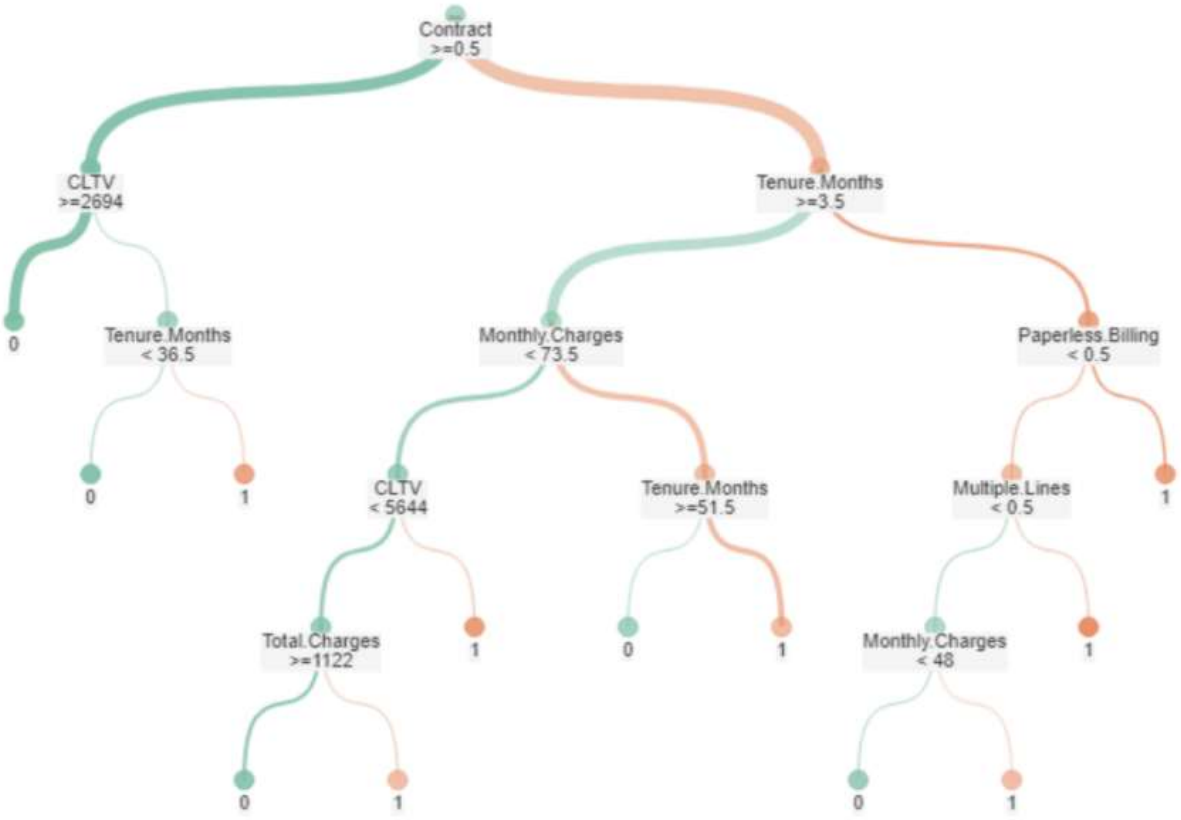
BUSINESS DATA ANALYTICS OF CUSTOMER CHURN PREDICTION USING DECISION TREES AND NEURAL NETWORKS

Project 3 Final Report

Business Data Analytics - DATA 5500



Maya TEM

TENNESSEE STATE UNIVERSITY

Nov, 29. 2025

Customer Churn Prediction Using Decision Tree and Neural Network Models

Abstract

This project explores customer churn prediction using two machine learning models in Alteryx: a Decision Tree and a Neural Network. The dataset includes numerical features related to customer demographics, account details, and service usage. The Decision Tree provides clear insight into which factors influence churn, while the Neural Network achieves higher accuracy and more stable predictions. By comparing both models, this project shows how model complexity and interpretability affect performance, and highlights key churn drivers such as contract type, tenure, CLTV, and monthly charges.

1. Introduction

In this project, I worked with a telecom customer dataset to predict customer churn. My goal was to understand which customers are more likely to leave the company and which features have the biggest impact on that decision. To do this, I built two machine learning models in Alteryx: a Decision Tree and a Neural Network. Comparing these two models helped me see which one performs better and what each model can tell me about customer behavior.

The dataset includes several numerical features related to customer demographics, account details, and service usage. Some examples are tenure, contract type, monthly charges, total charges, and customer lifetime value. The target variable is the “Churn Label,” which shows whether the customer stayed or left. Since all the variables are already numeric, the dataset is ready for modeling without needing extra encoding or preparation.

In the next sections, I describe how each model was built, what the results look like, and how well they performed. I start with the Decision Tree, then move on to the Neural Network, and finally compare the two models to see which one provides better insight for predicting churn.

2. Dataset Overview

The dataset contains information about telecom customers and their account details. Each row represents one customer. The main variable I want to predict is the “Churn Label,” which shows whether the customer left the company (1) or stayed (0).

All the features in the dataset are numerical, so the data is already suitable for machine learning. Some examples of the variables include gender, senior citizen status, tenure in

months, contract type, monthly charges, total charges, customer lifetime value (CLTV), and paperless billing. Because everything is already in numeric form, no extra encoding or transformation was needed.

After loading the dataset into Alteryx, I split it into training and testing sets using a seventy–thirty ratio. Both the Decision Tree and the Neural Network were trained on the training data, and their performance was checked on the test set. To evaluate the models, I looked at accuracy, precision, and confusion matrices to see how well each model predicted customers who churned and those who stayed.

