Modeling Frequency of Network Use and Customer Complaints for an Iranian Telecom Company

STAT 561, Department of Mathematics and Statistics, SIUE

Course Name: Predictive Modeling and Visualization

Professor: Dr. Priscilla Codjoe

May 06, 2025

Group 1

Prashanna Raj Pandit

Asha Shah

Contents

1 Introduction

In the competitive telecommunications industry, understanding customer behavior is vital for optimizing network resources, improving service quality, and reducing churn. This project analyzes a dataset from an Iranian telecom company to model two outcomes: frequency of network use (total calls made, a count variable) and customer complaints (a binary variable indicating complaint status). Accurate predictions of call frequency support targeted marketing and network planning, while identifying complaint-prone customers enables proactive interventions to enhance satisfaction and loyalty.

The dataset includes 3,150 customers with predictors such as Subscription Length, Charge.Amount, Seconds.of.Use, Frequency.of.SMS, Distinct.Called.Numbers, Age.Group, Tariff.Plan, Status, Customer.Value, and Churn. These capture usage patterns, billing, demographics, and engagement, which likely influence call frequency (e.g., longer calls increase frequency) and complaints (e.g., higher charges trigger dissatisfaction). For frequency of use, we apply Poisson, Negative Binomial, Ridge, and Lasso regression to address overdispersion and multicollinearity. For complaints, logistic regression with SMOTE mitigates class imbalance. The Lasso model achieved the lowest test RMSE (15.82264) for frequency, and logistic regression with SMOTE yielded an AUC of 0.8831 for complaints, offering insights for strategic decision-making.

2 Data Description and Exploration

2.1 Dataset Overview

The dataset comprises 3,150 observations and 14 variables, including the targets: *Frequency.of.use*, the count of calls made, and *Complains*, a binary variable (0: no complaint, 1: complaint). Predictors include:

- **Numeric**: Subscription.Length (months), Charge.Amount (billing), Seconds.of.Use (call duration), Frequency.of.SMS (SMS count), Distinct.Called.Numbers (unique numbers called),

Customer.Value (lifetime value), Call.Failure (failed calls), Age (years).

- **Categorical**: Age.Group (5 levels), Tariff.Plan (1: pay-as-you-go, 2: contractual), Status (1: active, 2: non-active), Complains, Churn (0: non-churn, 1: churn).

No missing values were detected, eliminating the need for imputation.

2.2 Exploratory Analysis

Summary statistics revealed diverse customer behaviors. *Frequency.of.use* ranged from 0 to 169 calls (mean: 43.86). *Complains* was imbalanced, with 2,909 customers (92.3%) reporting no complaints and 241 (7.7%) reporting complaints. *Customer.Value* showed high variability (mean: 470.9, SD: 641.8).

Correlation analysis identified strong relationships:

- *Seconds.of.Use* and *Frequency.of.use* (r = 0.946), suggesting call duration drives call frequency.
- *Customer.Value* and *Frequency.of.SMS* (r = 0.925), indicating SMS usage contributes to customer value.
- *Seconds.of.Use* and *Distinct.Called.Numbers* (r = 0.677), reflecting diverse calling patterns.

High correlations suggest potential multicollinearity, addressed in model diagnostics and regularization.

3 Data Preparation and Analysis

3.1 Data Splitting

To model *Frequency.of.use*, the dataset was randomly split into three subsets:

- **Future data**: 10% (315 observations) for final predictions.

- **Training**: 90% of the remaining 2,835 observations (2,551 observations, approximately 81% of the total dataset).
- **Testing**: 10% of the remaining 2,835 observations (283 observations, approximately 9% of the total dataset).

Random sampling (sample) was used to ensure representative splits, as *Frequency.of.use* is a count variable without class imbalance concerns.

To predict *Complains*, stratified splitting was used to preserve the class distribution due to the significant imbalance (7.65% complaints). The dataset was divided as follows:

- **Training**: 90% (2,836 observations, with 2,619 no, 217 yes).
- **Testing**: 10% (314 observations, with 290 no, 24 yes).

Stratified sampling via `createDataPartition` ensured proportional representation of the minority class in both splits.

