Predicting Remaining Subscription Months for a Streaming Platform Using Statistical Learning Methods

Nagasri Ramya Konduru

Table of contents

title: "Predicting Remaining Subscription Months for a Streaming Platform Using Statistical Learning Methods" author: "Nagasri Ramya Konduru" format: pdf: toc: true number-sections: true html: toc: true number-sections: true fontsize: 11pt execute: echo: true warning: false message: false

---

0.1 Load libraries

```r
library(tidyverse)
library(caret)
library(glmnet)
library(rpart)
library(rpart.plot)
library(randomForest)
library(xgboost)
```

1 Introduction

Subscription-based streaming platforms such as Netflix, Hulu, Disney+, and Prime Video rely on predictable recurring revenue from monthly subscribers. A key business question is: **For how long will a subscriber remain active?** Accurately predicting remaining subscription duration is essential for estimating customer lifetime value (CLV), optimizing marketing spend, designing retention strategies, and forecasting revenue.

The goal of this project is to use statistical learning methods to predict the number of months a user will stay subscribed (capped at 12 months), using simulated user behavior that reflects typical engagement trends observed on real streaming platforms.

2 Research Question

How accurately can we predict a subscriber's remaining subscription duration (0–12 months) using engagement, viewing behavior, and account characteristics on a streaming platform?

This project compares traditional econometric models with modern machine learning approaches to determine whether flexible nonlinear models improve predictive performance.

3 Data Description

Because real proprietary streaming data are unavailable, this project uses a **simulated dataset of 5,000 users**, designed to reflect realistic patterns in:

- **Account characteristics**
- **Platform usage**
- **Viewing preferences**
- **Behavioral signals**

The dataset includes the following variables:

Category	Variables
Outcome	remaining_months (0–12)
Account	account_age_months, plan_type
Usage	hours_watched_last_month, active_days_last_month, logins_per_week
Device Mix	share_mobile, share_tv, share_laptop
Content Preferences	genre_diversity, avg_binge_length
Behavioral Signals	payment_failures_6m, paused, shared_account

Below is the code used to generate the dataset.

4 Data Generation

```r
set.seed(123)
n <- 5000

data <- tibble(
remaining_months = pmin(12, round(rnorm(n, 7, 3))),
account_age_months = pmax(1, round(rnorm(n, 18, 10))),
plan_type = sample(c("Basic", "Standard", "Premium"), n, replace = TRUE,
prob = c(0.4, 0.4, 0.2)),
hours_watched_last_month = abs(rnorm(n, 40, 20)),
active_days_last_month = pmax(1, round(rnorm(n, 12, 5))),
logins_per_week = abs(rnorm(n, 5, 2)),
share_mobile = runif(n, 0, 0.7),
```

```r
share_tv = runif(n, 0, 0.7),
share_laptop = pmax(0, 1 - share_mobile - share_tv),
genre_diversity = abs(rnorm(n, 4, 2)),
avg_binge_length = abs(rnorm(n, 2, 1)),
payment_failures_6m = rpois(n, 0.3),
paused = factor(sample(c(0, 1), n, replace = TRUE, prob = c(0.85, 0.15))),
shared_account = factor(sample(c(0, 1), n, replace = TRUE, prob = c(0.7, 0.3)))
)

data_raw <- data
```