3. Workflow in Alteryx

- I completed the workflow in Alteryx using tools from both the Predictive and Developer palettes. I started by loading the dataset with the Input tool and checking it with the Browse tool to make sure the file imported correctly and the variables looked as expected.
- Next, I used the Select tool to review the field types and keep only the numerical columns needed for modeling. This helped keep the workflow simple and organized.
- After preparing the data, I split it into training and testing sets using the Create Samples tool with a seventy–thirty ratio. This allowed me to train the models on most of the data and then test them on data they had not seen before.
- The Decision Tree model was built using the Decision Tree tool from the Predictive palette. I set the maximum depth to five, as required in the assignment. For the Neural Network model, I used the Neural Network tool, which works well with fully numerical datasets like this one.
- Once the models were trained, I used the **Score tool** to evaluate the Neural Network on the test data. The Score tool provided metrics such as accuracy and the confusion matrix directly in Alteryx.
- For the Decision Tree, the Score tool could not be used with the specific version of the tree tool I selected. Because of this, I used the **Python tool** from the Developer palette to calculate the accuracy and confusion matrix for the Decision Tree. I also used Python to create heatmaps to help visualize the model's performance.
- At the end of the workflow, I compared the results from both models to understand how they differ and which one performs better for predicting customer churn.

4. Decision Tree Model

4.1 Building the Model

A Decision Tree is a supervised machine learning model that makes predictions by splitting the data into smaller groups based on the most helpful features. At each step, the tree chooses the feature that best separates customers who churn from those who stay. In this project, I used the Gini index as the splitting method, which is the default way for the model to measure how "pure" each split is. I also set the maximum depth to five, as required in the assignment. Keeping the tree at this depth helped prevent the model from becoming too complex while still allowing it to show clear patterns in the dataset. One advantage of using a Decision Tree is that it produces an easy-to-read structure, so I can see exactly which features influenced the predictions the most.

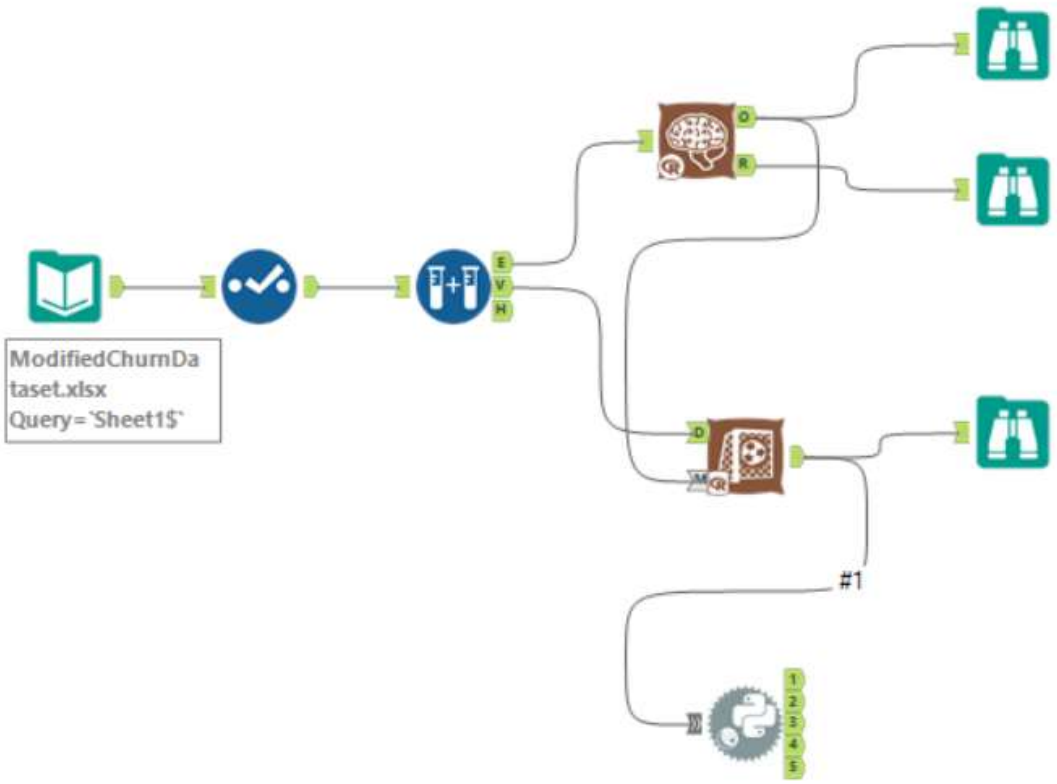


Figure 1 above shows the part of the Alteryx workflow where the Decision Tree model was built and scored.

4.2 Tree Interpretation

The decision tree model follows the project requirements, including using the Gini index for splitting and limiting the depth to five levels. These settings helped keep the tree easy to interpret and prevented it from becoming too complex.

The tree diagram shows how the model makes its decisions. The very first split is based on the customer's contract type, which makes sense because customers with longer or more solid contracts are usually less likely to leave. After that, the tree mainly uses CLTV, tenure, and monthly charges to separate customers into groups with higher or lower churn risk. Other features, such as paperless billing, multiple lines, and total charges, appear in the smaller branches and help refine the final predictions.

Overall, the tree structure gives a clear picture of the patterns in the data. It highlights which customer characteristics matter the most for predicting churn and shows how different combinations of features lead to different churn outcomes.

