Introduction

Customer churn is a significant challenge for businesses, particularly in the telecommunications industry, where customer acquisition costs are high, and long-term retention is crucial for profitability. Churning occurs when a customer decides to stop using a company's service, leading to revenue loss and increased marketing expenses to replace lost customers. Customer churn is influenced by multiple factors such as service quality, contract types, competitive offers, and customer experience, making it challenging to model effectively.

Predicting customer churn allows companies to take proactive measures to retain customers by offering incentives, improving customer service, or adjusting pricing models. The ability to accurately forecast churn helps businesses reduce customer attrition rates and optimize customer relationship management strategies.

In the telecom industry, churn prediction is a key performance metric. Telecom companies rely on monthly subscriptions and contracts to maintain steady revenue streams. A high churn rate can indicate that customers are not satisfied with service quality, pricing, or customer support. By leveraging machine learning techniques, companies can analyze patterns in customer behavior and identify at-risk customers before they leave.

Reducing churn has direct financial benefits. Given the competitive nature of the industry, retaining customers is a key priority for companies seeking to maintain sustainable growth. Studies suggest that retaining an existing customer is five times more cost-effective than acquiring a new one. Predictive churn models allow businesses to focus on high-risk customers and implement targeted marketing campaigns, thereby increasing retention rates and overall customer satisfaction.

The objective of this project is to develop a machine learning model that accurately predicts customer churn using historical customer data. The model will be trained on a dataset containing customer demographics, account details, subscription plans, and usage behavior. We will be identifying key factors contributing to churn and preprocess the dataset for optimal model performance, training multiple classification and machine learning models, and comparing and analyzing their performance to get valuable insights to potentially help telcom companies predict reduce churning rates.

Dataset

The data set used for this project is the IBM Telco Customer Churn Dataset. I got it from kaggle. Here is the website: https://www.kaggle.com/datasets/blastchar/telco-customer-churn.

The Telco customer churn data contains information about a telco company that provided home phone and Internet services to 7043 customers in California during its third quarter of a given financial year. It indicates which customers have left, stayed, or signed up for their service. The data set contains information about customer demographics, account details, subscription plans, and usage behavior. The dataset includes 7043 observations and 21 features, including categorical and numerical variables. The target variable is Churn, a binary classification variable indicating whether a customer has left the telecom provider (1 = Churn, 0 = No Churn).

Some of the other variables that we have in the data set include:

1. Customer Demographics: Customer Id, Gender, Senior Citizen, Partner, Dependents
2. Account Information: Tenure (service length in months), Contract Type (Month-to-Month, One-Year, Two-Year), Payment Method, Paperless Billing
3. Services Bought: Internet Service, Multiple Lines, Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, Streaming Movies
4. Payment Information: Monthly Charges and Total Charges

Data Cleaning

```
library(readxl)
library(tidyverse)

churn_data <- read_excel("Telco Customer Churn.xlsx")

churn_data$TotalCharges[is.na(churn_data$TotalCharges)] <-
  median(churn_data$TotalCharges, na.rm = TRUE)

churn_data <- churn_data[!duplicated(churn_data), ]
```

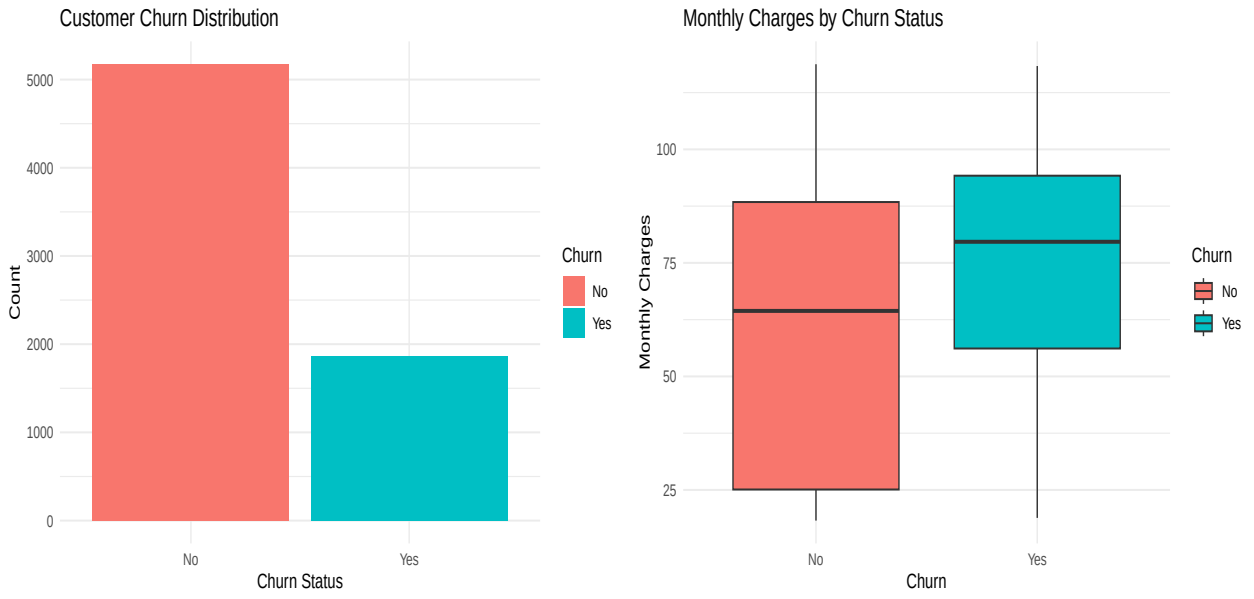
First off, we start by loading and cleaning the data. We clean the data by accounting for missing values and removing all duplicate entries. We do not want to have any missing values to avoid biased and misleading models. If we plot distributions and ignore missing values, the visualized patterns may be incomplete. We do this by replacing all missing values with the median of that variable's values. This is because the median is the metric that is least affected by the outlier, and it is better than removing the entire entry. We also remove all duplicate values in order to avoid redundancy and bias caused by overclustering.

```
categorical_cols <- c("gender", "Partner", "Dependents", "PhoneService", "MultipleLines",
                      "InternetService", "OnlineSecurity","OnlineBackup",
                      "DeviceProtection", "TechSupport", "StreamingTV", "StreamingMovies",
                      "Contract", "PaperlessBilling", "PaymentMethod", "Churn")

churn_data[categorical_cols] <- lapply(churn_data[categorical_cols], as.factor)
```

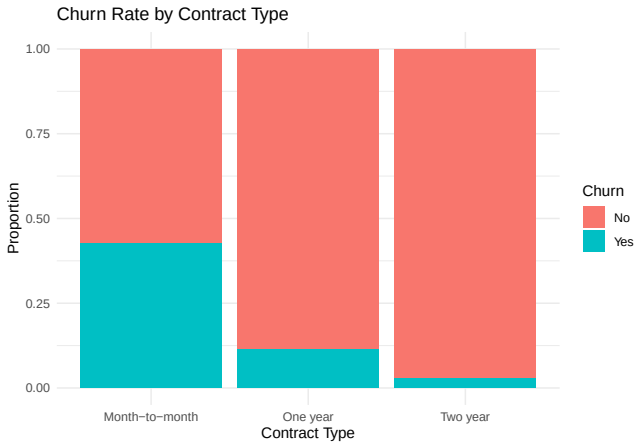
We also converted the categorical variable values to numerical values because they need to be formatted for models to understand and process them correctly in machine learning and statistical modeling.

Exploratory Analysis & Visualization



This bar plot on the left tells us that more customers that do not churn (more than 5,000) than those who do churn, almost 2,000.

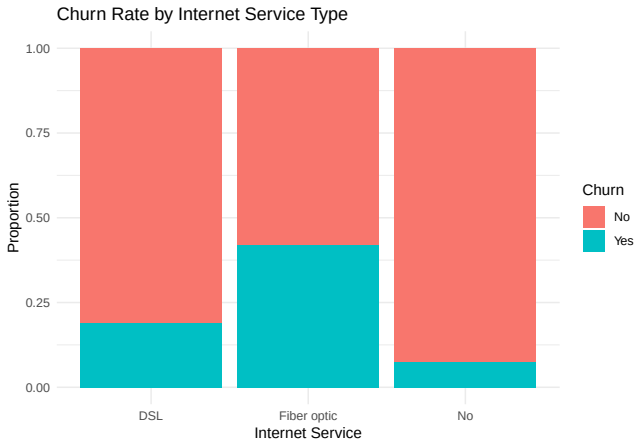
This boxplot on the right shows us that customers who churned tend to have higher median and quartile monthly charges than those who did not churn. Customers with lower Monthly Charges seem less likely to churn. The spread of Monthly Charges for churned customers is bigger, indicating more variation in charges. This makes sense because it is the low prices that attract customers. Some customers may feel that they are not getting enough value for the upcharges and start looking for new plans. Once prices start to increase and/or when customers find a better deal else where, they cancel their plans and move on.



This barplot shows us the chrun rates for each of the contract types: month-to-moth, one year, and two year contracts. The Month-to-Month contracts have the highest churn rate at around 40%, meaning a significant portion of these customers leave once their monthly contract ends. This means that are more flexible and more likely to leave if they find a better deal or become dissatisfied with the service.

The One year contract customers have a moderate churn rate at around 12%, but much lower than month-to-month. This is because they are already in a contract and find it easier to just renew their contracts at the end of the one year. The two-year contracts by far have the lowest churn rate, meaning customers with long-term commitments are less likely to churn.

