Understanding Telecom Churn Through Conditional Inference Trees

Vera Gak Anagrova

19 November, 2025

Introduction

Customer churn is a major challenge for telecom providers, as keeping existing customers is much more cost-effective than acquiring new ones. Identifying which customers are likely to leave helps companies target retention efforts and allocate resources efficiently. This report analyzes data from 2,000 telecom customers to identify the key factors driving churn, including service features, billing information, and customer tenure. To address this question, we use a conditional inference tree, a statistical method that segments customers into interpretable groups based on meaningful patterns in the data. Unlike traditional decision trees, this approach uses statistical tests to determine whether splits are justified, preventing unnecessary branches and reducing overfitting. The goal is to uncover the most important predictors of churn and evaluate how well the model performs on new, unseen customers.

Methodology

Before modelling, the dataset was checked for missing values, inconsistencies, and incorrect data types. All categorical variables were already stored as factors, ensuring correct handling by the tree-based algorithm. A correlation check showed a strong relationship between tenure and TotalCharges ($r \approx 0.83$)), which is expected since total charges grow with time; both variables were kept because conditional inference trees are not sensitive to multicollinearity. The data was then randomly divided into a 70% training set and a 30% test set. This split provides enough data for model learning while reserving a sufficiently large portion for an unbiased evaluation of out-of-sample performance.

To prepare the model, several conditional inference trees were first fitted to the training data using different values of the significance level α, which determines how strong the statistical evidence must be for a split to occur (implemented in R as mincriterion = 1 - α). Smaller α values produce simpler trees, whereas larger values allow more complex structures. Testing multiple α levels ensured that the final model achieved an appropriate balance between simplicity and flexibility. After selecting the preferred value, the final Conditional Inference Tree (CTree) model was trained on the full training set. Unlike classical Classification and Regression Trees (CART) — which rely on impurity measures such as Gini or entropy, tend to grow excessively large, and require a separate pruning step — CTrees base their splits on formal statistical hypothesis tests. This design avoids variable-selection bias, mitigates overfitting, and removes the need for pruning altogether. During the fitting process, the algorithm evaluates all predictors for their association with churn using permutation-based tests: if none are significant, the node becomes terminal; if at least one predictor is significant, the algorithm selects the best split and repeats the procedure for the resulting subgroups. Once trained, the final model was applied to the test set to generate out-of-sample predictions, as discussed in the Results section.

After training the conditional inference tree, its predictive performance was evaluated on the test set using a confusion matrix. From this, several metrics were computed, including accuracy, sensitivity, specificity, and Cohen's Kappa. Because the dataset is imbalanced, balanced accuracy was also reported to provide a fairer assessment across both classes. A heatmap of the confusion matrix was included to visually summarize classification performance.

Results

To determine the appropriate model complexity, α values ranging from 0.01 to 0.30 were evaluated. The tree continued to grow until $\alpha = 0.20$, after which increases in α (e.g., 0.25 and 0.30) produced identical models with the

same number of terminal nodes (20) and the same out-of-sample accuracy (78.3%). This suggests that the model becomes saturated at $\alpha = 0.20$, meaning that all statistically significant structure in the data has been captured and additional splits are no longer supported. The full comparison of α values is presented in Table 1 below.

Table 1: **CIT Performance for Different Alpha Levels**

Alpha (α)	Min Criterion	Terminal Nodes	Train Accuracy	Test Accuracy
0.01	0.99	8	0.793	0.770
0.05	0.95	10	0.798	0.752
0.10	0.90	15	0.798	0.752
0.20	0.80	20	0.816	0.783
0.25	0.75	20	0.816	0.783
0.30	0.70	20	0.816	0.783

Smaller values, such as $\alpha = 0.05$ and $\alpha = 0.10$, produced simpler trees (10–15 terminal nodes) but with noticeably lower test accuracy (75.2%). In the context of churn prediction, a difference of more than three percentage points is meaningful, as it corresponds to a substantially higher number of correctly classified customers in a dataset of 2,000 observations. Therefore, $\alpha = 0.20$ was selected as the optimal value, as it provides the best balance between model complexity and predictive performance.

The conditional inference tree shows that Contract type is the strongest predictor of churn (see Appendix A for the full tree). The first split separates customers with Month-to-month contracts from those with One-year or Two-year contracts, with the former group showing substantially higher churn rates. Among month-to-month customers, churn risk is further driven by InternetService, tenure, and security-related services such as OnlineSecurity and TechSupport; customers with fiber-optic internet, short tenure, and no security add-ons form the highest-risk segment. In contrast, customers with long-term contracts exhibit much lower churn overall. Within this group, InternetService, tenure, and in some branches Partner status still play a secondary role, with slightly higher churn among customers with very short tenure or no partner, though the differences are much smaller than in the month-to-month segment.