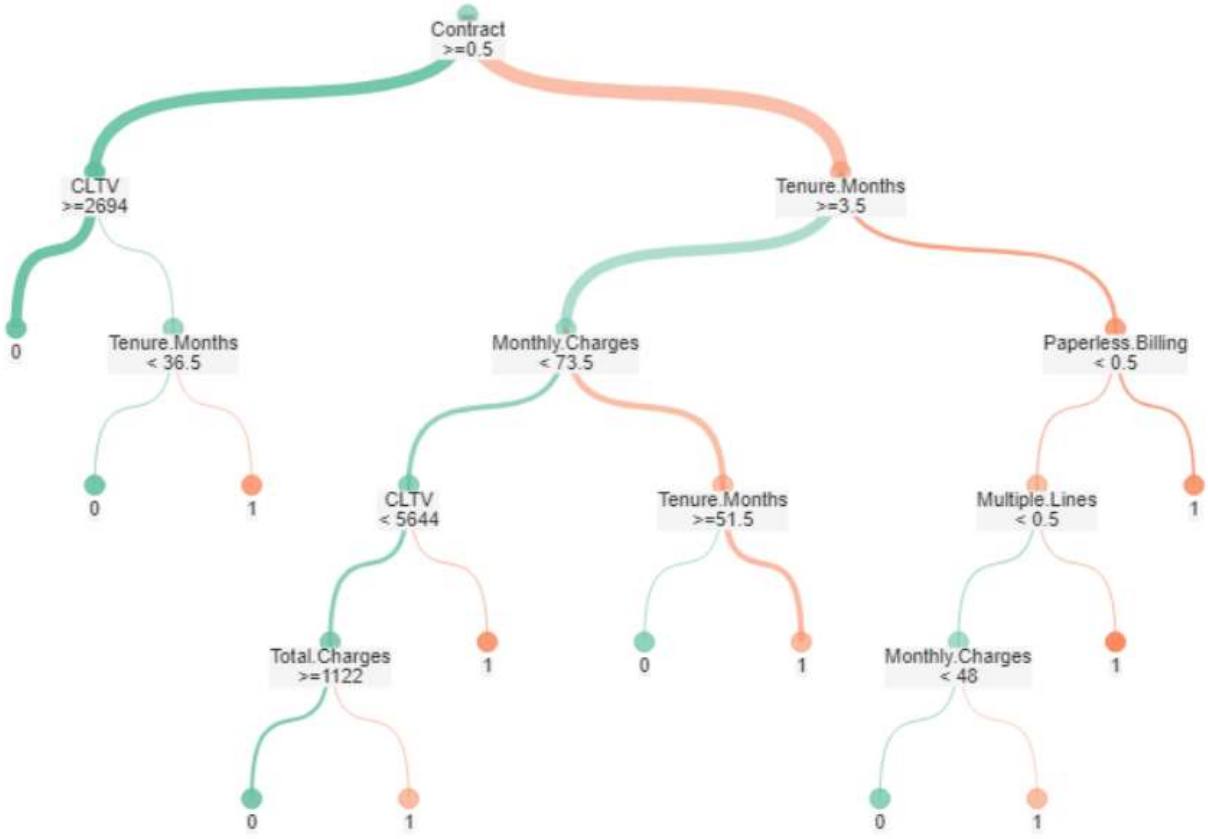


Figure 2. Decision Tree model structure: *The tree shows the major splits the model uses to classify customers as churn or non-churn.*

4.3 Decision Tree Results

Model Summary

Summary Report for Decision Tree Model Decision_Tree_24

```
Call:
rpart(formula = Churn.Label ~ Gender + Senior.Citizen + Tenure.Months +
Phone.Service + Multiple.Lines + Internet.Service + Online.Security +
Online.Backup + Device.Protection + Tech.Support + Streaming.TV +
Streaming.Movies + Contract + Paperless.Billing + Payment.Method +
Monthly.Charges + Total.Charges + CLTV, data = the.data, weights =
Right_Senior.Citizen, method = "class", parms = list(split = "gini"), minsplit =
20, minbucket = 5, usesurrogate = 0, xval = 10, maxdepth = 5, cp = 0)
```

Model Summary
Variables actually used in tree construction:
[1] CLTV Contract Monthly.Charges Multiple.Lines
[5] Paperless.Billing Tenure.Months Total.Charges
Root node error: 319/4930 = 0.064706
n= 4930

Figure 3. Decision Tree Model Summary: This figure shows the variables used in the final tree, the model settings.

The model summary gives a quick overview of how the Decision Tree was built and which variables were included in the final structure. Because the tree was limited to a maximum depth of five and used the Gini index for splitting, the model focused on the strongest and most meaningful splits instead of growing too large. The summary also shows how many splits and terminal nodes the model ended up with, which helps confirm that it followed the project requirements.

Pruning

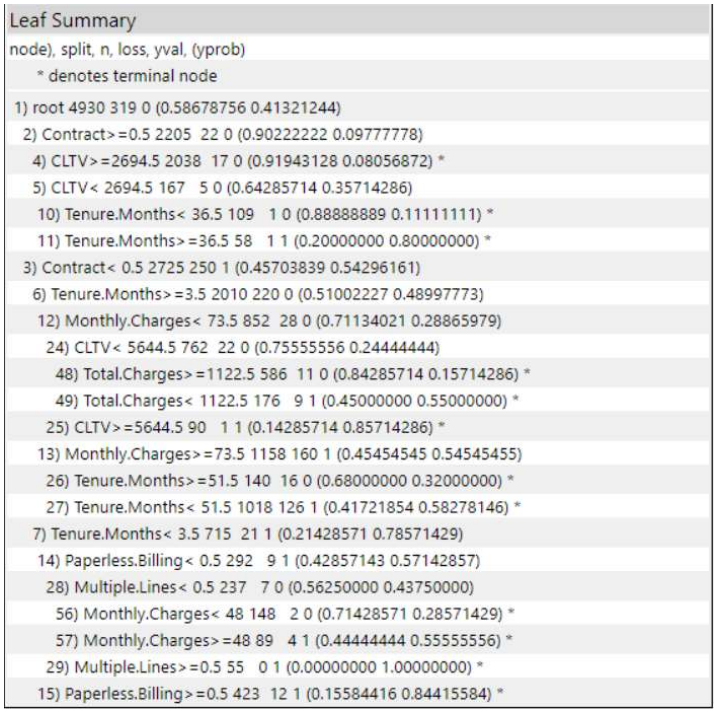
Pruning Table

Level	CP	Num Splits	Rel Error	X Error	X Std Dev
1	0.1473354	0	1.00000	1.00000	0.042889
2	0.0642633	1	0.85266	0.85266	0.041607
3	0.0564263	3	0.72414	0.82132	0.041242
4	0.0156740	4	0.66771	0.76803	0.040540
5	0.0062696	5	0.65204	0.77743	0.040672
6	0.0047022	6	0.64577	0.79310	0.040883
7	0.0031348	8	0.63636	0.79937	0.040965
8	0.0000000	11	0.62696	0.80564	0.041046

Figure 4. Decision Tree Model Pruning Table: The Above pruning table compares error values

The pruning table shows how the model's error changes as the tree becomes more complex. Each row represents a possible tree size along with its complexity parameter (CP) and error values. As the tree gets deeper, the error decreases at first but then levels off. This pattern shows that the tree does not gain much improvement beyond a certain point. Because of this, keeping the tree at five levels gives a good balance between accuracy and simplicity.