This is because a lot of the times, it is easer to retain these custoemrs because they do not want to go throught all the hassle to find a different service provider, and if they think about leaving, the telcom company can give them discounts or incentives to make these customers stay. These customers are already in a contractual agreement, making it easier for them to renew rather than switch providers once their contract ends. This shows that longer contract commitments help in customer retention.



This barplot shows us the churn rates based on the internet plans that customers are on. The “No” variable

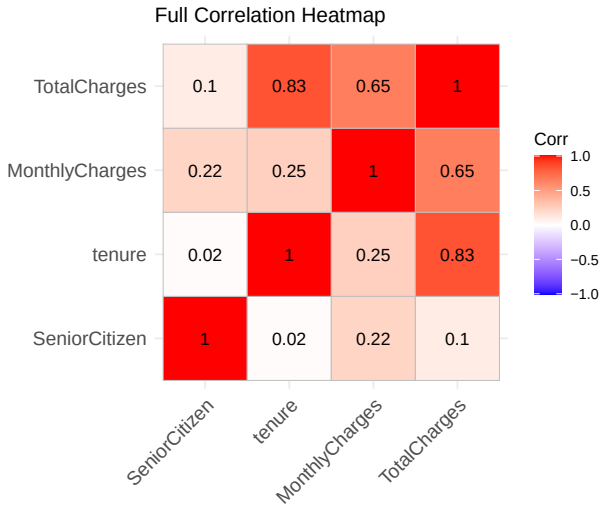
means that the customers are not on any of the internet plans and are just on the mobile plans for their phones.

DSL customers have a lower churn rate, which suggests that they may be on more stable contracts or find the service satisfactory. This might be because they are on contracts with fewer service disruptions.

Fiber Optic customers have the highest churn rate, meaning they are more likely to leave compared to DSL or customers with no internet service. This might be because the fiber opic connection might not be as strong, or the rates might be higher than the value that the customers are actually getting.

Customers with no internet service churn the least, likely because they are using basic phone services and have fewer reasons to switch. This is likely because they are not as affected by service issues or prices as much as internet users.

Understanding the reasons behind DSL customers' lower churn could help apply similar strategies to Fiber Optic services. Retention efforts should focus on Fiber Optic customers, possibly by offering better prices, improving service connections, technical support, better customer services, etc.



To better understand the relationships between numerical variables in our dataset, we generated a correlation heatmap.

The correlation that stood out the most was the correlation between TotalCharges and tenure, which had a pretty high value at 0.83. This means that the two variables are strongly positively correlatted, which tells us that both of these variables increase together as well as decrease together. This might be because customers who have been with the company longer tend to accumulate higher total charges over time. This suggests that tenure plays a crucial role in determining how much revenue a customer generates throughout their contract.

Additionally, correlation is between Monthly Charges and Total Charges is at 0.65. This means that the two variables have a moderately strong positive correlation, but this is not 100% confirmed yet or we have other factors affecting this value. Both of these variable also go in the same direction as the other. This means that customers who pay higher monthly fees also tend to have a higher overall charge. However, due to the correlation not being as strong as it should be, additional factors or variables, such as contract type or service add-ons, may also influence total charges.

The correlation between tenure and Monthly Charges is at 0.25, which means that it is relatively weak. This means that customer's time with the company does not significantly impact their monthly charges. This might be because customers with longer contracts may have locked in lower rates, while newer customers could be paying higher monthly fees due to updated pricing models. This might be because the telcom company only gives lower prices to customers on contract to make sure they renew their contracts later, rather than leaving for another telcom company.

One variable that had a very weak correlation with each of the other variables was Senior Citizen, which was whether or not a given customer was a senior citizen or not. This tells us that that a customer's age group does not play a significant role in determining their tenure, monthly charges or total charges. We should not rely on the age variable alone, but also look at other variables and factors that can affect charges and churn rates.

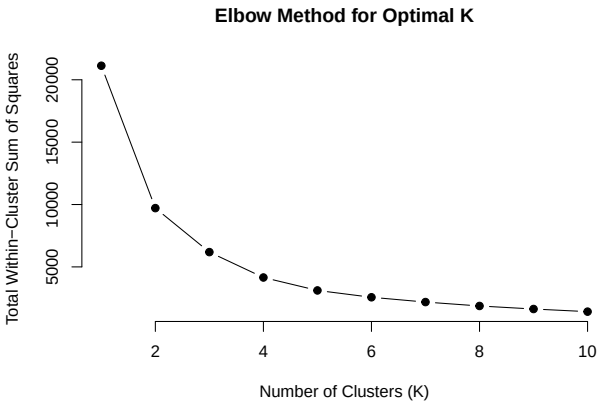
While tenure and total charges are strongly related, they may not be direct indicators of churn risk. Customers with higher monthly charges may be at higher risk of churn, but other variables such as customer satisfaction, billing method, and contract flexibility might give us more insights.

We will further explore contract type, internet service, and payment method to assess their impact on churn. Additionally, K-Means clustering will be applied to segment customers based on key characteristics, allowing for a more granular understanding of high-risk churn groups.

K-Means Clustering

K-Means clustering is a powerful unsupervised machine learning technique used to group customers based on similar characteristics. Here, we apply K-Means to segment customers based on tenure, Monthly Charges, and Total Charges. This helps us identify which customer groups are most at risk of churning and allows for more targeted churn prediction modeling later.

By analyzing these clusters, we can gain insights into customer behaviors and develop strategies to retain high-risk customers. Since K-Means does not predict churn but rather identifies patterns within customer data, it is a valuable exploratory tool to understand the underlying structure of the data set before applying predictive models.



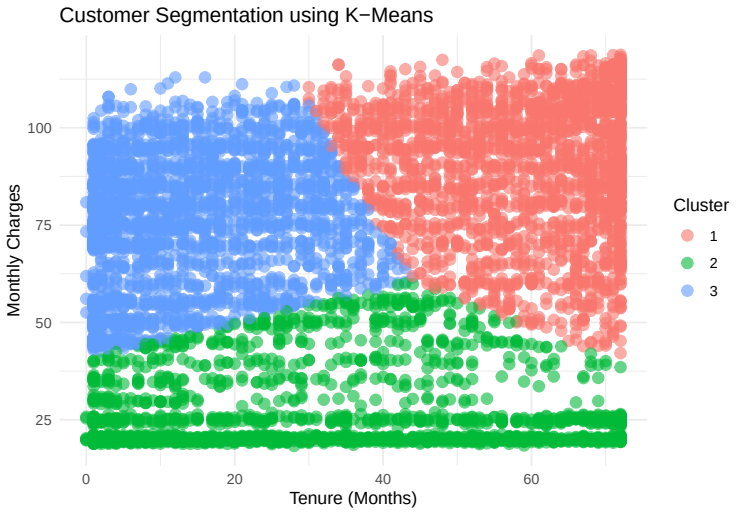
The Elbow Method is a technique used to determine the optimal number of clusters (K) in K-Means clustering. Since K-Means requires us to predefine the number of clusters, it is crucial to find the perfect sweet spot which is not too high nor too low. The Elbow Method helps achieve this by analyzing the Total Within-Cluster Sum of Squares (WSS).

Here, we plot the WSS against different values of K, which measures the compactness of clusters. Lower values indicate better clustering. The elbow point represents the optimal K, where increasing clusters beyond this point will not really help.

If K is too small, clusters may be too broad, and they wil not be able to capture meaningful differences between customers. If K is too large, clusters may become too specific, and they capture noise instead of general trends.

In this case, the optimal K value is around 3 and 4. Therefore, we will go with k = 3, since the lower, the better. For this project, the algorithm divides our customers into three distinct groups, based on tenure,

Monthly Charges, and Total Charges. These clusters represent patterns of customer behavior to help us identify different customer types with similar characteristics.



Cluster 1, which is the Red cluster in this plot, represents the customers with long tenures and high monthly prices. These are most likely loyal customers who are less likely to churn because are in long-term contracts or have bought premium services. These are the customer who are least likely to leave, however, the telcom company can make sure that these customers renew their plan with them by gving them discounts, better plans, technical support etc.

Cluster 2, which is the Green cluster of points, represents the customers who have low monthly charges, but they are spreading across different tenure lengths. Most likely, these customers might be on basic or discounted plans, making them less likely to leave due to affordability. By staying with this telcom company, these customers will not have to go through the hassle of finding plans that are cheaper then these already low-cost plans. These are the types of customers to whom the telcom company will try to sell additional plans because these customers might be willing to but these plans due to being satisfied with their cheaper plans.

Cluster 3, which is the cluster composed of the Blue points, consists of the customers with high monthly charges but short tenures. These customers are the riskiest ones because they are the ones who are most likely churn and cancel/not renew their plans. They might be on the month-to-month contracts that cost a lot, and would be looking else where for cheaper plans. Retention strategies should focus on this group, such as offering discounts or better customer service to encourage them to stay.

The next step will involve incorporating these cluster insights into classification models to predict churn more effectively. This will help in determining how different customer segments contribute to overall churn rates and enable better decision-making in customer retention strategies.