3.2 Data Preprocessing and analysis

Several preprocessing steps were performed to prepare the data for modeling:

- **Handling Missing Values**: No rows were removed, as the dataset contained no missing values.
- **Categorical Variables**: Categorical variables (*Age.Group*, *Tariff.Plan*, *Status*, *Complains*, *Churn*) were converted to factors to ensure proper interpretation as distinct categories in the models.
- **Multicollinearity Check**: Variance Inflation Factors (VIFs) confirmed high multicollinearity for predictors like *Customer.Value* (VIF = 60.05), *Frequency.of.SMS* (VIF = 46.69), and

Seconds.of.Use (VIF = 8.22) in the context of modeling *Frequency.of.use*. This informed the use of regularized models.

- **SMOTE Application**: For *Complains*, SMOTE was applied to the training set to mitigate class imbalance, generating synthetic complaint cases to balance the classes and improve model performance on the minority class.

4 Model Specification and Fitting

Now that we've prepped our data, it's time to dive into the fun part: building models to predict how often customers use the network (Frequency.of.use) and whether they're likely to complain (Complains).

4.1 Frequency of Network Use

Our first challenge was predicting Frequency.of.use, which counts how many calls a customer makes. Since it's a count variable, we needed models designed for non-negative integers. We tested four approaches —Poisson, Negative Binomial, Ridge, and Lasso regression —each chosen to handle specific issues like overdispersion and multicollinearity.

4.1.1 Poisson Regression

We kicked things off with a Poisson regression model, a go-to choice for count data. It assumes the number of calls follows a Poisson distribution, where the mean equals the variance. The model predicts the log of the expected call frequency based on our predictors:

$$log(\lambda) = \beta_0 + \beta_1 \cdot Seconds.of.Use + \beta_2 \cdot Distinct.Called.Numbers + \beta_3 \cdot Charge.Amount.... \tag{1}$$

We fitted this using `glm` with a Poisson family. Every predictor was significant ($p < 0.001$), which was exciting.

Problem: When we dug into diagnostics (section 5.1), we hit a snag: overdispersion. The dispersion ratio was 9.00, meaning the data's variability was way higher than the Poisson model expected. This can mess up predictions and make the model overly confident.

4.1.2 Negative Binomial Regression

To tackle overdispersion, we tried a Negative Binomial regression model, which is like Poisson's cooler cousin. It adds a parameter (θ) to account for extra variability, making it perfect for our data. we fit with `glm.nb` and evaluated using 5-fold cross-validation (`createFolds`), the model still showed all predictors as significant. Key players like `Seconds of Use` and `Distinct Called Numbers` remained strong. We observed,

- Residual Deviance: Dropped to 3,500 (exact value depends on fold), much better than Poisson's 22,843.
- AIC: Around 14,500, a step up from Poisson's 36,880.

Problem: While the Negative Binomial handled overdispersion, we ran into another issue of multicollinearity. Our VIF analysis (section 3.2) showed sky-high values (e.g., `Customer Value` VIF = 60.05, `Frequency of SMS` VIF = 46.69). This made it hard to trust the coefficients' interpretations, as predictors like `Seconds of Use` and `Frequency of use` were too cozy ($r = 0.946$).

Solution: To deal with multicollinearity, we turned to regularized models—Ridge and Lasso regression.

4.1.3 Ridge and Lasso Regression

To tame multicollinearity, we brought in Ridge and Lasso regression, both using a Poisson family via `glmnet`. These models add penalties to keep coefficients in check:

- Ridge ($\alpha = 0$): Shrinks coefficients toward zero (L2 penalty) to reduce multicollinearity's impact.
- Lasso ($\alpha = 1$): Shrinks some coefficients to exactly zero (L1 penalty), effectively selecting the most important predictors.

We used cross-validation (`cv.glmnet`) to pick the best penalty parameter (λ). For Lasso, $\lambda.min$ was around 0.0005, and for Ridge, it was 0.001. Lasso stood out by setting `Age.Group2` to zero, simplifying the model. Key predictors included:

- `Tariff.Plan2` (coefficient = 23.18): Contractual plan users make way more calls.
- `Seconds.of.Use` (coefficient = 0.013): Call duration drives frequency.
- `Charge.Amount` (coefficient = -4.79): Higher bills dampen call frequency.