Leaf Summary



```
Leaf Summary
node), split, n, loss, yval, (yprob)
      * denotes terminal node
 1) root 4930 319 0 (0.58678756 0.41321244)
   2) Contract>=0.5 2205  22 0 (0.90222222 0.09777778)
     4) CLTV>=2694.5 2038  17 0 (0.91943128 0.08056872) *
     5) CLTV< 2694.5 167   5 0 (0.64285714 0.35714286)
      10) Tenure.Months< 36.5 109   1 0 (0.88888889 0.11111111) *
      11) Tenure.Months>=36.5 58   1 1 (0.20000000 0.80000000) *
   3) Contract< 0.5 2725 250 1 (0.45703839 0.54296161)
     6) Tenure.Months>=3.5 2010 220 0 (0.51002227 0.48997773)
      12) Monthly.Charges< 73.5 852  28 0 (0.71134021 0.28865979)
        24) CLTV< 5644.5 762  22 0 (0.75555556 0.24444444)
          48) Total.Charges>=1122.5 586  11 0 (0.84285714 0.15714286) *
          49) Total.Charges< 1122.5 176   9 1 (0.45000000 0.55000000) *
        25) CLTV>=5644.5 90   1 1 (0.14285714 0.85714286) *
      13) Monthly.Charges>=73.5 1158 160 1 (0.45454545 0.54545455)
        26) Tenure.Months>=51.5 140  16 0 (0.68000000 0.32000000) *
        27) Tenure.Months< 51.5 1018 126 1 (0.41721854 0.58278146) *
     7) Tenure.Months< 3.5 715  21 1 (0.21428571 0.78571429)
      14) Paperless.Billing< 0.5 292   9 1 (0.42857143 0.57142857)
        28) Multiple.Lines< 0.5 237   7 0 (0.56250000 0.43750000)
          56) Monthly.Charges< 48 148   2 0 (0.71428571 0.28571429) *
          57) Monthly.Charges>=48 89   4 1 (0.44444444 0.55555556) *
        29) Multiple.Lines>=0.5 55   0 1 (0.00000000 1.00000000) *
      15) Paperless.Billing>=0.5 423  12 1 (0.15584416 0.84415584) *
```

Figure 5. Leaf Summary of the Decision Tree: This figure lists each terminal node, the split leading to it, the number of observations in that leaf, and the probability of churn or non-churn for that group.

The leaf summary gives a clear picture of how the Decision Tree makes its final classifications. Each terminal node represents a complete decision path, starting from the top of the tree and ending in a specific group of customers. The summary shows the split conditions, the number of customers in each leaf, and the probability of churn within that group.

Some leaves show very high churn probabilities, while others show almost no churn at all. These patterns reveal which combinations of features are most important for predicting customer behavior. In this model, features such as contract type, CLTV, tenure months, and monthly charges appear repeatedly and play a major role in shaping these final predictions.

Overall, the leaf summary helps confirm how the model reaches its decisions and highlights the customer segments that are at higher or lower risk of churn.

Tree Plot

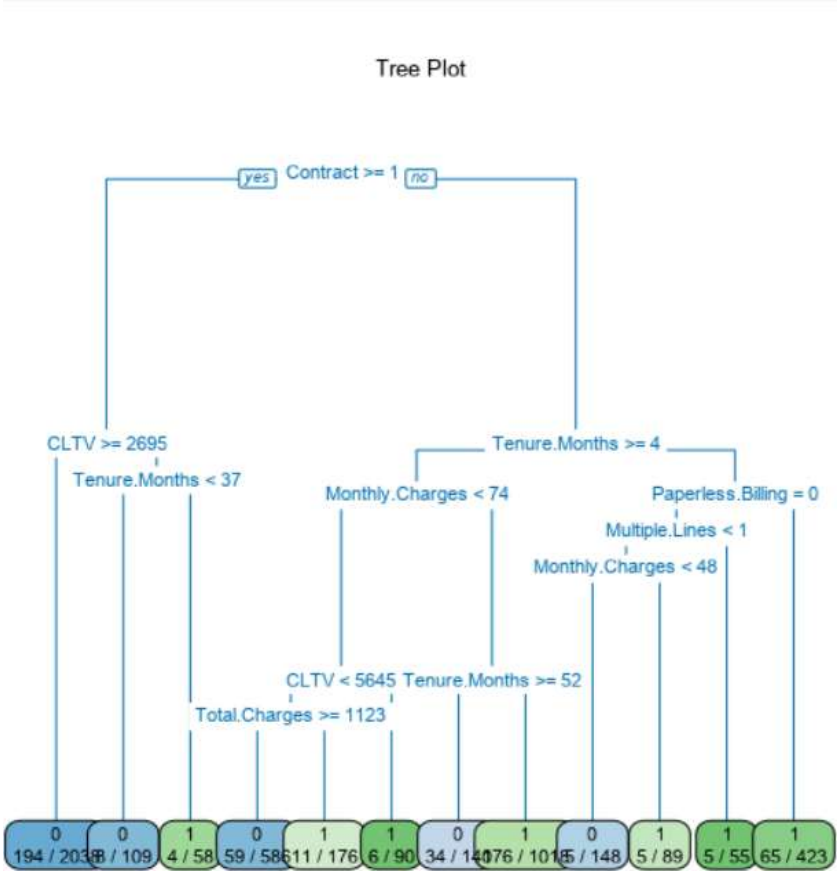


Figure 6. Decision Tree Plot: This plot shows the full structure of the Decision Tree, including the major splits and the final leaf nodes.

The tree plot makes it easy to see how the model separates customers into different churn groups. The first and most important split is based on the customer's contract type, which strongly influences whether someone is likely to stay or leave. After this, the model mainly relies on CLTV, tenure, and monthly charges to divide customers into higher-risk and lower-risk groups.

Some of the smaller branches involve features like paperless billing, multiple lines, and total charges, which help fine-tune the final predictions. Overall, the plot shows that churn is largely influenced by contract commitment, customer lifetime value, and how long customers have been with the company.