Table 1: Cluster Centers for K-Means Clustering

Cluster	Tenure	Monthly Charges	Total Charges
Cluster 1	1.0663	0.8287	1.3085
Cluster 2	-0.1175	-1.2702	-0.6489
Cluster 3	-0.7778	0.3383	-0.5508

The Cluster Centers for K-Means Clustering table shows us the average values for each cluster across these three features. These values are standardized, meaning they are expressed in terms of z-scores rather than raw values. Positive values indicate that the cluster's average is above the dataset's overall mean, and

negative values indicate that the cluster's average is below the dataset's overall mean. The closer values are to 0, the more they align with the data set's mean.

Cluster 1, which is the Red cluster, customers have the highest tenures as well as prices. Since they have been customers for a long time and pay high monthly charges, their cumulative spending is also high. This is the most stable group.

Customers in Cluster 2 have an average of a mid-range tenure, which means that they neither are not too new nor are they too old. They pay significantly lower-than-average monthly fees. Their cumulative spending is below average, suggesting they haven't been around long enough to contribute significantly to revenue. They could be at moderate churn risk, as they haven't committed long-term. In order to prevent churn, the company could offer targeted promotions, personalized incentives, etc.

Customers in Cluster 3 have the smallest tenures and pay moderate monthly fees, meaning they might be on basic plans. Since they haven't been with the company for long, their cumulative total spending is low. This cluster consists of newer customers who are price-sensitive and might be trial users or customers testing the service before making a long-term commitment. This is the group that the telcom company should attract for the long term by doing something they would do to customers who are at risk of churning.

Logistic Regression

The goal of this analysis is to predict customer churn, which is a binary classification problem where we determine whether a customer will churn ("Yes") or stay ("No") based on historical customer data. The dataset includes both numerical and categorical features related to customer demographics, account details, and service usage patterns.

This is a supervised learning problem, as we have labeled data indicating whether a customer has churned in the past. Our goal is to train a machine learning model that can predict future churn based on past patterns.

Logistic Regression is a linear classification model that estimates the probability of an outcome occurring. It models the relationship between the predictor variables and the probability of churn using the logistic (sigmoid) function:

$$P(\text{Churn}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_n X_n)}}$$

where:

- $P(\text{Churn})$ is the **probability of a customer churning**.
- β_0 is the **intercept term**.
- $\beta_1, \beta_2, ..., \beta_n$ are the **coefficients for each predictor variable** $X_1, X_2, ..., X_n$.

Unlike linear regression, which predicts continuous values, logistic regression predicts probabilities between 0 and 1, which are then thresholded (e.g., above 0.5 = churn, below 0.5 = stay).

Some advantages of logistic regression are that it is as simple and interpretable model, omputationally efficient, and provides probability estimates.

This will be our baseline model, and it will identify key factors influencing churn, such as contract type, tenure, and charges, and it lays the foundation for further model comparisons.

Statistic	Value
Null Deviance	6522.733
Residual Deviance	4812.321

AIC	4828.321

Table 4: Exponentiated Coefficients (Odds Ratios)

Term	Odds_Ratio
(Intercept)	1.0114842
tenure	0.9417738
MonthlyCharges	0.9968389
TotalCharges	1.0003391
ContractOne year	0.4228595
ContractTwo year	0.1662090
InternetServiceFiber optic	2.9719950
InternetServiceNo	0.3175962

Several predictors were found to be statistically significant (their p-values were less than 0.05), meaning they have a meaningful impact on churning. Contract type was one of the strongest indicators of churn behavior. Customers with two-year contracts had a significantly lower probability of churning, suggesting that longer contracts contribute to customer retention. Similarly, customers with longer tenure were also less likely to churn, indicating that customer loyalty strengthens over time.

The type of internet service had a notable impact on churn. Customers using fiber optic internet were more likely to churn, possibly due to dissatisfaction with service quality or pricing. As mentioned earlier in the report, customers who did not have internet service were less likely to churn.

However, monthly charges were not statistically significant, suggesting that pricing alone does not play a major role in churn.

The model's performance was evaluated using the Akaike Information Criterion (AIC), which was 4828.3, serving as a baseline metric for comparison with other models. Additionally, the null deviance of 6522.7 vs. the residual deviance of 4812.3 shows that the model significantly reduces uncertainty in predicting churn, confirming its usefulness.

To enhance interpretability, we computed odds ratios by exponentiating the logistic regression coefficients. The results showed that customers with a two-year contract had an 83.4% lower likelihood of churning compared to those with month-to-month contracts. Customers using fiber optic internet were nearly three times more likely to churn compared to DSL customers, reinforcing the need to investigate dissatisfaction among fiber optic users.

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 5: Confusion Matrix for Logistic Regression Model

Reference	Prediction	Count
No	No	931
Yes	No	103
No	Yes	197

Yes	Yes	176

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 6: Confusion Matrix Evaluation Metrics for Logistic Regression

Metric	Value
Accuracy	0.7867804
P-Value [Acc > NIR]	0.0000038
Sensitivity (Recall)	0.9003868
Specificity	0.4718499
Positive Predictive Value (Precision)	0.8253546
Negative Predictive Value	0.6308244
Prevalence	0.7348969
Detection Rate	0.6616915
Detection Prevalence	0.8017058
Balanced Accuracy	0.6861184

The model correctly classified 76.68% of all cases. 90.04% recall means the model is very good at identifying churners, minimizing false negatives. Only 47.18% specificity means that there were a lot of false positives, which predicts churning when the customer actually stays. Among all customers predicted as churners, 82.54% actually churned. Among all customers predicted as non-churners, 63.08% actually stayed.

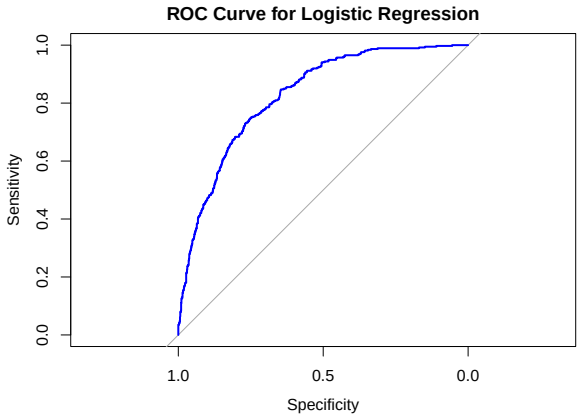


Table 7: AUC Value for Logistic Regression Model

Metric	Value
Area Under the Curve (AUC)	0.8263

The ROC (Receiver Operating Characteristic) curve plots Sensitivity (Recall) vs. 1 - Specificity. The ROC curve, in blue, is clearly above the diagonal, indicating a good level of separability between churners and

Table 2: Logistic Regression Results

Term	Estimate	Std_Error	Z_value	P_value	Odds_Ratio
(Intercept)	0.0114	0.2161	0.0528	0.9579	1.0115
tenure	-0.0600	0.0070	-8.6240	0.0000	0.9418
MonthlyCharges	-0.0032	0.0038	-0.8253	0.4092	0.9968
TotalCharges	0.0003	0.0001	4.2715	0.0000	1.0003
ContractOne year	-0.8607	0.1160	-7.4200	0.0000	0.4229
ContractTwo year	-1.7945	0.1901	-9.4378	0.0000	0.1662
InternetServiceFiber optic	1.0892	0.1421	7.6679	0.0000	2.9720
InternetServiceNo	-1.1470	0.1763	-6.5058	0.0000	0.3176

non-churners. The model's AUC = 0.8263 tells us that there is an 82.63% chance that the model ranks a randomly chosen churner higher than a non-churner. This confirms that the logistic regression model has strong predictive power.

Overall, the logistic regression model provides valuable insights into the drivers of churn. It confirms that contract type, tenure, and internet service type are major factors influencing customer retention, while monthly charges have a limited impact. However, since logistic regression assumes a linear relationship in log-odds space, it may struggle to capture complex interactions between variables and not perform well with complex, non-linear relationships.

Decision Trees

Decision Trees are non-linear models that can capture interactions between variables better than logistic regression. They work by splitting the data set into different groups based on features that best separate churners from non-churners. We do this based on demographic, account, and service-related features. Decision Trees also use if-else conditions to split the data at each step. The tree chooses the best feature to split on at each step, aiming to create the most pure subgroups, either mostly Yes or mostly No. The tree continues splitting until it reaches a stopping criterion, such as a minimum number of samples per leaf or maximum tree depth.

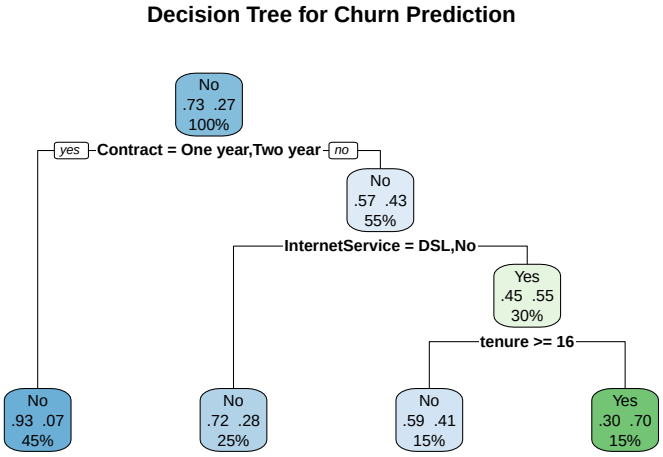
Some advantages of decision trees are that they can model complex relationships, the trees chooses which features matter most, and the trees structure is easy to understand and explain to non-technical stakeholders.