Model performance was evaluated using a confusion matrix (Appendix A) and the summary metrics reported in Table 2. The model achieved an accuracy of 0.783, with a high specificity of 0.888 and a moderate sensitivity of 0.497. This indicates that the classifier correctly identifies most non-churners (390 true negatives), but detects only about half of the customers who actually churn (80 true positives versus 81 missed). This imbalance is expected due to the predominance of non-churners in the dataset. For this reason, balanced accuracy was also considered as a more informative metric. The balanced accuracy of 0.693 confirms that the model performs well above chance for both classes despite the unequal class distribution.

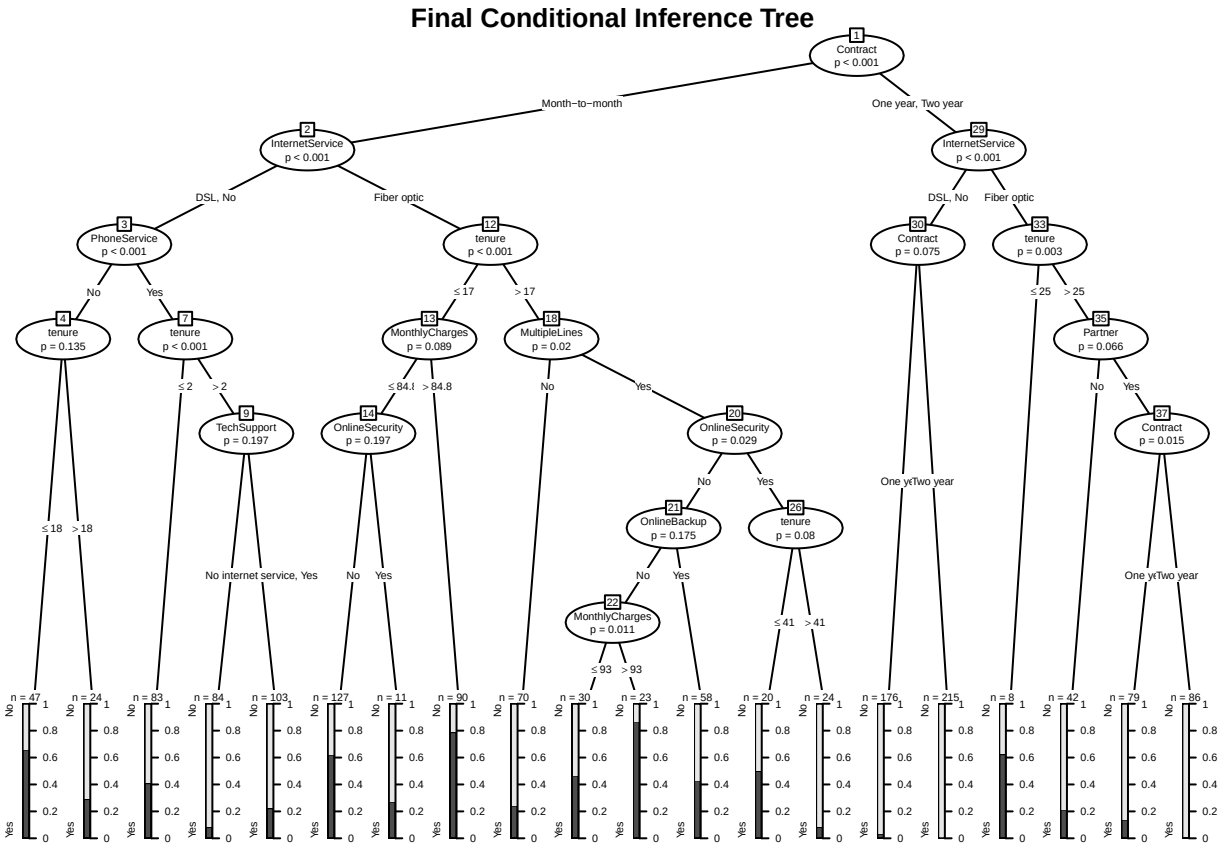
Table 2: **Performance Metrics**

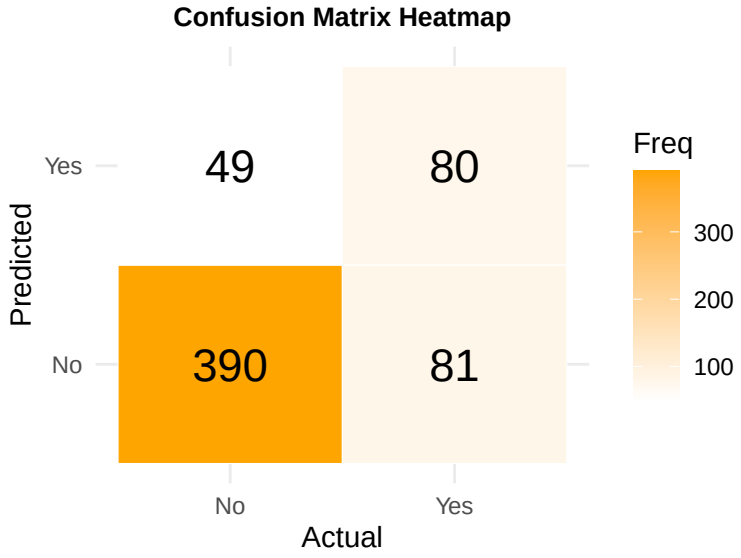
	Train.Accuracy	Test.Accuracy	Kappa	Sensitivity	Specificity	Balanced.Accuracy
Accuracy	0.816	0.783	0.411	0.497	0.888	0.693

Conclusion

This study applied conditional inference trees to predict customer churn using telecom customer data. By tuning the α parameter, the model was constrained to grow only as far as justified by statistically significant splits. The optimal model was obtained at $\alpha = 0.20$, which maximized predictive accuracy without overfitting. The resulting tree reveals clear and intuitive churn drivers: month-to-month contracts, fiber-optic internet, low