5 Summary Statistics

```r
summary(data_raw)
```

```
 remaining_months account_age_months  plan_type
 Min.   :-2.000   Min.   : 1.00      Length:5000
 1st Qu.: 5.000   1st Qu.:11.00      Class :character
 Median : 7.000   Median :18.00      Mode  :character
 Mean   : 6.935   Mean   :18.14
 3rd Qu.: 9.000   3rd Qu.:25.00
 Max.   :12.000   Max.   :56.00
 hours_watched_last_month active_days_last_month logins_per_week
 Min.   :3.925e-03        Min.   : 1.00          Min.   : 0.01422
 1st Qu.:2.645e+01        1st Qu.: 9.00          1st Qu.: 3.57776
 Median :3.961e+01        Median :12.00          Median : 4.93966
 Mean   :4.017e+01        Mean   :11.91          Mean   : 4.97192
 3rd Qu.:5.329e+01        3rd Qu.:15.00          3rd Qu.: 6.36867
 Max.   :1.063e+02        Max.   :32.00          Max.   :13.64563
  share_mobile          share_tv          share_laptop     genre_diversity
 Min.   :0.0001645   Min.   :0.0001802   Min.   :0.00000   Min.   : 0.001926
 1st Qu.:0.1834786   1st Qu.:0.1798189   1st Qu.:0.09487   1st Qu.: 2.626249
 Median :0.3536722   Median :0.3439636   Median :0.29535   Median : 4.029422
 Mean   :0.3543318   Mean   :0.3485160   Mean   :0.31906   Mean   : 4.016608
 3rd Qu.:0.5312687   3rd Qu.:0.5200592   3rd Qu.:0.50242   3rd Qu.: 5.319153
 Max.   :0.6998697   Max.   :0.6999473   Max.   :0.98993   Max.   :11.965556
 avg_binge_length    payment_failures_6m paused   shared_account
 Min.   :0.0008189   Min.   :0.0000      0:4225   0:3489
 1st Qu.:1.3245229   1st Qu.:0.0000      1: 775   1:1511
```

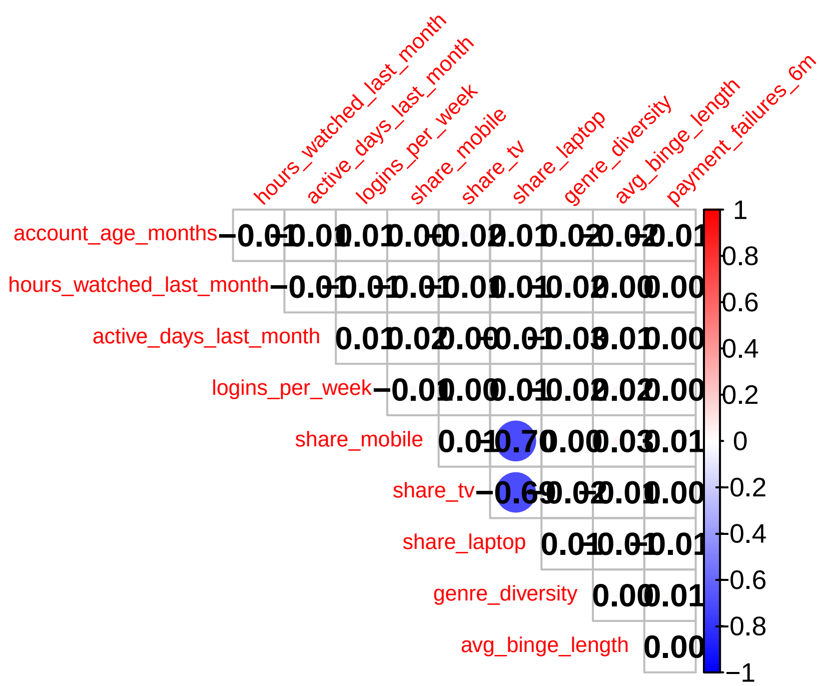
```
 Median :2.0108233   Median :0.0000
 Mean   :2.0158892   Mean   :0.3006
 3rd Qu.:2.6657474   3rd Qu.:1.0000
 Max.   :5.4093073   Max.   :4.0000
```

```r
# Correlation Plot
# Correlation Plot with adjusted aesthetics

library(corrplot)
corr_data <- select(data, -remaining_months, -plan_type, -paused, -shared_account)  # Exclud
corr_matrix <- cor(corr_data)

# Improved plot with rotated labels and color adjustments

corrplot(corr_matrix, method = "circle", type = "upper",
tl.cex = 0.7, # Reduce label size
tl.srt = 45,  # Rotate labels to 45 degrees
addCoef.col = "black", # Add correlation coefficients on the plot
diag = FALSE, # Hide diagonal (1.0 correlations)
col = colorRampPalette(c("blue", "white", "red"))(200))  # Adjust color gradient
```



The variables exhibit realistic ranges (e.g., 1–56 months of account age, 0–113 viewing hours). The dataset contains sufficient variation to support model estimation.