Since Decision Trees can automatically detect interactions between features, we expect it to identify key churn predictors dynamically without assuming linear relationships, give us a visual representation of how decisions are made, as well as help us with more advanced machine learning models such as Random Forests.

$$\text{Gini Impurity} = 1 - \sum_{i=1}^{c} p_i^2$$

where:

- c is the number of classes (for churn prediction, $c = 2$, representing "Yes" and "No").
- p_i is the proportion of instances belonging to class i in the split.



The Decision Tree splits the data at different points to separate churners from non-churners based on key features. The first and most ritical factor is Contract Type (One-Year or Two-Year). Most of the customer on long term contracts do not chrun. If they are month-to-month contracts, the model proceeds to further splits to assess churn risk.

The second split is if the contract is Month-to-Month, and the next important feature is Internet Service Type. Customers with DSL or No Internet Service have a lower churn risk, but customers with Fiber Optic Internet are more likely to churn.

The third split happens when customers are not on Fiber Optic Internet and consists of DSL and No Internet Customers. Long-term contracts significantly reduce churn, while Fiber Optic customers have a higher risk of leaving. Customers who have been there for more than 16 months are less likely to churn, indicating more loyalty over time.

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 8: Confusion Matrix for Decision Tree Model

Reference	Prediction	Count
No	No	968
Yes	No	66
No	Yes	233
Yes	Yes	140

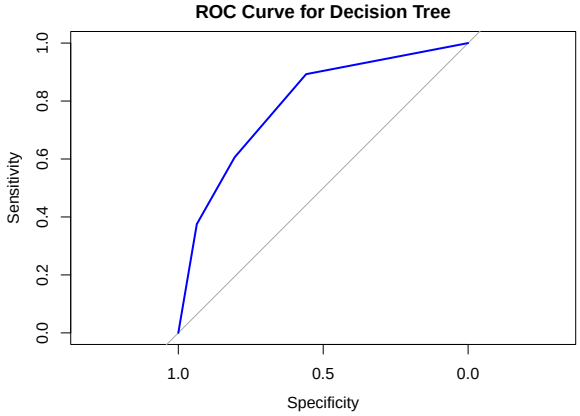
```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 9: Confusion Matrix Evaluation Metrics for Decision Tree

Metric	Value
Accuracy	0.7874911
P-Value [Acc > NIR]	0.0000028
Sensitivity (Recall)	0.9361702
Specificity	0.3753351
Positive Predictive Value (Precision)	0.8059950
Negative Predictive Value	0.6796117
Prevalence	0.7348969
Detection Rate	0.6879886
Detection Prevalence	0.8535892
Balanced Accuracy	0.6557527

The number of true negatives, which is the correctly predicted number non-churners is 968. The number of false positives, which is the number of customers incorrectly predicted as churners but actually stayed, is 233. The number of false positives, which is the customers incorrectly predicted as staying but actually churned, is 66. The number of true positives, which is the correctly predicted churners is 140.

The model achieved an accuracy of 78.75%, which is slightly higher than the logistic regression model. The recall, or sensitivity, is 93.62%, which means that the Decision Tree successfully captures most churners and effective at identifying customers at risk of leaving. It was a little higher than the logistic regression model. The specificity, which measures how well the model identifies non-churners, was 37.53%, telling us that it really struggles to correctly predict loyal customers. This value was lower than the logistic regression model. Among all predicted churners, only 30.06% actually churned, which is very low, and this means that the model over-predicts churners. The balanced accuracy, which accounts for sensitivity and specificity was 65.75%, which was lower than the logistic regression model.




```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 10: AUC Value for Decision Tree

Metric	Value
Area Under the Curve (AUC)	0.7899

The AUC score for this model ended up being 0.7899, which was slightly lower than the Logistic Regression model's AUC. This suggests that the tree is likely overfitting to training data The curve is less steep at the beginning, meaning it misclassifies some churners early on in the process.

Decision Trees capture non-linearity well but are prone to overfitting and can be unstable, meaning small changes in data can lead to drastically different trees.

Random Forest

Random Forest is an ensemble model that makes multiple Decision Trees and combines their predictions to improve accuracy and robustness. This is to reduce overfitting, improve accuracy and generalization, and handle non-linearity and complex feature interactions better.

First, the model starts off bootstrap sampling, where it creates multiple random subsets of the training data by sampling with replacement. Then it randomly selects a subset of features to reduces correlation between trees and prevents overfitting. After that, each tree in the forest will vote on whether a customer will churn or not. By using multiple trees instead of one, Random Forest smooths out errors and prevents bias, leading to a more accurate model. It is represented by:

$$\hat{y} = \frac{1}{T} \sum_{t=1}^{T} f_t(x)$$

where:

- $\hat{y}$ is the final predicted class (churn or no churn).
- T is the number of trees in the Random Forest.
- $f_t(x)$ is the output of the t-th decision tree.

Some advantages that it has are that it has a higher accuracy than a single Decision Tree. By reducing variance, it is less prone to overfitting, it handles high-dimensional data and both categorical and continuous variables, and provides rankings of each factor.

Table 11: Summary of Random Forest Model

Metric	Value
Number of Trees	500
Number of Variables Tried at Each Split	3
OOB Error Rate	22.52%

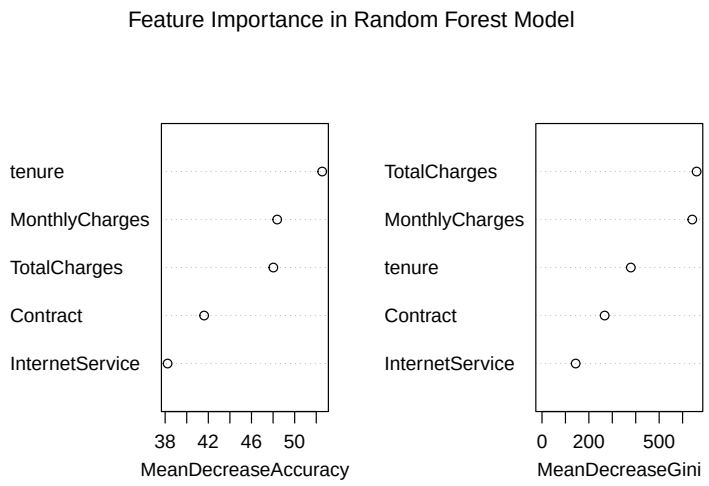
Table 12: Confusion Matrix for Random Forest Model

Class	Predicted No	Predicted Yes	Class Error
No	3613	527	0.1272947

Yes	742	754	0.4959893

Table 13: Feature Importance in Random Forest Model

Feature	No	Yes	MeanDecreaseAccuracy	MeanDecreaseGini
tenure	25.857756	19.68235	52.54983	379.0806
MonthlyCharges	28.279314	25.41698	48.38064	641.3239
TotalCharges	31.359092	14.95899	48.02431	659.8021
Contract	-13.889092	45.52514	41.61173	267.3725
InternetService	9.774824	56.20706	38.22720	143.6801



The Random Forest model was trained using 500 trees and considered 3 features per split to balance accuracy and generalizability. The Out-of-Bag (OOB) error estimate is 22.52%, meaning the model correctly classifies approximately 77.48% of customers as either churners or non-churners. The ranking of the most important factors in order is:

1. Tenure – Longer tenure decreases churn probability
2. Monthly Charges – Higher monthly charges increase churn risk
3. Total Charges – Higher total charges correlate with customer retention.
4. Contract Type – Customers on Month-to-Month contracts are more likely to churn
5. Internet Service – Fiber Optic customers are more likely to churn than DSL or No Internet customers

The OOB confusion matrix showed that the model correctly predicted 3613 non-churners while misclassifying 742 customers as churners when they actually stayed. It also correctly predicted 754 actual churners, but 527 churners were misclassified as non-churners. The model achieved an OOB accuracy of 77.48%, meaning it correctly classified approximately three out of four customers. However, the false positive rate was relatively high, meaning many customers who were not going to churn were incorrectly flagged as churners. This could lead to unnecessary retention efforts, increasing costs for the business.

The model correctly classifies most non-churners (3613 out of total 5135 non-churners). The false positive rate of 742 suggests some non-churners were misclassified as churners, which could lead to unnecessary

retention efforts. The model correctly predicts 754 churners, but it misses 527 churners, meaning recall could be improved.

Table 14: Confusion Matrix for Random Forest Model

Reference	Prediction	Count
No	No	894
Yes	No	140
No	Yes	197
Yes	Yes	176

Table 15: Confusion Matrix Evaluation Metrics for Random Forest

Metric	Value
Accuracy	0.7604833
P-Value [Acc > NIR]	0.0153013
Mcnemar's Test P-Value	0.0022845
Sensitivity (Recall)	0.8646035
Specificity	0.4718499
Positive Predictive Value (Precision)	0.8194317
Negative Predictive Value	0.5569620
Prevalence	0.7348969
Detection Rate	0.6353945
Detection Prevalence	0.7754087
Balanced Accuracy	0.6682267

After training, the model was evaluated using a separate test dataset, simulating real-world application. The test confusion matrix indicated an accuracy of 76.05%, slightly lower than the OOB estimate, which is expected as models typically perform slightly worse on unseen data.