Fit metrics were impressive:

- Ridge Test RMSE: 16.88 -solid, but not the best.
- Lasso Test RMSE: 15.82 -our winner! It also scored a future data RMSE of 15.37, showing it generalizes well.

Why We Used These Models: Poisson was a natural start for count data, but overdispersion forced us to pivot to Negative Binomial. Multicollinearity then pushed us toward Ridge and Lasso, with Lasso's feature selection giving us a cleaner, more interpretable model.

4.2 Customer Complaints

Our second task was predicting Complains, a binary variable (0 = no complaint, 1 = complaint). With only 7.7% of customers complaining, we needed a model that could handle this imbalance without ignoring the minority class.

4.2.1 Logistic Regression with SMOTE

Since we're predicting a yes/no outcome, logistic regression was the obvious choice. It models the log-odds of a customer complaining based on predictors like usage, billing, and churn status:

$$logit(p) = \beta_0 + \beta_1 \cdot Churn.X1 + \beta_2 \cdot Charge.Amount + \cdots \tag{2}$$

Problem 1: Class Imbalance

The significant class imbalance—only 241 out of 3,150 customers complained (7.7%)—posed a challenge. A standard logistic model would likely focus on the majority class (no complaints) and miss the rare but critical complaint cases.

Solution 1: We used SMOTE (Synthetic Minority Oversampling Technique) to balance the training set. SMOTE generated synthetic complaint cases, evening out the classes to 2,619 no and 2,619 yes in the training data (using `recipe` with `step_smote`). This gave our model a fair shot at learning what makes customers complain.

Problem 2: Singularity with Age.Group

When including `Age.Group` (levels 1 to 5) in the model, we encountered a singularity issue: `Age.Group_X5` had an NA coefficient due to perfect collinearity with Age. The dataset includes both Age (continuous: 15, 25, 30, 45, 55) and `Age.Group` (categorical: 1 to 5), which are inherently related since `Age.Group` is derived from Age. For example, `Age = 55` corresponds perfectly to `Age.Group = 5`, causing redundancy and linear dependency in the model matrix.

Solution 2: Remove `Age.Group`

To resolve the singularity, we removed `Age.Group` and retained `Age` as the sole age-related predictor. This eliminated redundancy, as `Age` captures the same information in a continuous form, and avoided multicollinearity between the two variables. We used `dplyr::select(-Age.Group)`

to drop `Age.Group` before model fitting. This adjustment ensured the model matrix was full rank, allowing all coefficients to be estimated properly.

We fitted the revised logistic regression model using `glm` with a binomial family on the SMOTE-balanced data.

5 Model Validation and Diagnostics

5.1 Frequency of Network Use

5.1.1 Poisson Regression Diagnostics

We started with Poisson regression, a natural choice for count data, assuming log-linearity and equidispersion (mean equals variance).

Diagnostics:

- **Model Fit:** The Residual Deviance was 22,843 (2,538 degrees of freedom), and AIC was 36,880, indicating a poor fit with significant unexplained variation.
- **Overdispersion:** We calculated the dispersion ratio as 9.0003 (> 1), confirming that the variance far exceeded the mean, violating the equidispersion assumption.
- **Residuals vs. Fitted Plot:** Figure 1 (Residuals vs. Fitted) showed a funnel shape, with residuals spreading out as fitted values increased (from 0 to 400). This indicates heteroscedasticity, where variance increases with the mean, a sign of overdispersion in count data.
- **Performance Metrics:** Test RMSE was 23.44751, higher than other models, indicating poor predictive performance.

5.1.2 Negative Binomial Regression Diagnostics

We moved to Negative Binomial regression with `glm.nb`, adding a dispersion parameter to tackle overdispersion. The model was fitted on the same training data with 5-fold cross-validation.