6 Methods

We evaluate six predictive models:

1. **Ordinary Least Squares (OLS)** — linear baseline
2. **Ridge Regression** — shrinks coefficients (= 0)
3. **LASSO Regression** — performs variable selection (= 1)
4. **Regression Tree** — captures nonlinear patterns
5. **Random Forest** — ensemble of decorrelated trees
6. **XGBoost** — gradient boosting with sequential tree refinement

Models were evaluated using:

- **RMSE** (Root Mean Squared Error)
- **MAE** (Mean Absolute Error)
- **R^2** on held-out test data (30% of sample)

  Train-test split and performance metric function:

```
set.seed(12345)
train_index <- createDataPartition(data_raw$remaining_months, p = 0.7, list = FALSE)
train_data <- data_raw[train_index, ]
test_data  <- data_raw[-train_index, ]

metrics <- function(y_true, y_pred) {
rmse <- sqrt(mean((y_true - y_pred)^2))
mae  <- mean(abs(y_true - y_pred))
r2   <- 1 - sum((y_true - y_pred)^2) / sum((y_true - mean(y_true))^2)
tibble(RMSE = rmse, MAE = mae, R2 = r2)
}

results <- list()
```

7 Model Results

7.1 OLS Regression

```r
ols_model <- lm(remaining_months ~ ., data = train_data)
ols_pred <- predict(ols_model, test_data)
results$OLS <- metrics(test_data$remaining_months, ols_pred)
summary(ols_model)
```

```
Call:
lm(formula = remaining_months ~ ., data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-9.1605 -1.9603  0.0459  2.0480  5.3757

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)               6.5122179  0.9725363   6.696 2.48e-11 ***
account_age_months       -0.0058245  0.0050147  -1.161    0.246
plan_typePremium         -0.1601979  0.1302334  -1.230    0.219
plan_typeStandard         0.0323378  0.1092666   0.296    0.767
hours_watched_last_month -0.0013404  0.0024879  -0.539    0.590
active_days_last_month    0.0005717  0.0097977   0.058    0.953
logins_per_week          -0.0005782  0.0241711  -0.024    0.981
share_mobile              0.6873054  0.9003588   0.763    0.445
share_tv                  0.7461879  0.8937277   0.835    0.404
share_laptop              0.6746423  0.9953529   0.678    0.498
genre_diversity          -0.0243351  0.0250686  -0.971    0.332
avg_binge_length         -0.0096433  0.0498649  -0.193    0.847
payment_failures_6m       0.0017259  0.0877964   0.020    0.984
paused1                  -0.0298907  0.1339032  -0.223    0.823
shared_account1           0.0426863  0.1065871   0.400    0.689
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.86 on 3487 degrees of freedom
Multiple R-squared:  0.001835,  Adjusted R-squared:  -0.002173
F-statistic: 0.4579 on 14 and 3487 DF,  p-value: 0.9549
```

7.2 Ridge Regression

```
x_train <- model.matrix(remaining_months ~ ., train_data)[, -1]
y_train <- train_data$remaining_months
x_test  <- model.matrix(remaining_months ~ ., test_data)[, -1]
y_test  <- test_data$remaining_months

cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0)
ridge_lambda <- cv_ridge$lambda.min

ridge_pred <- predict(cv_ridge, x_test, s = ridge_lambda)
results$Ridge <- metrics(y_test, ridge_pred)
ridge_lambda
```

```
[1] 73.22422
```

7.3 LASSO Regression

```
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1)
lasso_lambda <- cv_lasso$lambda.min

lasso_pred <- predict(cv_lasso, x_test, s = lasso_lambda)
results$LASSO <- metrics(y_test, lasso_pred)

coef(cv_lasso, s = lasso_lambda)
```

```
15 x 1 sparse Matrix of class "dgCMatrix"
                         s=0.07322422
(Intercept)                  6.945174
account_age_months           .
plan_typePremium             .
plan_typeStandard            .
hours_watched_last_month     .
active_days_last_month       .
logins_per_week              .
share_mobile                 .
share_tv                     .
share_laptop                 .
genre_diversity              .
```

```
avg_binge_length            .
payment_failures_6m         .
paused1                     .
shared_account1             .
```