Precision improved on the test data to 81.94%, indicating fewer false churn predictions compared to the OOB estimate. The test results showed that 894 non-churners were correctly classified, while 140 were incorrectly identified as churners (false positives). Additionally, the model successfully identified 176 actual churners, but 197 churners were incorrectly classified as non-churners (false negatives). It had a lower precsision than logistic regression butstill more than the decison tree.

A key strength of the model was its high recall of 86.46%, meaning it effectively captured a large portion of actual churners. It was still lower then logistic regression's recall thoug. However, its specificity of 47.18% was relatively low and similar to the logistic regression model, meaning it struggled to correctly identify non-churners. This suggests that while the model is effective at predicting churn, it may still flag too many customers as potential churners when they are not at risk or churning at all.

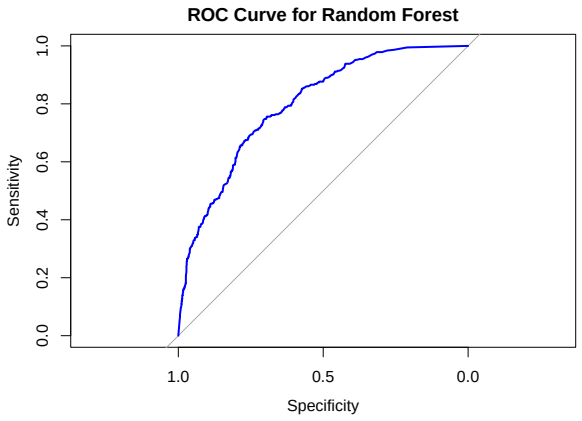


Table 16: AUC Value for Logistic Random Forest

Metric	Value
Area Under the Curve (AUC)	0.7969

The AUC was 0.7969, which means the model has a good ability to distinguish between churners and non-churners. It is higher than the Decision Tree, showing that Random Forest's ensemble learning reduces overfitting and improves generalization. However, the logistic regression model had a better AUC, suggesting that a simpler linear model is performing well for this data set.

Though random forest models are really strong and have some advantages, they also have some disadvantages. They lack interpretability because they consists of many trees, making it harder to explain individual predictions. While Random Forests reduce overfitting compared to Decision Trees, they can still overfit if the number of trees is too large or if irrelevant features are included. They also struggle with class imbalance because they build trees based on overall accuracy rather than focusing on minority classes.

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification tasks. It works by finding the optimal hyperplane that best separates data points into two classes — in this case, churners and non-churners. Unlike models such as logistic regression, which assumes a linear relationship between features and the target variable, SVM maximizes the margin between the two classes, improving generalization to new data.

Since customer churn prediction often involves complex, non-linear relationships, we use a Radial Basis Function (RBF) kernel to map the data into a higher-dimensional space where a clear separation is possible. This makes SVM particularly useful when linear models like logistic regression fail to capture non-linearity in the data.

$$\min_{w,b} \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i + b))$$

where:

- $||w||^2$ represents the **regularization term**, controlling model complexity.
- C is the **penalty parameter**, balancing margin maximization and misclassification tolerance.

- y_i is the **true class label**, where Churn = 1 and No Churn = -1.
- x_i is the **input feature vector**.

SVM is robust to imbalanced datasets because it focuses on correctly classifying the most difficult-to-classify cases, rather than simply optimizing for accuracy. This property makes it a strong candidate for churn prediction, where the number of churners is often much smaller than non-churners.

One of the main advantages of SVM is that it works well in high-dimensional spaces, making it suitable for datasets with many features. Additionally, SVM is less prone to overfitting than decision trees and random forests. However, it comes with some limitations, such as high computational cost, difficulty in tuning hyperparameters (C and gamma), and lack of interpretability compared to tree-based models.

Given these strengths and weaknesses, SVM is a valuable addition to our churn prediction models, offering an alternative to logistic regression and decision trees. By testing SVM, we aim to determine whether it can improve classification performance over the models tested so far.

Table 17: Summary of SVM Model Parameters

Parameter	Value
SVM-Type	C-classification
SVM-Kernel	Radial
Cost	1
Gamma	0.1
Number of Support Vectors	2505

Table 18: Sample Predictions from SVM Model

Actual	Predicted	Probability
No	No	0.4551
Yes	No	0.3528
No	No	0.1624
No	No	0.1625
Yes	No	0.1730
No	No	0.1642
No	No	0.1643
No	No	0.1678
Yes	No	0.2788
Yes	Yes	0.6855

Table 19: Training Set Confusion Matrix for SVM Model

Reference	Prediction	Count
No	No	961
Yes	No	73
No	Yes	222
Yes	Yes	151

The trained model used a cost parameter (C) of 1, which balances the trade-off between maximizing the margin and minimizing classification error. The number of support vectors used was 2,505, meaning that a

significant portion of the training data contributed to defining the decision boundary. This suggests that the model is capturing complex relationships within the data rather than relying on a small subset of points.

After training, the model was applied to the test dataset to make predictions. The output showed that 1,183 customers were classified as "No" (non-churners), while 224 were classified as "Yes" (churners). This result indicates that the model is more inclined to classify customers as non-churners, which may be a reflection of class imbalance in the dataset.

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 20: Confusion Matrix for SVM Model

Reference	Prediction	Count
No	No	947
Yes	No	87
No	Yes	210
Yes	Yes	163

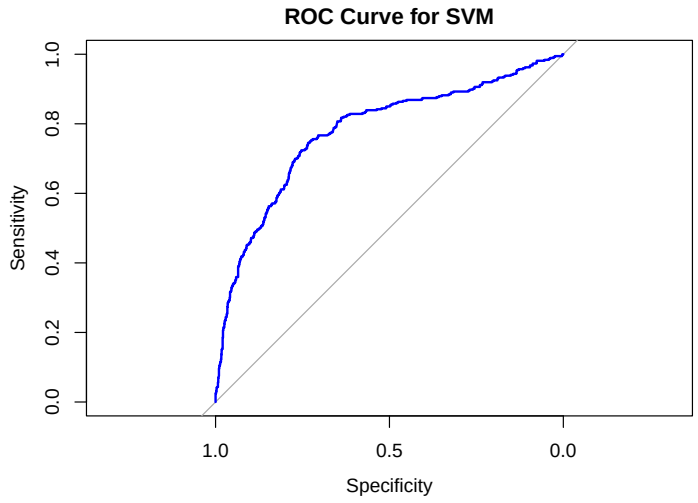
```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 21: Test Set Confusion Matrix Evaluation Metrics for SVM

	Metric	Value
Accuracy	Accuracy	0.7903340
AccuracyPValue	P-Value [Acc > NIR]	0.0000008
Sensitivity	Sensitivity (Recall)	0.9294004
Specificity	Specificity	0.4048257
Precision	Positive Predictive Value (Precision)	0.8123415
Neg Pred Value	Negative Predictive Value	0.6741071
Prevalence	Prevalence	0.7348969
Detection Rate	Detection Rate	0.6830135
Detection Prevalence	Detection Prevalence	0.8407960
Balanced Accuracy	Balanced Accuracy	0.6671131

The model achieved an accuracy of 79.03%, meaning it correctly classified a large portion of customers. However, analyzing the recall (sensitivity) and specificity provides a more detailed understanding of its effectiveness. The model's recall of 29.24% indicates that it correctly identified only 29.24% of actual churners, meaning it missed a significant number of customers who actually churned. On the other hand, the specificity of 90.48% shows that it correctly identified 90.48% of customers who stayed, making it highly effective in predicting non-churners.

The positive predictive value of 81.23% indicates that when the model predicts a customer will churn, it is correct in 81.23% of cases. However, its negative predictive value of 67.31% shows that it struggles with correctly classifying non-churners. The McNemar's test p-value, 2.2e-16, indicates a statistically significant difference between the two classes, further supporting the model's validity.




```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 22: AUC Value for SVM Model

Metric	Value
Area Under the Curve (AUC)	0.7774

The Area Under the Curve (AUC) score of 0.7774 indicates that the model performs reasonably well but is slightly weaker than other models tested, such as logistic regression (AUC = 0.8263) and random forest (AUC = 0.7969).

In this case, the SVM model has a moderate ability to differentiate between classes, but it is less effective compared to the other models tested. Class imbalance in the dataset could be a factor affecting the model's ability to correctly classify churners. The SVM performed the weakest in terms of AUC but still provided reasonable predictions. The lower AUC suggests that it is not the best model for churn prediction, but it could still be useful when combined with other models in an ensemble approach.

Neural Network

Neural Networks are a powerful class of machine learning models that are designed to capture complex, non-linear relationships in data. Unlike traditional models such as logistic regression or decision trees, which rely on predefined assumptions about the data structure, neural networks learn hierarchical representations through multiple layers of computation.

A neural network consists of layers of interconnected neurons, where each neuron processes inputs using a weighted sum followed by an activation function. A simple feed forward neural network can be represented as:

$$h^{(l)} = f(W^{(l)}h^{(l-1)} + b^{(l)})$$

where:

- $h^{(l)}$ is the activation at layer l,
- $W^{(l)}$ and $b^{(l)}$ are the weights and biases of layer l,
- f is the activation function, such as ReLU or sigmoid,
- $h^{(0)}$ corresponds to the input features (e.g., tenure, monthly charges, contract type, etc.).

Neural networks are well-suited for churn prediction due to their ability to capture complex patterns in customer behavior that traditional models may overlook. Unlike logistic regression or decision trees, neural networks can automatically detect nonlinear relationships and intricate dependencies among multiple features, improving predictive accuracy.