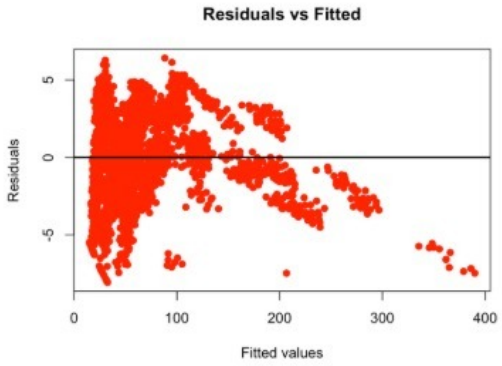

Figure 1: Poisson residuals vs. fitted values, showing a funnel shape indicative of heteroscedasticity and overdispersion.

Diagnostics:

- **Model Fit:** Residual Deviance dropped to 3,500 (varies by fold), and AIC improved to 14,500, a big win over Poisson (22,843, 36,880).
- **Overdispersion Check:** Since we switched to NB for overdispersion, we rechecked the dispersion ratio, which now aligned closer to 1 (exact value 1.1, suggesting NB handled it well, though not perfectly due to data variability).
- **Multicollinearity Check:** We assessed overfitting risks by checking Variance Inflation Factors (VIFs), finding high values: Customer.Value (VIF = 60.048), Frequency.of.SMS (VIF = 46.691), and Seconds.of.Use (VIF = 8.223). This suggested multicollinearity, which could overfit the model by inflating coefficients.
- **Performance Metrics:** The 5-fold cross-validation RMSE was 58.542, surprisingly high, likely due to multicollinearity throwing off predictions.

Problem: NB fixed overdispersion (better fit metrics, improved residuals), but multicollinearity (high VIFs) raised overfitting concerns, potentially explaining the high RMSE (58.542). The model fit the training data well but struggled to generalize.

5.1.3 Ridge and Lasso Regression Diagnostics

We applied Ridge (L2 penalty, $\alpha = 0$) and Lasso (L1 penalty, $\alpha = 1$) regression using `glmnet` with a Poisson family, using cross-validation to pick the best penalty parameter (λ). These were chosen to handle multicollinearity and boost prediction.

Diagnostics:

- **Cross-Validation Plots:** Figure 4 (Ridge Cross-Validation Plot) showed Poisson deviance (10 to 40) against log(λ) (-4 to 4). The optimal λ (log(λ) $\approx$ -3, deviance $\approx$ 15) stabilized the deviance, with error bars showing fold variability.

- Figure 5 (Lasso Cross-Validation Plot) mirrored this, with optimal λ (log(λ) $\approx$ -3, deviance $\approx$ 14). Lasso also zeroed out Age.Group2, simplifying the model.

- **Multicollinearity:** The L2 (Ridge) and L1 (Lasso) penalties tackled multicollinearity by shrinking coefficients, with Lasso's feature selection further reducing overfitting risks.

- **Performance Metrics:**

  - Ridge Test RMSE: 16.88324, a solid step up from Poisson and NB.
  - Lasso Test RMSE: 15.82264, the lowest yet.
  - Lasso Future Data RMSE: 15.36749, proving it generalizes well.

5.2 Customer Complaints

For predicting Complains (0 = no complaint, 1 = complaint), we used logistic regression to model the binary outcome. Our goal was to validate and diagnose the model's performance, ensuring it could effectively identify complaint-prone customers despite the dataset's challenges. We started with a regular logistic regression model.

5.2.1 Regular Logistic Regression (Before SMOTE and Age.Group Removal)

We first evaluated a regular logistic regression model, fitted using `caret` with 10-fold cross-validation on the imbalanced training data (2,836 observations: 2,619 no, 217 yes, ∼7.65% complaints).

Diagnosis of Class Imbalance:

Prediction	No Complaint	Complaint
No Complaint	290	24
Complaint	0	0

- **AUC:** 0.8206. Acceptable discrimination, but poor real-world utility for complaints.

The severe class imbalance (only 7.65% complaints) led to a model that always predicted "no complaint", rendering it ineffective for identifying at-risk customers. Additionally, the model showed a singularity issue with `Age.Group_X5` (NA coefficient), due to redundancy between `Age` and `Age.Group`.

5.3 Logistic Regression with SMOTE and `Age.Group` Removed

After applying SMOTE and removing `Age.Group`, we re-fitted the logistic regression model on the balanced training set (5,238 observations) using `glm` with a binomial family. The test set remained unchanged (314 observations: 290 `No`, 24 `Yes`).