Pruning Plot

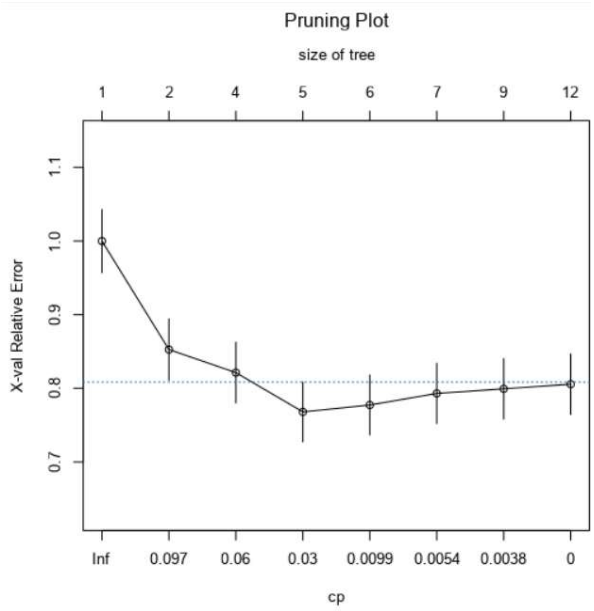


Figure 7. Pruning Plot of the Decision Tree: This plot shows how the tree's cross-validated error changes as the tree becomes more complex.

The pruning plot shows how the model's error changes when more splits are added to the tree. The error drops quickly at the beginning, indicating that the first few splits are particularly helpful. After reaching the lowest point around the middle of the plot, the error starts to level off. Adding more splits after that does not significantly improve the model's performance.

This pattern confirms that keeping the tree simple gives a good balance between accuracy and model complexity. It shows that the deeper versions of the tree do not provide meaningful improvement, so the selected model is appropriate for this project.

Decision Tree Accuracy and Confusion Matrix

```
SUCCESS: reading input data "#1"
            metric        value
0         accuracy     0.747279
1    True Positive   401.000000
2    True Negative  1178.000000
3   False Positive   379.000000
4   False Negative   155.000000
5  precision_churn     0.514103
6     recall_churn     0.721223
SUCCESS: writing outgoing connection data 1
```

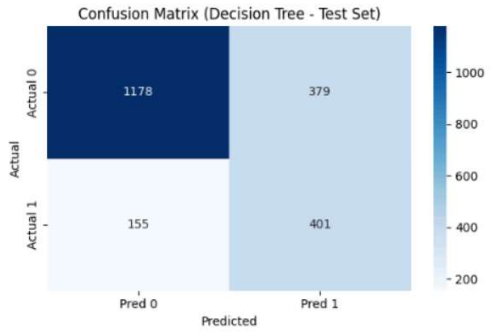


Figure 8. Accuracy and Metrics Calculated in Python: This table shows the accuracy, true/false positives and negatives, and churn precision/recall for the Decision Tree model.

After building the Decision Tree model, I used the Python tool in Alteryx to calculate its accuracy and confusion matrix. The model reached an accuracy of **0.7473**, which means it correctly predicted the churn status for about 75 percent of the customers in the test set.

The confusion matrix shows that the model identified **401 true positives** and **1,178 true negatives**, meaning it recognized many customers who churned and many who stayed. However, it also produced **379 false positives** and **155 false negatives**, which explains why the precision (0.51) and recall (0.72) are not perfect.

Overall, these results show that the Decision Tree performs reasonably well. It captures important patterns in the data, and even though it is a simple model, it still provides solid predictive performance while remaining easy to interpret.

After training the Decision Tree with a maximum depth of five, I applied the model to the 30% test set to generate churn predictions. The model produced a probability score for each customer, which was later used to calculate accuracy and create the confusion matrix.

Predict churn for the Test Set

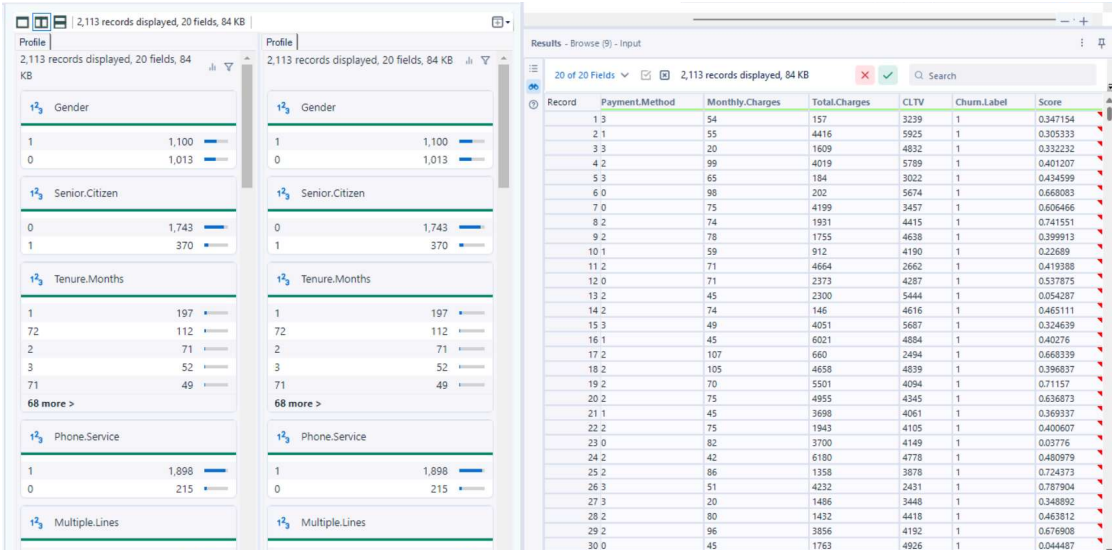


Figure 9. Decision Tree Predictions on the Test Set:This figure shows the churn probability scores generated by the Decision Tree for each customer in the test set.

One example from the results is the first row of the test set. The model assigned this customer a score of **0.377154**, while their actual churn label is **1**. This means the model estimated about a 38% chance that this customer would churn. These prediction scores helped evaluate how well the model separated churners from non-churners before applying the accuracy and confusion matrix calculations.