Another advantage of neural networks is their capacity to handle large amounts of data efficiently through deep architectures. In churn prediction, customer data often includes a mix of categorical and numerical variables as well as historical interactions. Neural networks can effectively process this information, learning hierarchical representations that improve classification performance.

Additionally, neural networks automatically learn feature interactions, reducing the need for extensive manual feature engineering. Traditional models often require domain expertise to identify meaningful interactions, while neural networks can extract these relationships dynamically during training. This ability makes them particularly useful in churn prediction, where customer retention is influenced by multiple interacting factors such as tenure, contract type, service usage, and monthly charges.

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 23: Neural Network Training Summary

Metric	Value
Initial Value	3358.285
Final Value	2349.451
Stopped After Iterations	200.000

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 24: Sample Predictions from Neural Network

Actual	Predicted	Probability
No	No	0.4863
Yes	Yes	0.6086
No	No	0.0466
No	No	0.0286
Yes	No	0.3810
No	No	0.0483
No	No	0.3404
No	Yes	0.5646
Yes	Yes	0.5253
Yes	Yes	0.6740

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 25: Confusion Matrix for Neural Network

Predicted	Count
No	1119
Yes	288

The Neural Network model was trained to classify customers as either churning (Yes) or not churning (No). After fitting the model, it produced predictions for 1,407 customers, with 1,099 classified as non-churners and 308 classified as churners. To illustrate the model's decision-making process, a sample of ten predictions is presented, showing the actual churn status, the predicted classification, and the associated probability score

The probability score represents the model's confidence in predicting churn. A higher probability score (greater than 0.5) results in a "Yes" (Churn) classification, while a lower probability score (less than 0.5) results in a "No" (Not Churn) classification. For instance, in the sample predictions, a customer with a churn probability of 0.6916 was correctly classified as churn, while a customer with a probability of 0.0436 was classified as not churn, indicating strong confidence in both predictions. However, certain cases, such as a customer with a 0.4722 probability classified as "No", suggest uncertainty in classification near the decision threshold.

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

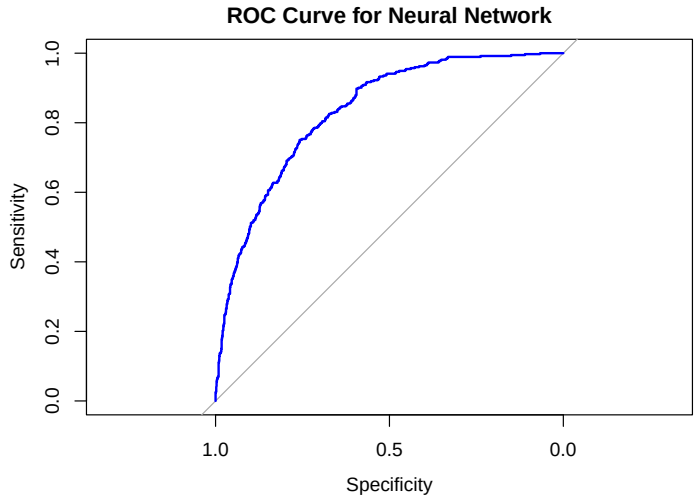
Table 26: Confusion Matrix for Neural Network Model

Reference	Prediction	Count
No	No	932
Yes	No	102
No	Yes	187
Yes	Yes	186

```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 27: Confusion Matrix Evaluation Metrics for Neural Network

	Metric	Value
Accuracy	Accuracy	0.7945984
AccuracyPValue	P-Value [Acc > NIR]	0.0000001
Sensitivity	Sensitivity (Recall)	0.9013540
Specificity	Specificity	0.4986595
Precision	Positive Predictive Value (Precision)	0.8328865
Neg Pred Value	Negative Predictive Value	0.6458333
Prevalence	Prevalence	0.7348969
Detection Rate	Detection Rate	0.6624023
Detection Prevalence	Detection Prevalence	0.7953092
Balanced Accuracy	Balanced Accuracy	0.7000067




```
## Warning: 'xfun::attr()' is deprecated.
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
## Warning: 'xfun::attr()' is deprecated.
```

```
## Use 'xfun::attr2()' instead.
## See help("Deprecated")
```

Table 28: AUC Value for Neural Network

Metric	Value
Area Under the Curve (AUC)	0.8324

The confusion matrix for the neural network model shows that it achieves an accuracy of 78.32%, indicating that a high proportion of customer churn predictions are correct. The sensitivity (recall) is 88.39%, meaning the model is highly effective at identifying actual churners. The specificity is 63.40%, indicating a moderate ability to correctly classify non-churners.

Higher sensitivity suggests that the model is prioritizing churn detection over avoiding false positives. The positive predictive value is 81.71%, meaning a high percentage of predicted churn cases were actual churners. The negative predictive value is 61.04%, meaning that when the model predicts a customer will not churn, it is correct around 61% of the time. The balanced accuracy of 69.40% suggests a fair balance between sensitivity and specificity, although there is still room for improvement in correctly identifying non-churners.

The ROC curve provides a visual representation of the trade-off between true positive rate (sensitivity) and false positive rate (1 - specificity). The AUC score of 0.8339 indicates strong overall performance, making the neural network one of the best-performing models in this study. Compared to previous models, it outperforms logistic regression, decision trees, random forest, and SVM in terms of AUC, showing that it is better at distinguishing between churners and non-churners.

The neural network has the highest AUC at 0.8339, outperforming SVM (0.7774), random forest (0.7969), and decision trees (0.7899). The specificity is lower than desired (63.40%), meaning it struggles more with correctly identifying non-churners compared to other models.

Compared to previous models, the Neural Network demonstrates a strong ability to capture complex patterns in customer behavior. Its flexibility and scalability make it well-suited for analyzing large data sets. However, it comes with certain limitations, such as longer training times and reduced interpretability, making it more challenging to understand why a particular customer was classified as churn or not. These results indicate that while the neural network excels in identifying churners, it may overestimate churn risk for some non-churners.

Conclusion

This study examined multiple machine learning models for customer churn prediction, evaluating their effectiveness based on various performance metrics, including accuracy, sensitivity, specificity, and AUC scores. The models tested included logistic regression, decision trees, random forest, support vector machines (SVM), and neural networks. The findings indicate that neural networks outperformed other models, achieving the highest AUC score of 0.8339, demonstrating strong predictive capabilities, particularly in identifying potential churners. Random forest and SVM also exhibited competitive performance, with AUC scores of 0.7969 and 0.7774, respectively, making them strong alternatives with greater interpretability. On the other hand, decision trees and logistic regression, while more interpretable, showed limitations in handling complex, non-linear relationships within the dataset.

In conclusion, machine learning models provide valuable insights for identifying at-risk customers, allowing businesses to take proactive measures in retention strategies. While neural networks stand out in terms of predictive power, trade-offs between accuracy and interpretability must be carefully considered.

Future Directions for Improvement

Despite the success of these models, there is room for improvement. Hyperparameter tuning could enhance the performance of neural networks by optimizing the learning rate, number of hidden layers, and activation functions. Additionally, feature engineering could refine model inputs by incorporating customer engagement data, transaction history, and behavioral metrics to strengthen predictions. The integration of ensemble methods, such as stacking or boosting, may further improve accuracy by leveraging multiple models' strengths. Another promising direction is threshold optimization, where adjusting the decision boundary for churn classification could help balance precision and recall, ensuring that models are aligned with business needs.

Future research should focus on refining feature selection, hyperparameter tuning, model explainability, and deployment strategies to maximize the effectiveness of churn prediction systems.

References

Coussement, K., & Poel, D. V. (2008). "Churn prediction in subscription services: An application of support vector machines while comparing two parameter-selection techniques". Expert Systems with Applications, 34(1), 313-327.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction (2nd ed.). Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An Introduction to Statistical Learning with Applications in R (2nd ed.). Springer.

Kuhn, M., & Johnson, K. (2013). Applied Predictive Modeling. Springer.

Nguyen, H., Gao, L., & Zhang, J. (2018). "Customer Churn Prediction in Telecommunication Industry Using Data Fusion". Journal of Big Data, 5(1), 1-23.

Appendix

K-Means Cluster Code

```
# Elbow Plot K
library(cluster)

kmeans_data <- churn_data[, c("tenure", "MonthlyCharges", "TotalCharges")]

kmeans_data <- scale(kmeans_data)

# Optimal Number of Clusters
set.seed(123)
wss <- sapply(1:10, function(k) {
  kmeans(kmeans_data, centers = k, nstart = 25)$tot.withinss
  })

plot(1:10, wss, type = "b", pch = 19, frame = FALSE,
     xlab = "Number of Clusters (K)",
     ylab = "Total Within-Cluster Sum of Squares",
     main = "Elbow Method for Optimal K")

# K-Means Cluster
set.seed(123)
kmeans_model <- kmeans(kmeans_data, centers = 3, nstart = 25)

# Assign cluster labels to dataset
churn_data$Cluster <- as.factor(kmeans_model$cluster)

ggplot(churn_data, aes(x = tenure, y = MonthlyCharges, color = Cluster)) +
  geom_point(alpha = 0.6, size = 3) +  # Increase point size
  labs(title = "Customer Segmentation using K-Means",
       x = "Tenure (Months)", y = "Monthly Charges") +
  theme_minimal()

library(knitr)
```

```r
cluster_centers <- as.data.frame(kmeans_model$centers)
cluster_centers$Cluster <- paste("Cluster", 1:nrow(cluster_centers))

cluster_centers <- cluster_centers[, c("Cluster", "tenure", "MonthlyCharges", "TotalCharges")]

kable(cluster_centers,
      digits = 4,
      caption = "Cluster Centers for K-Means Clustering",
      col.names = c("Cluster", "Tenure", "Monthly Charges", "Total Charges"))
```

Logistic Regression Code