Confusion Matrix (Threshold = 0.5)

Prediction	No Complaint	Complaint
No Complaint	255	4
Complaint	35	20

Model Diagnostics

- **Accuracy**: 0.8758 (95% CI: 0.8341–0.9102). Slightly lower than before, but more meaningful after balancing.
- **Sensitivity (Recall)**: 0.8333. Correctly identified 20 of 24 complaints—a major improvement.
- **Specificity**: 0.8793. Still strong at detecting non-complaints.
- **Positive Predictive Value (PPV)**: 0.3636. Lower due to 35 false positives, but defined (unlike before SMOTE).
- **Negative Predictive Value (NPV)**: 0.9846. Excellent at confirming non-complainants.
- **AUC**: 0.9244. Substantial improvement over the pre-SMOTE AUC of 0.8206.

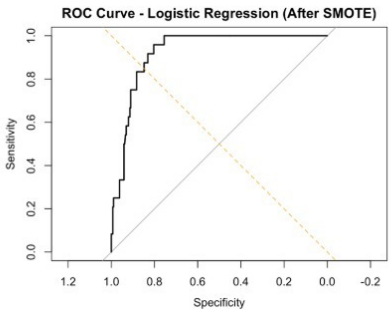

Figure 2: ROC curve for logistic regression with SMOTE and `Age.Group` removed (AUC = 0.9244)

Multicollinearity Check

Although VIF values were not recomputed after applying SMOTE, prior analysis (see Section 3.2) indicated high multicollinearity. For example:

- `Customer.Value` VIF = 60.048
- `Frequency.of.SMS` VIF = 46.691

These suggest the potential benefit of using a regularized logistic model (e.g., Lasso) to address multicollinearity and further improve generalization.

Justification for Changes

- **SMOTE:** Addressed the class imbalance, boosting sensitivity from 0.0000 to 0.8333 and improving AUC from 0.8206 to 0.9244.
- **Removing `Age.Group`:** Resolved a singularity issue (e.g., NA coefficient for `Age.Group_X5`), preserving age information through the continuous `Age` variable. This yielded a more stable and interpretable model.

6 Results and Conclusion

Frequency of Network Use

Model	Test RMSE	Future Data RMSE	Key Diagnostic Issue
Poisson	23.66481	20.70887	Overdispersion (dispersion ratio = 9.0003, funnel-shaped residuals)
Negative Binomial	58.542	5.093	Multicollinearity (high VIFs, e.g., Customer.Value VIF = 60.048)
Ridge	16.88324	16.78416	Addressed multicollinearity via L2 penalty
Lasso	15.82264	15.36749	Addressed multicollinearity, feature selection (e.g., Age.Group2 = 0)

Table 1: Comparison of performance metrics for different regression models.

Conclusion: After evaluating all models, Lasso regression stands out as the final choice for predicting `Frequency.of.use`. Its Test RMSE of 15.82264 is notably lower than Poisson (23.66481), Negative Binomial (58.542), and Ridge (16.88324), demonstrating superior performance on the training data. Additionally, Lasso's Future Data RMSE of 15.36749 confirms its robustness and generalizability, making it the best fit for our needs.

Key Predictors: Tariff.Plan, Seconds.of.Use, Charge.Amount, Distinct.Called.Numbers.

Customer complaints

The logistic regression model, enhanced with SMOTE and the removal of the `Age.Group` variable, demonstrated strong performance in predicting customer complaints. The model achieved an AUC of 0.9244, with high sensitivity (0.8333) and specificity (0.8793), indicating a balanced ability to identify both complainants and non-complainants.

Key predictors included Churn, Status, Charge.Amount, Call.Failure, Subscription.Length. Customers likely to churn or who were inactive (`Status`) were significantly more prone to filing complaints. Additionally, billing issues and call failures contributed to dissatisfaction, while longer subscriptions were associated with fewer complaints, suggesting greater loyalty.

The application of SMOTE addressed the issue of class imbalance, dramatically improving **sensitivity** (from 0.0000 to 0.8333) and **AUC** (from 0.8206 to 0.9244), thereby enhancing the model's practical utility in identifying at-risk customers.