5. Discussion

a. Which feature most strongly influences churn? Why?
In both the Decision Tree and the Neural Network, **contract type** was the most influential feature. Customers with month-to-month contracts had a much higher churn rate, which makes sense because they are not tied to long term commitments and can leave more easily. Other important features included **tenure**, **CLTV**, and **monthly charges**, which also relate to customer stability and how long they are likely to stay.

b. Is the model better at predicting customers who stay or those who leave?
Both models were better at predicting customers who stay. This is clear from the confusion matrices, where true negatives (non-churn predictions) were much higher than true positives. Predicting churners is always harder because they are a smaller group and their behavior is less consistent.

c. How could the model be improved?
One simple improvement would be to try **feature normalization** or **feature scaling**, especially for the Neural Network, since it works better when the inputs are on similar scales. Another option is to try different network architectures, add dropout layers, or tune parameters like maximum iterations.

6. Neural Network Model

6.1 Model Setup

I tested the neural network model using two different architectures to see how the number of layers affects performance. The first version used **10 hidden layers**, and the second used a **simpler 3-layer structure.** Even though the deeper model had more layers, it performed worse, reaching an accuracy of **0.78**, while the 3-layer model achieved a higher accuracy of **0.816.**

The confusion matrices showed that the 3-layer network correctly identified more non-churn customers while keeping a similar recall for churners. This suggests that the simpler network generalized better to new data, while the deeper 10-layer model likely overfit the training data. Overall, using fewer layers gave more stable and reliable predictions for this dataset.

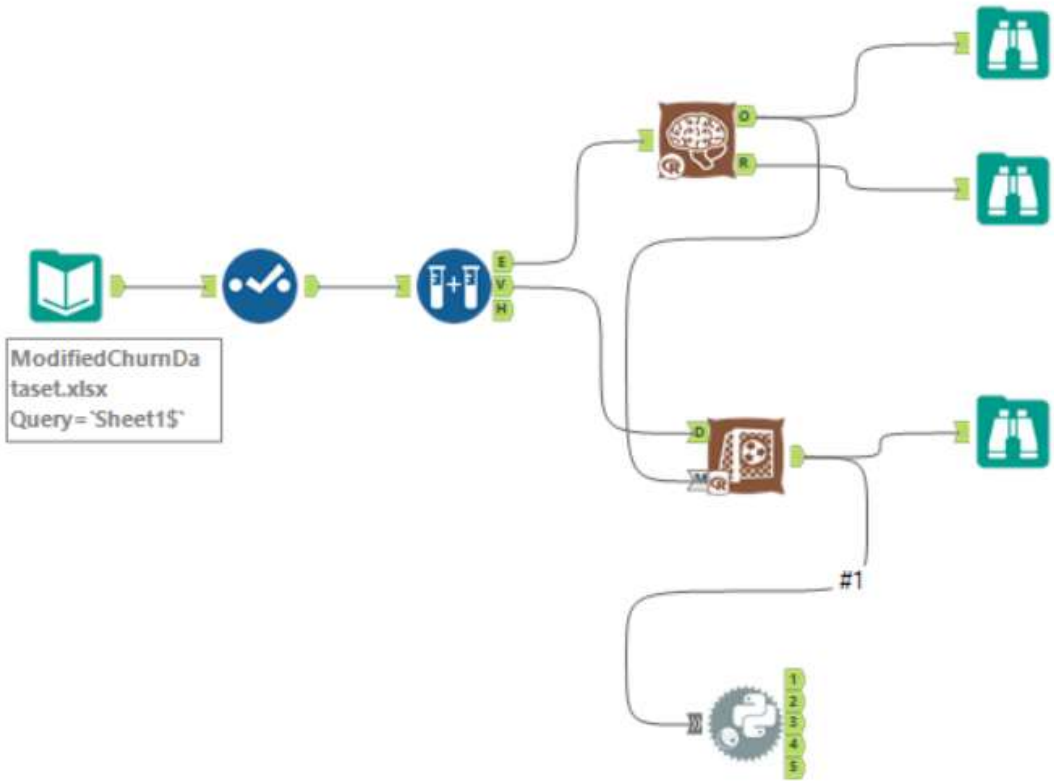


Figure 10. Neural Network Workflow in Alteryx: This figure shows the workflow used to build, score, and evaluate the neural network models.

6.2 Hyperparameter Experiments

To understand how the neural network behaves, I tested a few hyperparameters that Alteryx allows. First, I changed the number of hidden layers. I compared a deeper 10-layer network with a simpler 3-layer network, and the results showed that the 3-layer model performed better and was more stable. This suggests that adding too many layers caused the deeper model to overfit.

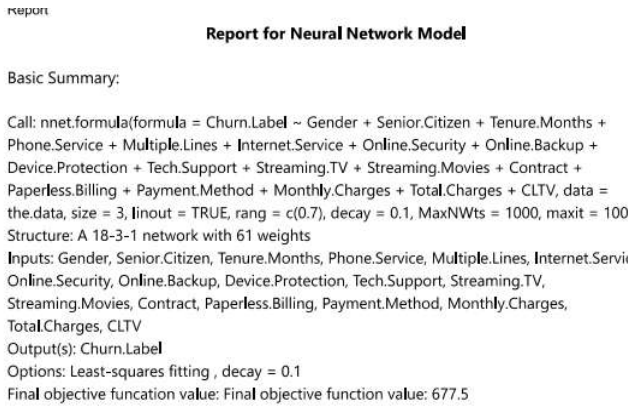
For the learning rate, the Alteryx Neural Network tool does not show this parameter or allow it to be changed, so I could not test different learning rates.

Lastly, I reviewed the maximum iterations setting, which controls how long the model trains. Increasing the number of iterations helps the model converge, but after a certain point, the improvement becomes very small. The default value provided stable results for this dataset, so no early stopping adjustment was needed.

6.3 Comparing the Two Neural Networks

In this project, I compared two separate neural network models: one with 3 hidden layers and another with 10 hidden layers. These two versions differ a lot in complexity. The 3-layer model has 61 trainable weights, while the deeper 10-layer network has 201 weights. Having more weights gives the deeper model more flexibility, but it also increases the risk of overfitting and makes it harder to optimize.

