Validation**
   - Cross-validate the engineered features
   - Assess feature stability across different time periods
2. **Model Integration**
   - Prepare features for model training
   - Document feature requirements for production

Save Engineered Features for Modeling

```
# Save the engineered features (including Churn) to a new CSV for modeling
engineered_for_modeling <- data_transformed %>% select(-ID)
write.csv(engineered_for_modeling, "data/EngineeredChurnData.csv", row.names = FALS
        E)
cat("Engineered features saved to data/EngineeredChurnData.csv\n")
```

```
## Engineered features saved to data/EngineeredChurnData.csv
```