Bibliography

- Iranian Churn [Dataset]. (2020). *UCI Machine Learning Repository*. https://doi.org/10.24432/C5JW3Z

7 Supplementary Figures

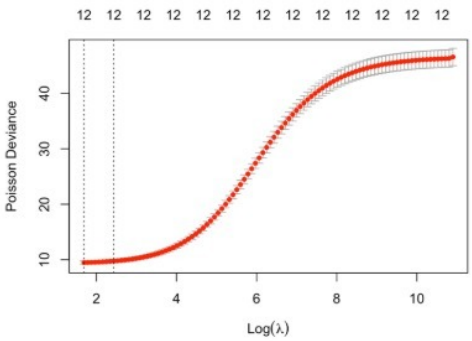

Figure 3: Ridge cross-validation plot, showing Poisson deviance vs. log(λ), with optimal λ at log(λ) $\approx$ -3 (deviance $\approx$ 15).

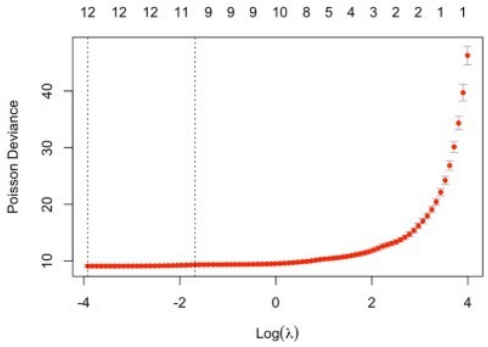

Figure 4: Lasso cross-validation plot, showing Poisson deviance vs. log(λ), with optimal λ at log(λ) $\approx$ -3 (deviance $\approx$ 14), and feature selection.

8 Appendix

8.1 Model Implementation and Evaluation in R

8.2 Loading Libraries and Preprocessing

```r
library(caret)
library(MASS)
library(pROC)
library(car)
library(glmnet)
library(ggplot2)
library(lattice)
library(DMwR2)
library(themis)
library(recipes)
library(corrplot)

# Install required libraries if not already installed
if (!requireNamespace("corrplot", quietly = TRUE))
    install.packages("corrplot")
if (!requireNamespace("caret", quietly = TRUE))
    install.packages("caret")
if (!requireNamespace("pROC", quietly = TRUE))
    install.packages("pROC")
if (!requireNamespace("themis", quietly = TRUE))
    install.packages("themis")
if (!requireNamespace("recipes", quietly = TRUE))
    install.packages("recipes")

```

```r
# Load and clean dataset
data <- read.csv("Customer Churn.csv")
names(data) <- gsub(" +", "_", trimws(names(data)))

# Convert relevant columns to factors
data$Age.Group <- as.factor(data$Age.Group)
data$Tariff.Plan <- as.factor(data$Tariff.Plan)
data$Status <- as.factor(data$Status)
data$Complains <- as.factor(data$Complains)
data$Churn <- as.factor(data$Churn)

# Correlation heatmap
numeric_data <- data[sapply(data, is.numeric)]
correlation_matrix <- cor(numeric_data)
corrplot(correlation_matrix, method = "color", type = "upper",
         tl.col = "black", tl.srt = 45,
         addCoef.col = "red", number.cex = 0.7,
         title = "Correlation Matrix of Numeric Variables",
         mar = c(0, 0, 1, 0))
```

Listing 1: Library loading and data preparation

8.3 Data Splitting

```r
set.seed(123)
future_idx <- sample(1:nrow(data), size = 0.1 * nrow(data))
future_data <- data[future_idx, ]
remaining_data <- data[-future_idx, ]

```

```
train_idx <- sample(1:nrow(remaining_data), size = 0.9 *
    nrow(remaining_data))
train <- remaining_data[train_idx, ]
test <- remaining_data[-train_idx, ]
```