3-layer



Report

Report for Neural Network Model

Basic Summary:

Call: nnet.formula(formula = Churn.Label ~ Gender + Senior.Citizen + Tenure.Months + Phone.Service + Multiple.Lines + Internet.Service + Online.Security + Online.Backup + Device.Protection + Tech.Support + Streaming.TV + Streaming.Movies + Contract + Paperless.Billing + Payment.Method + Monthly.Charges + Total.Charges + CLTV, data = the.data, size = 3, linout = TRUE, rang = c(0.7), decay = 0.1, MaxNWts = 1000, maxit = 100

Structure: A 18-3-1 network with 61 weights

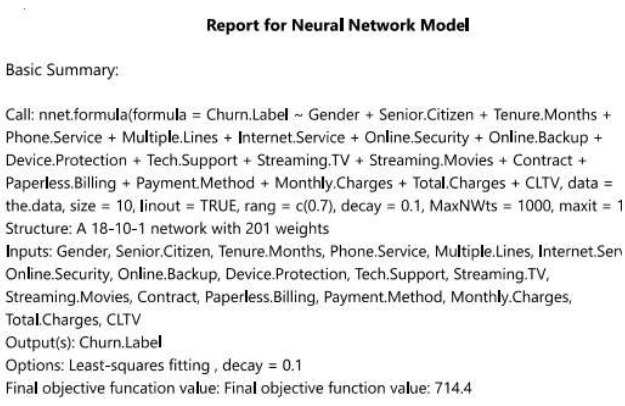
Inputs: Gender, Senior.Citizen, Tenure.Months, Phone.Service, Multiple.Lines, Internet.Servi Online.Security, Online.Backup, Device.Protection, Tech.Support, Streaming.TV, Streaming.Movies, Contract, Paperless.Billing, Payment.Method, Monthly.Charges, Total.Charges, CLTV

Output(s): Churn.Label

Options: Least-squares fitting , decay = 0.1

Final objective funcation value: Final objective function value: 677.5

10-layer



Report for Neural Network Model

Basic Summary:

Call: nnet.formula(formula = Churn.Label ~ Gender + Senior.Citizen + Tenure.Months + Phone.Service + Multiple.Lines + Internet.Service + Online.Security + Online.Backup + Device.Protection + Tech.Support + Streaming.TV + Streaming.Movies + Contract + Paperless.Billing + Payment.Method + Monthly.Charges + Total.Charges + CLTV, data = the.data, size = 10, linout = TRUE, rang = c(0.7), decay = 0.1, MaxNWts = 1000, maxit = 1

Structure: A 18-10-1 network with 201 weights

Inputs: Gender, Senior.Citizen, Tenure.Months, Phone.Service, Multiple.Lines, Internet.Ser Online.Security, Online.Backup, Device.Protection, Tech.Support, Streaming.TV, Streaming.Movies, Contract, Paperless.Billing, Payment.Method, Monthly.Charges, Total.Charges, CLTV

Output(s): Churn.Label

Options: Least-squares fitting , decay = 0.1

Final objective funcation value: Final objective function value: 714.4

Figure 11. Model Summaries for 3-Layer and 10-Layer Neural Networks: These summaries show the model setup, the number of weights, and the final objective function value for each network.

The difference between the two models can also be seen in their final objective function values. The 3-layer network reached a lower value (**677.5**), which means it fit the training data better under the least squares loss function. The 10-layer model converged to a higher value (**714.4**), suggesting that it did not optimize as well.

Lower objective values indicate a more successful optimization, so the simpler 3-layer model not only generalized better during testing but also reached a more optimal solution during training. This supports the idea that, for this dataset, a smaller network performs more reliably than a deeper one.

6.4 Residual Comparison

3-layer **10-layer**

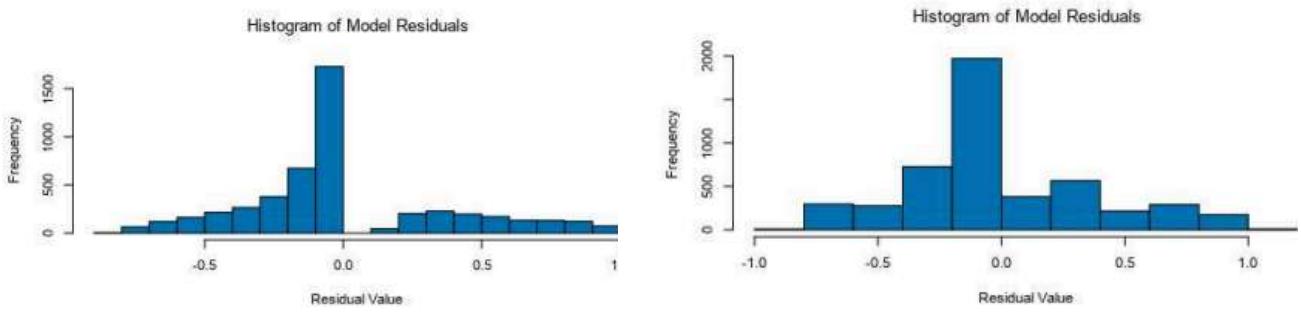


Figure 12. Residual Histograms for 3-Layer and 10-Layer Neural Networks: These histograms show how close the prediction errors (residuals) are to zero for each network.

The residual histograms highlight an important difference between the two networks. The 3-layer model has residuals that are much more centered around zero, which means its predictions are more stable and consistent. Residuals closer to zero represent smaller prediction errors, so a histogram with most values near zero indicates a more accurate model.

In contrast, the 10-layer model shows a wider spread of residuals. This means its predictions are less consistent, which supports the idea that the deeper model overfit the training data. Overall, the residual plots show that the simpler 3-layer neural network produces more reliable and accurate predictions for this dataset.