tenure, and the absence of security-related services are the strongest indicators of churn risk. The model achieved good out-of-sample performance, with an accuracy of 78.3% and strong specificity. While sensitivity is moderate, the model still identifies meaningful high-risk customer profiles, which are directly interpretable from the structure of the tree. Overall, the conditional inference tree provides both competitive predictive performance and easily interpretable insights, making it a suitable tool for understanding churn behaviour and informing potential retention strategies.

Appendix A


Final Conditional Inference Tree
1
Contract
p < 0.001
Month−to−month
One year, Two year
2
InternetService
p < 0.001
29
InternetService
p < 0.001
DSL, No
Fiber optic
DSL, No
Fiber optic
3
PhoneService
p < 0.001
12
tenure
p < 0.001
30
Contract
p = 0.075
33
tenure
p = 0.003
No
Yes
≤ 17
> 17
≤ 25
> 25
4
tenure
p = 0.135
7
tenure
p < 0.001
13
MonthlyCharges
p = 0.089
18
MultipleLines
p = 0.02
35
Partner
p = 0.066
≤ 2
> 2
≤ 84.8
> 84.8
No
Yes
No
Yes
9
TechSupport
p = 0.197
14
OnlineSecurity
p = 0.197
20
OnlineSecurity
p = 0.029
37
Contract
p = 0.015
≤ 18
> 18
No internet service, Yes
No
Yes
No
Yes
21
OnlineBackup
p = 0.175
26
tenure
p = 0.08
22
MonthlyCharges
p = 0.011
≤ 41
> 41
≤ 93
> 93
n = 47
n = 24
n = 83
n = 84
n = 103
n = 127
n = 11
n = 90
n = 70
n = 30
n = 23
n = 58
n = 20
n = 24
n = 176
n = 215
n = 8
n = 42
n = 79
n = 86