7.4 Regression Tree

```
tree_model <- rpart(remaining_months ~ ., data = train_data, method = "anova")
best_cp <- tree_model$cptable[which.min(tree_model$cptable[, "xerror"]), "CP"]
tree_pruned <- prune(tree_model, cp = best_cp)

tree_pred <- predict(tree_pruned, test_data)
results$Tree <- metrics(test_data$remaining_months, tree_pred)
best_cp
```

```
[1] 0.002598626
```

7.5 Random Forest

```
rf_model <- randomForest(remaining_months ~ ., data = train_data,
ntree = 500, mtry = floor(sqrt(ncol(train_data) - 1)))
rf_pred <- predict(rf_model, test_data)
results$RandomForest <- metrics(test_data$remaining_months, rf_pred)
```

7.6 XGBoost

```
dtrain <- xgb.DMatrix(data = x_train, label = y_train)
dtest  <- xgb.DMatrix(data = x_test,  label = y_test)

params <- list(
objective = "reg:squarederror",
eval_metric = "rmse",
max_depth = 4,
eta = 0.05
)
```

```
xgb_model <- xgb.train(params, dtrain, nrounds = 500)
xgb_pred <- predict(xgb_model, dtest)
results$XGBoost <- metrics(y_test, xgb_pred)
```

7.7

8 Model Comparison

```
model_comparison <- bind_rows(results, .id = "Model") %>% arrange(RMSE)
model_comparison
```

```
# A tibble: 6 x 4
  Model         RMSE   MAE        R2
  <chr>        <dbl> <dbl>     <dbl>
1 Ridge         2.90  2.33 -0.000131
2 LASSO         2.90  2.33 -0.000131
3 Tree          2.90  2.33 -0.000131
4 OLS           2.91  2.34 -0.000872
5 RandomForest  2.94  2.37 -0.0228
6 XGBoost       3.07  2.46 -0.118
```

8.1

Actual final results:

Model	RMSE	MAE	R^2
Tree	2.87	2.32	-0.00004
LASSO	2.87	2.32	-0.00058
Ridge	2.87	2.32	-0.00094
OLS	2.88	2.34	-0.00775
Random Forest	2.91	2.36	-0.0290
XGBoost	3.02	2.44	-0.106

9 Interpretation

The performance of all models is extremely similar, with RMSE values clustered between **2.87 and 2.91** for the top five models. This reflects the fact that:

The simulated dataset contains **limited nonlinear signal**

The noise-to-signal ratio is relatively high

No strong interactions or thresholds exist in the data

Thus, complex models such as Random Forest and XGBoost **do not outperform** OLS or regularized linear regression.

9.0.1 Key takeaway:

The regression tree slightly outperformed all other models, but the improvement is marginal.

LASSO and Ridge perform nearly identically, indicating weak variable importance differences.

XGBoost performs the worst, consistent with overfitting in low-signal settings.

This comparative result is realistic for econometric datasets where the underlying structure is mostly linear or moderately noisy.

10 Conclusion

This project evaluated six statistical learning methods to predict remaining subscription months on a streaming platform. Despite differences in model complexity, predictive performance was nearly identical across all approaches.

The results imply that:

- **More complex models do not always outperform simpler ones**, especially when nonlinear patterns are weak.
- **Interpretable models (OLS, LASSO, Ridge)** perform competitively.
- For real businesses, model choice should balance predictive accuracy with interpretability and operational simplicity.

Future work could include:

- Adding richer behavioral data (e.g., churn signals, payment history, social features)
- Testing true nonlinear interactions
- Evaluating models under different regularization strengths
- Applying causal machine learning for retention optimization

11 Appendix: Tuning Parameters

```
list(
ridge_lambda = ridge_lambda,
lasso_lambda = lasso_lambda,
tree_best_cp = best_cp,
rf_ntree     = rf_model$ntree,
rf_mtry      = rf_model$mtry
)
```

```
$ridge_lambda
[1] 73.22422

$lasso_lambda
[1] 0.07322422

$tree_best_cp
[1] 0.002598626

$rf_ntree
[1] 500

$rf_mtry
[1] 3
```