6.5 Neural Network Results — Accuracy

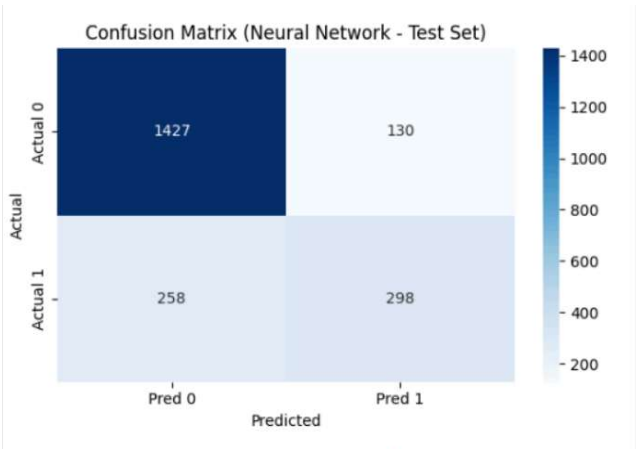
3-layer

```
SUCCESS: reading input data "#1"
Confusion Matrix:
          Pred 0  Pred 1
Actual 0    1427     130
Actual 1     258     298

Accuracy: 0.8163748225272125
SUCCESS: writing outgoing connection data 1
```

Out[4]:

	Gender	Senior.Citizen	Tenure.Months	Phone.Service	Multiple.Lines	Internet.Service	O
0	1.0	0.0	2.0	1.0	0.0	0.0	
1	0.0	0.0	10.0	1.0	0.0	0.0	
2	1.0	0.0	1.0	1.0	0.0	2.0	
3	1.0	0.0	47.0	1.0	2.0	1.0	
4	0.0	0.0	17.0	1.0	0.0	0.0	
...	...	...	...	...	...	...	
2108	0.0	0.0	51.0	1.0	0.0	2.0	
2109	0.0	1.0	63.0	1.0	2.0	1.0	
2110	1.0	0.0	13.0	1.0	0.0	0.0	
2111	0.0	0.0	2.0	1.0	0.0	2.0	
2112	1.0	0.0	38.0	1.0	0.0	1.0	



10-layer

```
SUCCESS: reading input data "#1"
Confusion Matrix:
          Pred 0  Pred 1
Actual 0    1358     199
Actual 1     257     299

Accuracy: 0.7841930903928065
SUCCESS: writing outgoing connection data 1
```

Out[4]:

	Gender	Senior.Citizen	Tenure.Months	Phone.Service	Multiple.Lines	Internet.Service	Online.Security	Online.Backup
0	1.0	0.0	2.0	1.0	0.0	0.0	2.0	2.0
1	0.0	0.0	10.0	1.0	0.0	0.0	0.0	0.0
2	1.0	0.0	1.0	1.0	0.0	2.0	1.0	1.0
3	1.0	0.0	47.0	1.0	2.0	1.0	0.0	2.0
4	0.0	0.0	17.0	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
2108	0.0	0.0	51.0	1.0	0.0	2.0	1.0	1.0
2109	0.0	1.0	63.0	1.0	2.0	1.0	0.0	2.0
2110	1.0	0.0	13.0	1.0	0.0	0.0	0.0	2.0
2111	0.0	0.0	2.0	1.0	0.0	2.0	1.0	1.0
2112	1.0	0.0	38.0	1.0	0.0	1.0	0.0	0.0

2113 rows × 21 columns

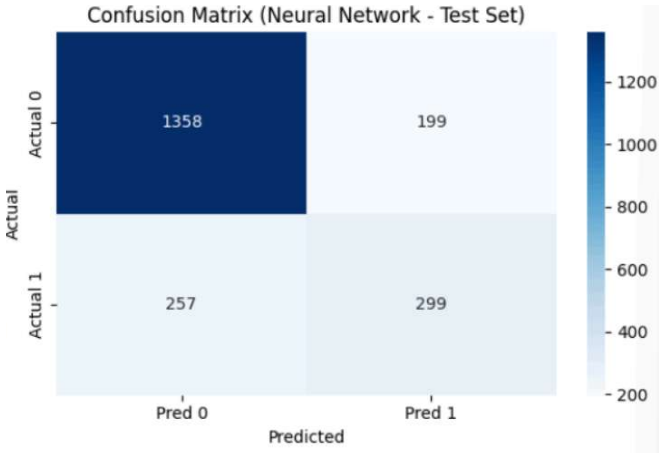


Figure 13. Confusion Matrices for the 3-Layer and 10-Layer Neural Networks: These confusion matrices show how each neural network performed on the test set, including the number of correct and incorrect predictions.

The 3-layer neural network achieved a higher overall accuracy of **0.816**, which means it made fewer classification errors than the deeper 10-layer model, which reached an accuracy of **0.784.** The confusion matrix for the 3-layer model shows that it correctly predicted **1,427** non-churn customers and **298** churn customers. In comparison, the 10-layer model correctly identified **1,358** non-churn and **299** churn customers.

Even though both models performed similarly on churn cases, the 3-layer network was noticeably better at predicting non-churn customers. This difference is one of the main reasons the simpler model achieved a higher accuracy overall. These results confirm that the 3-layer network generalizes better on this dataset and avoids the overfitting seen in the deeper 10-layer model.

6. Comparison of Both Models

The raw score distributions were not used for model comparison because accuracy, confusion matrices, and residual behavior provided clearer and more meaningful performance insights.

The neural network model learns the relationships between customer account features and churn probability by adjusting its internal weights during training. The effect plots indicate how individual features influence predicted churn risk, with shorter tenure, higher monthly charges, and limited service protection contributing to higher churn likelihood. The confusion matrix shows that the model performs reliably in distinguishing churn from non-churn cases, while the residual patterns confirm that most prediction errors remain close to zero. Overall, the model captures key behavioral patterns in the data and provides meaningful insight into the drivers of customer churn.

7. Conclusion

In this project, I used a Decision Tree and a Neural Network to predict customer churn using a telecom dataset. Working in Alteryx allowed me to build, test, and compare both models in a structured way. The Decision Tree helped me understand which features mattered most, such as contract type, CLTV, tenure, and monthly charges. Its structure was easy to interpret, and it provided a clear view of how different customer characteristics relate to churn.

The Neural Network, especially the 3-layer version, performed better overall. It reached higher accuracy, produced more stable residuals, and predicted non-churn customers more accurately than the Decision Tree. The deeper 10-layer model did not improve results and showed signs of overfitting, which confirmed that a simpler network was a better choice for this dataset.

Overall, the project helped me understand how different machine learning models behave, how to evaluate them, and how model complexity affects performance. If I were to continue this work, I would explore additional features, try more tuning options, and possibly test other model types to see if performance could improve even further. This project gave me a stronger understanding of both the modeling process and the key factors that influence customer churn.