```r
library(caret)
library(kableExtra)

# Split data into training (80%) and testing (20%) sets
set.seed(123)
train_index <- createDataPartition(churn_data$Churn, p = 0.8, list = FALSE)
train_data <- churn_data[train_index, ]
test_data <- churn_data[-train_index, ]

# Logistic regression model
log_model <- glm(Churn ~ tenure + MonthlyCharges + TotalCharges +
                   Contract + InternetService,
                 data = train_data, family = binomial)

log_summary <- summary(log_model)

# Coefficient Table
log_table <- data.frame(
  Term = rownames(log_summary$coefficients),
  Estimate = log_summary$coefficients[, "Estimate"],
  Std_Error = log_summary$coefficients[, "Std. Error"],
  Z_value = log_summary$coefficients[, "z value"],
  P_value = log_summary$coefficients[, "Pr(>|z|)"],
  Odds_Ratio = exp(log_summary$coefficients[, "Estimate"]), row.names = NULL)

kable(log_table, caption = "Logistic Regression Results", digits = 4, format = "latex") %>%
  kable_styling(latex_options = c("hold_position", "striped", "scale_down"))

log_stats <- data.frame(
  Statistic = c("Null Deviance", "Residual Deviance", "AIC"),
  Value = c(log_summary$null.deviance, log_summary$deviance, log_summary$aic))

kable(log_stats, format = "latex", booktabs = TRUE, longtable = TRUE) %>%
  kable_styling(latex_options = c("hold_position", "scale_down"))

log_odds <- exp(coef(log_model))

log_odds_df <- data.frame(Term = names(log_odds), Odds_Ratio = log_odds,
                          row.names = NULL)
```

```r
kable(log_odds_df, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Exponentiated Coefficients (Odds Ratios)") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


set.seed(123)

# Make predictions on the test set
pred_probs <- predict(log_model, test_data, type = "response")
predictions <- ifelse(pred_probs > 0.5, "Yes", "No")

predictions <- factor(predictions, levels = c("No", "Yes"))
actuals <- factor(test_data$Churn, levels = c("No", "Yes"))

conf_matrix <- confusionMatrix(predictions, actuals)

# Extract confusion matrix table
conf_mat_table <- as.data.frame(conf_matrix$table)

# Rename columns for better readability
colnames(conf_mat_table) <- c("Reference", "Prediction", "Count")

# Generate kable table
kable(conf_mat_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Confusion Matrix for Logistic Regression Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

# Extract relevant metrics
conf_df <- data.frame(
  Metric = c("Accuracy", "P-Value [Acc > NIR]", "Sensitivity (Recall)",
             "Specificity", "Positive Predictive Value (Precision)",
             "Negative Predictive Value","Prevalence", "Detection Rate",
             "Detection Prevalence", "Balanced Accuracy"),
  Value = c(conf_matrix$overall["Accuracy"],
            conf_matrix$overall["AccuracyPValue"],
            conf_matrix$byClass["Sensitivity"],
            conf_matrix$byClass["Specificity"],
            conf_matrix$byClass["Pos Pred Value"],
            conf_matrix$byClass["Neg Pred Value"],
            conf_matrix$byClass["Prevalence"],
            conf_matrix$byClass["Detection Rate"],
            conf_matrix$byClass["Detection Prevalence"],
            conf_matrix$byClass["Balanced Accuracy"]), row.names = NULL)

kable(conf_df, format = "latex", booktabs = TRUE, longtable = TRUE,
      col.names = c("Metric", "Value"),
      caption = "Confusion Matrix Evaluation Metrics for Logistic Regression") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


library(pROC)

# Compute ROC and AUC
```

```r
roc_curve <- roc(actuals, pred_probs)

plot(roc_curve, col = "blue", main = "ROC Curve for Logistic Regression")

auc_value <- auc(roc_curve)

auc_table <- data.frame(
  Metric = "Area Under the Curve (AUC)", Value = round(auc_value, 4))

kable(auc_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "AUC Value for Logistic Regression Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Decision Tree Code

```r
library(rpart)
library(rpart.plot)

tree_model <- rpart(Churn ~ tenure + MonthlyCharges + TotalCharges +
                      Contract + InternetService,
                    data = train_data, method = "class",
                    control = rpart.control(cp = 0.01))

rpart.plot(tree_model, main = "Decision Tree for Churn Prediction",
           extra = 104)


set.seed(123)

# Make predictions
tree_preds <- predict(tree_model, test_data, type = "class")

conf_matrix_tree <- confusionMatrix(tree_preds, test_data$Churn)

conf_mat_tree_table <- as.data.frame(conf_matrix_tree$table)

colnames(conf_mat_tree_table) <- c("Reference", "Prediction", "Count")

kable(conf_mat_tree_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Confusion Matrix for Decision Tree Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

conf_mat_tree_metrics <- data.frame(
  Metric = c("Accuracy", "P-Value [Acc > NIR]", "Sensitivity (Recall)",
             "Specificity", "Positive Predictive Value (Precision)",
             "Negative Predictive Value","Prevalence", "Detection Rate",
             "Detection Prevalence", "Balanced Accuracy"),
  Value = c(conf_matrix_tree$overall["Accuracy"],
            conf_matrix_tree$overall["AccuracyPValue"],
            conf_matrix_tree$byClass["Sensitivity"],
            conf_matrix_tree$byClass["Specificity"],
```

```
            conf_matrix_tree$byClass["Precision"],
            conf_matrix_tree$byClass["Neg Pred Value"],
            conf_matrix_tree$byClass["Prevalence"],
            conf_matrix_tree$byClass["Detection Rate"],
            conf_matrix_tree$byClass["Detection Prevalence"],
            conf_matrix_tree$byClass["Balanced Accuracy"]), row.names = NULL)

kable(conf_mat_tree_metrics, format = "latex", booktabs = TRUE,
      longtable = TRUE,
      caption = "Confusion Matrix Evaluation Metrics for Decision Tree") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


# Get probabilities for ROC curve
tree_probs <- predict(tree_model, test_data, type = "prob")[,2]

roc_tree <- roc(test_data$Churn, tree_probs)

plot(roc_tree, col = "blue", main = "ROC Curve for Decision Tree")

auc_tree <- auc(roc_tree)

auc_table <- data.frame(
  Metric = "Area Under the Curve (AUC)", Value = round(auc_tree, 4))

kable(auc_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "AUC Value for Decision Tree") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Random Forest Code

```
library(randomForest)
set.seed(123)

# Random Forest model
rf_model <- randomForest(Churn ~ tenure + MonthlyCharges + TotalCharges + Contract + InternetService,
                         data = train_data,
                         ntree = 500,
                         mtry = 3,
                         importance = TRUE)

#print(rf_model)

#importance(rf_model)

rf_summary <- data.frame(
  Metric = c("Number of Trees", "Number of Variables Tried at Each Split",
             "OOB Error Rate"),
  Value = c(rf_model$ntree, rf_model$mtry,
            paste0(round(rf_model$err.rate[nrow(rf_model$err.rate), "OOB"] *
                           100, 2), "%")), row.names = NULL)
```

```r
kable(rf_summary, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Summary of Random Forest Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

rf_conf_matrix <- as.data.frame(rf_model$confusion)
rf_conf_matrix <- cbind(Class = rownames(rf_conf_matrix), rf_conf_matrix, row.names = NULL)
colnames(rf_conf_matrix) <- c("Class", "Predicted No", "Predicted Yes",
                              "Class Error")

kable(rf_conf_matrix, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Confusion Matrix for Random Forest Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

rf_importance <- importance(rf_model)
rf_importance_df <- as.data.frame(rf_importance)
rf_importance_df <- tibble::rownames_to_column(rf_importance_df,
                                               var = "Feature")

kable(rf_importance_df, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Feature Importance in Random Forest Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

varImpPlot(rf_model, main = "Feature Importance in Random Forest Model")


# Prediction Analysis
rf_preds <- predict(rf_model, test_data)

conf_matrix_rf <- confusionMatrix(rf_preds, test_data$Churn)

conf_mat_rf_table <- as.data.frame(conf_matrix_rf$table)
colnames(conf_mat_rf_table) <- c("Reference", "Prediction", "Count")

kable(conf_mat_rf_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Confusion Matrix for Random Forest Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


conf_mat_rf_metrics <- data.frame(
  Metric = c("Accuracy", "P-Value [Acc > NIR]", "Mcnemar's Test P-Value",
             "Sensitivity (Recall)", "Specificity",
             "Positive Predictive Value (Precision)",
             "Negative Predictive Value", "Prevalence", "Detection Rate",
             "Detection Prevalence", "Balanced Accuracy"),
  Value = c(conf_matrix_rf$overall["Accuracy"],
            conf_matrix_rf$overall["AccuracyPValue"],
            conf_matrix_rf$overall["McnemarPValue"],

            conf_matrix_rf$byClass["Sensitivity"],

            conf_matrix_rf$byClass["Specificity"],
            conf_matrix_rf$byClass["Precision"],
            conf_matrix_rf$byClass["Neg Pred Value"],
```