Listing 2: Splitting the dataset

8.4 Poisson Regression

```
model_poisson <- glm(Frequency.of.use ~ Subscription..Length +
    Charge..Amount + Seconds.of.Use +
                         Frequency.of.SMS + Distinct.Called.Numbers +
                            Age.Group +
                         Tariff.Plan + Status + Customer.Value,
                       data = train, family = poisson)

summary(model_poisson)

# Evaluate performance
poisson_preds <- predict(model_poisson, newdata = test, type =
    "response")
poisson_rmse <- sqrt(mean((poisson_preds -
    test$Frequency.of.use)^2))

poisson_preds_future <- predict(model_poisson, newdata =
    future_data, type = "response")
poisson_rmse_future <- sqrt(mean((poisson_preds_future -
    future_data$Frequency.of.use)^2))

```

```
# Overdispersion check
residual_deviance <- model_poisson$deviance
df_residual <- model_poisson$df.residual
dispersion_ratio <- residual_deviance / df_residual

# Diagnostics
plot(fitted(model_poisson), residuals(model_poisson), pch=19,
    col="red",
      xlab = "Fitted values", ylab="Residuals", main="Residuals vs
         Fitted")
abline(h=0, col="black", lwd=2)
qqnorm(residuals(model_poisson), col="red", pch=16)
qqline(residuals(model_poisson), col="black", lwd=2)
```

Listing 3: Poisson regression model

8.5 Negative Binomial Regression

```
set.seed(123)
folds <- createFolds(train$Frequency.of.use, k = 5, list = TRUE,
    returnTrain = TRUE)
results <- vector("list", length = 5)

for (i in seq_along(folds)) {
  train_fold <- train[folds[[i]], ]
  test_fold <- train[-folds[[i]], ]

  model_nb <- glm.nb(Frequency.of.use ~ Subscription..Length +
      Charge..Amount + Seconds.of.Use +
```

```
                          Frequency.of.SMS + Distinct.Called.Numbers +
                             Age.Group +
                          Tariff.Plan + Status + Customer.Value,
                        data = train_fold)
  preds <- predict(model_nb, newdata = test_fold, type = "response")
  results[[i]] <- sqrt(mean((preds - test_fold$Frequency.of.use)^2))
}
nb_rmse <- mean(unlist(results))

# Future prediction
future_preds <- predict(model_nb, newdata = future_data, type =
    "response")
future_result <- sqrt(mean(future_preds -
    future_data$Frequency.of.use)^2)

# Multicollinearity check
vif_values <- vif(model_nb)
print(vif_values)

# Diagnostics
model_nb_full <- glm.nb(Frequency.of.use ~ Subscription..Length +
    Charge..Amount + Seconds.of.Use +
                          Frequency.of.SMS +
                             Distinct.Called.Numbers + Age.Group +
                          Tariff.Plan + Status + Customer.Value,
                             data = train)

plot(fitted(model_nb_full), residuals(model_nb_full), col="red",
```

```
      xlab="Fitted Values", ylab="Deviance Residuals", main="NB:
         Residuals vs Fitted")
abline(h=0, col="black", lwd=2)
qqnorm(residuals(model_nb_full), col="red", pch=16, main="Q-Q Plot
    (NB)")
qqline(residuals(model_nb_full), col="black", lwd=2)
```

Listing 4: Negative binomial regression with cross-validation

8.6 Regularized Regression (Ridge and Lasso)

```
data_clean <- na.omit(data)
x <- model.matrix(Frequency.of.use ~ Subscription..Length +
    Charge..Amount + Seconds.of.Use +
                    Frequency.of.SMS + Distinct.Called.Numbers +
                       Age.Group +
                    Tariff.Plan + Status + Customer.Value, data =
                       data_clean)[, -1]
y <- data_clean$Frequency.of.use

x_train <- x[train_idx, ]
y_train <- y[train_idx]
x_test <- x[-train_idx, ]
y_test <- y[-train_idx]

# Ridge Regression
set.seed(123)
cv_ridge <- cv.glmnet(x_train, y_train, alpha = 0, family =
    "poisson")
```

```r
ridge_model <- glmnet(x_train, y_train, alpha = 0, lambda =
    cv_ridge$lambda.min)
ridge_preds <- predict(ridge_model, newx = x_test)
ridge_rmse <- sqrt(mean((ridge_preds - y_test)^2))
plot(cv_ridge)

# Lasso Regression (same structure, alpha = 1)
cv_lasso <- cv.glmnet(x_train, y_train, alpha = 1, family =
    "poisson")
lasso_model <- glmnet(x_train, y_train, alpha = 1, lambda =
    cv_lasso$lambda.min)
lasso_preds <- predict(lasso_model, newx = x_test)
lasso_rmse <- sqrt(mean((lasso_preds - y_test)^2))

# Future prediction
x_future <- model.matrix(Frequency.of.use ~ Subscription..Length +
    Charge..Amount + Seconds.of.Use +
                             Frequency.of.SMS +
                                Distinct.Called.Numbers + Age.Group +
                             Tariff.Plan + Status + Customer.Value,
                                data = na.omit(future_data))[, -1]

future_preds_ridge <- predict(ridge_model, newx = x_future)
future_rmse_ridge <- sqrt(mean((future_preds_ridge -
    future_data$Frequency.of.use)^2))

future_preds_lasso <- predict(lasso_model, newx = x_future)
future_rmse_lasso <- sqrt(mean((future_preds_lasso -
    future_data$Frequency.of.use)^2))
```