Confusion Matrix Heatmap
Yes
No
49
80
390
81
Predicted
Actual
Freq
300
200
100

Appendix B

```
################################################################
# Load Libraries
################################################################
library(tidyverse)
#install.packages("partykit")
library(partykit)   # for ctree
library(caret)      # for confusionMatrix
library(knitr)
library(kableExtra)
library(ggplot2)
library(car)

################################################################
# Import and Clean Data
################################################################
load("Churn.2526.RData")
list.files()

#Checking data
str(data)
head(data)
summary(data)
anyNA(data)

# Select only numeric predictors
numeric_vars <- data[sapply(data, is.numeric)]
# Compute correlation matrix
cor_matrix <- cor(numeric_vars, use = "everything")
cor_matrix

################################################################
# Train-Test Split
################################################################
n <- nrow(data) # number of rows
set.seed(123) # make the random split reproducible
train_idx <- sample(seq_len(n), size = 0.7 * n) # pick 70% indices at random

train <- data[train_idx, ] # training data (what the model "sees")
test  <- data[-train_idx,] # test data (held-out, for evaluation)

nrow(train); nrow(test)

################################################################
# Tune  for the Conditional Inference Tree
################################################################
alphas <- c(0.01, 0.05, 0.10, 0.20, 0.25, 0.30)

#empty table for results
results <- data.frame(
  alpha     = alphas,
  mincrit   = 1 - alphas, # mincriterion (0-1), required evidence for a split
  nodes     = NA, #number of terminal nodes (final group of costumers created by the splits)
  train_acc = NA, #accuracy on trainning set
  test_acc  = NA #accuracy on the testing set
```

```
)

for (i in seq_along(alphas)) {
  a <- alphas[i]
  ctrl <- ctree_control(mincriterion = 1 - a) #This tells the tree how strict to be.
  #fit a CIT predicting Churn using all other variables (.) using the training data and the current
   fit  <- ctree(Churn ~ ., data = train, control = ctrl)
  # number of terminal nodes
  results$nodes[i] <- width(fit)
  # training accuracy
  pred_train <- predict(fit)
  #calculates training accuracy (how many predictions match the true Churn values)
  results$train_acc[i] <- mean(pred_train == train$Churn)
  # test accuracy
  pred_test <- predict(fit, newdata = test)
  #calculates testing accuracy (how many predictions match the true Churn values)
  results$test_acc[i]  <- mean(pred_test == test$Churn)
}

results %>%
  kable(format = "latex", digits = 3,booktabs = TRUE,
    col.names = c(
      "Alpha ($\\alpha$)",
      "Min Criterion",
      "Terminal Nodes",
      "Train Accuracy",
      "Test Accuracy"
    ),
    caption  = "\\textbf{CIT Performance for Different Alpha Levels}",
    escape   = FALSE
  ) %>%
  kable_styling(latex_options = c("striped", "hold_position"),full_width = FALSE,
    position = "center")

################################################################
# Fit Final Conditional Inference Tree
################################################################
alpha_final <- 0.20
ctrl_final  <- ctree_control(mincriterion = 1 - alpha_final)
tree_final <- ctree(Churn ~ ., data = train, control = ctrl_final)

# Plot the tree
plot(tree_final,gp = grid::gpar(fontsize = 4),tp_args = list(id = FALSE))

#variable importance
varimp(tree_final)

# Get variable importance
vi <- varimp(tree_final)

# Convert to 1-row data frame (variables as columns)
vi_row <- as.data.frame(t(vi))   # transpose

# Pretty table
kable(vi_row, digits = 3, caption = "Variable Importance (CTree Model)") %>%
  kable_styling(full_width = FALSE, position = "center")
```

```r
############################################################
# Evaluate out-of-sample performance (test set)
############################################################
pred_test <- predict(tree_final, newdata = test)

# Confusion matrix
conf_mat <- table(Predicted = pred_test, Actual = test$Churn)
cm_df <- as.data.frame(conf_mat)

ggplot(cm_df, aes(x = Actual, y = Predicted, fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), size = 6) +
  scale_fill_gradient(low = "white", high = "orange") +
  theme_minimal() +labs(title = "Confusion Matrix Heatmap") +
  theme(plot.title = element_text(size = 10, face = "bold", hjust = 0.5))

# Overall accuracy
test_accuracy <- mean(pred_test == test$Churn)
pred_train <- predict(tree_final)
train_accuracy <- mean(pred_train == train$Churn)

cm_raw <- confusionMatrix(pred_test, test$Churn, positive = "Yes")
cm_raw

cm <- cm_raw$table
stats <- data.frame(
  `Train Accuracy` = train_accuracy,
  `Test Accuracy` = cm_raw$overall["Accuracy"],
  Kappa = cm_raw$overall["Kappa"],
  Sensitivity = cm_raw$byClass["Sensitivity"],
  Specificity = cm_raw$byClass["Specificity"],
  `Balanced Accuracy` = cm_raw$byClass["Balanced Accuracy"]
)

kable(stats, digits = 3, caption = "\\textbf{Performance Metrics}") %>%
  kable_styling(full_width = FALSE, position = "center")
```

References

GeeksforGeeks. 2025. "Conditional Inference Trees in r Programming." 2025. https://www.geeksforgeeks.org/r-language/conditional-inference-trees-in-r-programming/.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2023. *An Introduction to Statistical Learning: With Applications in r*. 2nd ed. New York: Springer.

Schweinberger, Martin. 2023a. *An Introduction to Conditional Inference Trees in r*. Bonn, Germany: Rheinische Friedrich Wilhelms-Universität Bonn. https://martinschweinberger.github.io/TreesUBonn/.

———. 2023b. *Tree-Based Models in r*. Brisbane, Australia: Language Technology; Data Analysis Laboratory (LADAL), The University of Queensland. https://ladal.edu.au/tutorials/tree/tree.html.

———. 2023c. "Workshop: An Introduction to Conditional Inference Trees in r." YouTube video. https://www.youtube.com/watch?v=mc8EyrR7kio.