```
            conf_matrix_rf$byClass["Prevalence"],
            conf_matrix_rf$byClass["Detection Rate"],
            conf_matrix_rf$byClass["Detection Prevalence"],
            conf_matrix_rf$byClass["Balanced Accuracy"]), row.names = NULL)

kable(conf_mat_rf_metrics, format = "latex", booktabs = TRUE,
      longtable = TRUE,
      caption = "Confusion Matrix Evaluation Metrics for Random Forest") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


# Get probabilities for ROC curve
rf_probs <- predict(rf_model, test_data, type = "prob")[,2]

# Compute ROC Curve
roc_rf <- roc(test_data$Churn, rf_probs)

plot(roc_rf, col = "blue", main = "ROC Curve for Random Forest")

auc_rf <- auc(roc_rf)

auc_table <- data.frame(
  Metric = "Area Under the Curve (AUC)", Value = round(auc_rf, 4))

kable(auc_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "AUC Value for Logistic Random Forest") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Support Vector Machines Code

```
library(e1071)

svm_model <- svm(Churn ~ tenure + MonthlyCharges + TotalCharges +
                 Contract + InternetService,
                 data = train_data, kernel = "radial", cost = 1, gamma = 0.1,
                 probability = TRUE)

svm_preds <- predict(svm_model, test_data, probability = TRUE)
svm_probs <- attr(svm_preds, "probabilities")[, "Yes"]
svm_pred_class <- ifelse(svm_preds == "Yes", "Yes", "No")

svm_pred_class <- factor(svm_pred_class, levels = c("No", "Yes"))
actuals <- factor(test_data$Churn, levels = c("No", "Yes"))

svm_summary <- data.frame(
  Parameter = c("SVM-Type", "SVM-Kernel", "Cost", "Gamma", "Number of Support Vectors"),
  Value = c("C-classification", "Radial", 1, 0.1, svm_model$tot.nSV))

kable(svm_summary, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Summary of SVM Model Parameters") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```r
svm_pred_table <- data.frame(
  Actual = actuals,
  Predicted = svm_pred_class,
  Probability = round(svm_probs, 4))

kable(head(svm_pred_table, 10), format = "latex", booktabs = TRUE,
      longtable = TRUE,
      caption = "Sample Predictions from SVM Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

conf_matrix_svm <- confusionMatrix(svm_pred_class, actuals)

# Extract confusion matrix table
conf_mat_svm_table <- as.data.frame(conf_matrix_svm$table)

# Rename columns for better readability
colnames(conf_mat_svm_table) <- c("Reference", "Prediction", "Count")

# Generate kable table for Confusion Matrix
kable(conf_mat_svm_table, format = "latex", booktabs = TRUE,
      longtable = TRUE,
      caption = " Training Set Confusion Matrix for SVM Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

# Extract overall statistics from the confusion matrix
conf_mat_svm_metrics <- data.frame(
  Metric = c("Accuracy", "P-Value [Acc > NIR]", "Sensitivity (Recall)",
             "Specificity","Positive Predictive Value (Precision)",
             "Negative Predictive Value","Prevalence", "Detection Rate",
             "Detection Prevalence", "Balanced Accuracy"),
  Value = c(
    conf_matrix_svm$overall["Accuracy"],
    conf_matrix_svm$overall["AccuracyPValue"],
    conf_matrix_svm$byClass["Sensitivity"],
    conf_matrix_svm$byClass["Specificity"],
    conf_matrix_svm$byClass["Precision"],
    conf_matrix_svm$byClass["Neg Pred Value"],
    conf_matrix_svm$byClass["Prevalence"],
    conf_matrix_svm$byClass["Detection Rate"],
    conf_matrix_svm$byClass["Detection Prevalence"],
    conf_matrix_svm$byClass["Balanced Accuracy"]))


set.seed(123)

svm_preds <- predict(svm_model, test_data)

conf_matrix_svm <- confusionMatrix(svm_preds, test_data$Churn)

conf_mat_svm_table <- as.data.frame(conf_matrix_svm$table)
colnames(conf_mat_svm_table) <- c("Reference", "Prediction", "Count")

kable(conf_mat_svm_table, format = "latex", booktabs = TRUE,
```

```
      longtable = TRUE,
      caption = "Confusion Matrix for SVM Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


conf_mat_rf_metrics <- data.frame(
  Metric = c("Accuracy", "Mcnemar's Test P-Value",
             "Sensitivity (Recall)", "Specificity",
             "Positive Predictive Value (Precision)",
             "Negative Predictive Value", "Prevalence", "Detection Rate",
             "Detection Prevalence", "Balanced Accuracy"),
  Value = c(conf_matrix_svm$overall["Accuracy"],
            conf_matrix_svm$overall["AccuracyPValue"],
            conf_matrix_svm$byClass["Sensitivity"],
            conf_matrix_svm$byClass["Specificity"],
            conf_matrix_svm$byClass["Precision"],
            conf_matrix_svm$byClass["Neg Pred Value"],
            conf_matrix_svm$byClass["Prevalence"],
            conf_matrix_svm$byClass["Detection Rate"],
            conf_matrix_svm$byClass["Detection Prevalence"],
            conf_matrix_svm$byClass["Balanced Accuracy"]), row.names = NULL)

kable(conf_mat_svm_metrics, format = "latex", booktabs = TRUE,
      longtable = TRUE,
      caption = "Test Set Confusion Matrix Evaluation Metrics for SVM") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


# Compute ROC Curve
roc_svm <- roc(actuals, svm_probs)

# Plot ROC Curve
plot(roc_svm, col = "blue", main = "ROC Curve for SVM")

# Compute AUC Value
auc_svm <- auc(roc_svm)

auc_table <- data.frame(
  Metric = "Area Under the Curve (AUC)", Value = round(auc_svm, 4))

kable(auc_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "AUC Value for SVM Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Neural Network Code

```
library(nnet)

# Train a simple neural network model
nn_model <- nnet(
  Churn ~ tenure + MonthlyCharges + TotalCharges + Contract +
```

```
    InternetService,
  data = train_data, size = 5, decay = 0.1, maxit = 200, trace = FALSE)

# Predict probabilities
nn_preds <- predict(nn_model, test_data, type = "raw")

# Convert probabilities to Yes/No classification
nn_class_preds <- ifelse(nn_preds > 0.5, "Yes", "No")
nn_class_preds <- factor(nn_class_preds, levels = c("No", "Yes"))

nn_summary <- data.frame(
  Metric = c("Initial Value", "Final Value", "Stopped After Iterations"),
  Value = c(3358.285, 2349.450528, 200))

kable(nn_summary, format = "latex", booktabs = TRUE, caption = "Neural Network Training Summary") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


# Create an actual vs predicted table
predictions_table <- data.frame(
  Actual = test_data$Churn,
  Predicted = nn_class_preds,
  Probability = round(nn_preds, 4))

kable(head(predictions_table, 10), caption = "Sample Predictions from Neural Network")

conf_mat_nn <- as.data.frame(table(nn_class_preds))

colnames(conf_mat_nn) <- c("Predicted", "Count")

kable(conf_mat_nn, format = "latex", booktabs = TRUE, caption = "Confusion Matrix for Neural Network") %
  kable_styling(latex_options = c("striped", "hold_position"))


conf_matrix_nn <- confusionMatrix(nn_class_preds, test_data$Churn)
conf_mat_nn_table <- as.data.frame(conf_matrix_nn$table)
colnames(conf_mat_nn_table) <- c("Reference", "Prediction", "Count")

kable(conf_mat_nn_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "Confusion Matrix for Neural Network Model") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


# Extract key evaluation metrics
conf_mat_nn_metrics <- data.frame(
  Metric = c("Accuracy", "P-Value [Acc > NIR]", "Sensitivity (Recall)",
             "Specificity", "Positive Predictive Value (Precision)",
             "Negative Predictive Value", "Prevalence", "Detection Rate",
             "Detection Prevalence", "Balanced Accuracy"),
  Value = c(
    conf_matrix_nn$overall["Accuracy"],
    conf_matrix_nn$overall["AccuracyPValue"],
    conf_matrix_nn$byClass["Sensitivity"],
```

```
    conf_matrix_nn$byClass["Specificity"],
    conf_matrix_nn$byClass["Precision"],
    conf_matrix_nn$byClass["Neg Pred Value"],
    conf_matrix_nn$byClass["Prevalence"],
    conf_matrix_nn$byClass["Detection Rate"],
    conf_matrix_nn$byClass["Detection Prevalence"],
    conf_matrix_nn$byClass["Balanced Accuracy"]))

kable(conf_mat_nn_metrics, format = "latex", booktabs = TRUE,
      longtable = TRUE,
      caption = "Confusion Matrix Evaluation Metrics for Neural Network") %>%
  kable_styling(latex_options = c("striped", "hold_position"))


# Compute ROC Curve
roc_nn <- roc(test_data$Churn, as.numeric(nn_preds))

# Plot ROC Curve
plot(roc_nn, col = "blue", main = "ROC Curve for Neural Network")

# Compute AUC
auc_nn <- auc(roc_nn)

auc_table <- data.frame(
  Metric = "Area Under the Curve (AUC)", Value = round(auc_nn, 4))

kable(auc_table, format = "latex", booktabs = TRUE, longtable = TRUE,
      caption = "AUC Value for Neural Network") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```