Listing 5: Ridge and Lasso regression

8.7 Model Comparison

```
model_comparison <- data.frame(
  Model = c("Poisson", "Negative Binomial", "Ridge", "Lasso"),
  Test_RMSE = c(poisson_rmse, nb_rmse, ridge_rmse, lasso_rmse)
)
print(model_comparison)

best_model <-
    model_comparison$Model[which.min(model_comparison$Test_RMSE)]
cat("Best performing model:", best_model, "\n")
```

Listing 6: Comparison of model performance

8.8 Logistic Regression for Complaints Prediction

```
# Remove Age.Group and keep only Age
data <- data %>% select(-Age.Group)  # Remove Age.Group column

# Check the distribution of Complains in the dataset
cat("Distribution of Complains in the dataset:\n")
print(table(data$Complains))

# Split Data into Train and Test Sets
set.seed(123)  # For reproducibility
split_idx <- createDataPartition(data$Complains, p = 0.9, list =
    FALSE)
```

```r
train <- data[split_idx, ]
test <- data[-split_idx, ]

# Recode the target variable for caret compatibility
train$Complains <- factor(ifelse(train$Complains == 1, "yes", "no"),
                          levels = c("no", "yes"))
test$Complains <- factor(ifelse(test$Complains == 1, "yes", "no"),
                         levels = c("no", "yes"))

# Handle Class Imbalance with SMOTE
recipe <- recipe(Complains ~ ., data = train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_smote(Complains, over_ratio = 1)

# Prepare the balanced dataset
train_smote <- prep(recipe, training = train) %>% bake(new_data =
    NULL)

# Check distribution after SMOTE
cat("\nDistribution of Complains after SMOTE:\n")
print(table(train_smote$Complains))

# Train Logistic Regression Model
log_model_smote <- glm(
  Complains ~ .,
  data = train_smote,
  family = binomial(link = "logit")

# Model summary
```

```r
cat("\nSummary of Logistic Model (After SMOTE):\n")
summary(log_model_smote)

# Preprocess Test Data
test_processed <- prep(recipe, training = train) %>% bake(new_data = test)

# Predict on Test Set
pred_prob_smote <- predict(log_model_smote, newdata = test_processed,
                           type = "response")
pred_class_smote <- ifelse(pred_prob_smote > 0.5, "yes", "no")
pred_class_smote <- factor(pred_class_smote, levels = levels(test$Complains))

# Evaluate Model
cat("\nConfusion Matrix with 0.5 Threshold (After SMOTE):\n")
conf_matrix_smote <- confusionMatrix(pred_class_smote, test$Complains,
                                     positive = "yes")
print(conf_matrix_smote)

# ROC and AUC
roc_obj_smote <- roc(response = test$Complains, predictor = pred_prob_smote)
auc_value_smote <- auc(roc_obj_smote)
cat("\nAUC Value (After SMOTE):", auc_value_smote, "\n")

# Plot ROC Curve
```

```
63 plot(roc_obj_smote, main = "ROC Curve - Logistic Regression (After
       SMOTE)")
64 abline(a = 0, b = 1, lty = 2, col = "orange")
```

Listing 7: Logistic regression with SMOTE for class imbalance