Predicting Goodbyes: Evaluating Machine Learning Models for Customer Churn Prediction

Justin Hatch
University Of Oregon CS 472/572
jhatch3@uoregon.edu

ABSTRACT - Customer churn is defined as the loss of a customer or subscriber, and is essential for businesses, as it directly impacts profitability. The ability to accurately predict churn before it happens enables companies to take proactive steps toward customer retention and minimize lost profits. This project aims to evaluate the performance of different machine learning models in predicting whether a customer will leave a service. The models analyzed include Decision Trees, K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Multilayer Perceptron (MLP) Neural Networks. For each model, they will be trained on the training data, optimized on the validation data. The models with the highest validation accuracy will be evaluated on the test data.

I. INTRODUCTION

In 2024, the average American had 4.5 subscription services and spent nearly $1,000 annually for these services [2]. As technology use and online consumerism continue to rise, companies like Netflix, Hulu, and Amazon are not just competing to attract new customers, but also to retain them. According to Harvard Business School Professor Sunil Gupta, “It costs far more to acquire a new customer than it does to retain that customer” [1].

For this reason, accurately predicting which users are likely to leave can help companies reduce costs significantly. Not only is retention more cost-effective than acquisition, but repeat customers also tend to spend more per transaction [4]. Given these trends, predicting customer churn is essential for any subscription-based business looking to maximize profits and reduce losses. This can be approached through customer surveys, analysis of historical data, or by leveraging machine learning models. In this report, several machine learning models are implemented and optimized to classify customer churn. These include Decision Trees, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Multilayer Perceptron (MLP) Neural Networks. Each model is tested using an experimental framework, with tuning applied to optimize their hyperparameters. And the best tuned models will be evaluated based on F1 score.

The remainder of this report includes Section II, background information on the machine learning models used. Section III outlines the methodology, including information on the data, preprocessing steps and evaluation metrics. Section IV presents the experiments conducted and analyzes the performance of each model. Finally, Section V concludes the report by summarizing key findings.

II. Background

A. Decision Trees

The Decision Trees machine is a classification and regression learning model that first gained traction in the 1950s. It works like a diagram made up of a set of rules. You follow the tree from top to bottom, going through each rule (internal node) until you reach a result (a leaf node). The **ID3** algorithm builds a decision tree by splitting at each step based on the feature with the **highest information gain.**

$$\mathrm{H}(S) = \sum_{x \in X} -p(x) \log_2 p(x)$$

Fig 1: Entropy of S - or the measure of uncertainty in the set S

$$IG(S,A) = \mathrm{H}(S) - \sum_{t \in T} p(t)\mathrm{H}(t) = \mathrm{H}(S) - \mathrm{H}(S|A).$$

Fig 2: Information gain, measures the entropy of set S before and after split on feature A.

The decision tree has one parameter, **max depth**. It allows your model to stop splitting when the depth is met. This allows the training to stop before the model overfits. Because the rules of a decision tree can be written out, it's easy to understand why the model makes a certain prediction. This makes decision trees useful in areas like credit card approval and predicting cancer.

B. K-nearest Neighbors (KNN)

The K-nearest neighbors is a classification model that works on the idea that data points that are close together are the same classification. Overall the model does not need to be trained, but predicting takes a lot of time. The **K** is a parameter that refers to the amount of data points you look at nearest to your point, k=5 means you look at the 5 closest points by calculating the euclidean distance and take the count of each classification, the one with the maximum value is the classification of your new point. Changing K allows you to help prevent overfitting.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

Fig 3: Euclidean Distance Function for N-dimensions

C. Support Vector Machines (SVM)

Support Vector Machines find the **most optimal hyperplane** that separates the data into different classes. Optimal means the distance between the hyperplane and each side is the largest it can be. We call this the **margin**. The data points that lay on the margin, are called **support vectors**. And are what are used to classify new unseen points. SVM uses **kernel functions** to transform the data into a higher-dimensional space, allowing SVM to handle non linear data. As well as kernel function, SVM has a parameter **C** that allows us to determine how many misclassifications can happen. Higher C, means less misclassifications and thus a smaller margin. Lower C, means more misclassifications and a wider margin. Lastly, SVM also has **gamma** that determines how much influence a training example has on the decision boundary. High gamma will result in overfitting, and too low of a gamma results in underfitting.

D. Multi Layered Perceptron (MLP)

A multi-layer perceptron (MLP) is a type of neural network composed of multiple layers of **single perceptrons** or neurons. Each perceptron receives input values, multiplies them by **weights**, adds a **bias** term, and then passes the result through an activation function. For example **tanh**, **relu** or **sigmoid**.

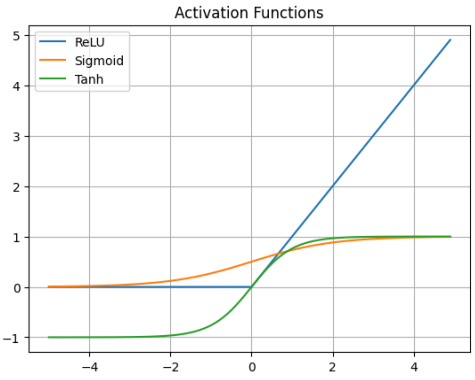

Fig 4. Graph of Relu, Sigmoid, Tanh

This output becomes the input to the next layer.The process continues layer by layer until the final output layer, which generates the network's prediction. In binary classification tasks, the output layer typically contains one neuron with a sigmoid activation function, producing a value between 0 and 1 that can be interpreted as a probability.

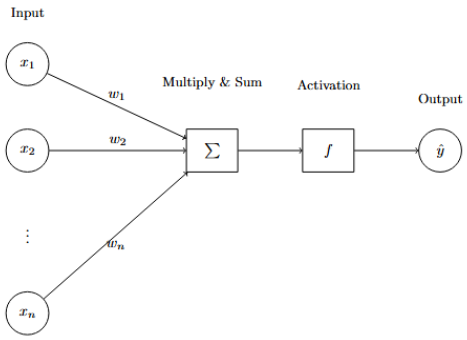

Fig 5: Single Perceptron (neuron)

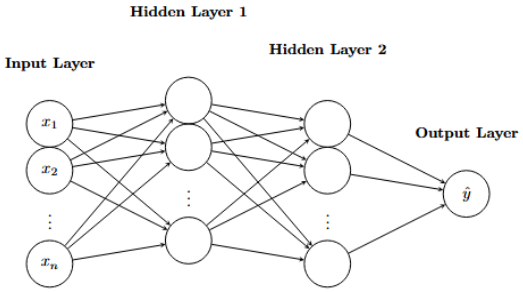

Fig 6: Neural Network - collection of single perceptrons

An MLP (Multi-Layer Perceptron) learns by making predictions and comparing them to the true labels using a loss function, such as **cross-entropy** for classification tasks. To improve its predictions, the network uses **backpropagation** to compute the gradients of the loss with respect to each weight and bias. These gradients are then used to update the parameters using **gradient descent**.

In practice, **stochastic gradient descent (SGD)** is used. Instead of computing gradients over the entire training dataset at once (which can be slow and memory-intensive), SGD updates the model parameters using only a single example or a small mini-batch at each step. This makes training faster and can help the model escape local minima by introducing noise into the updates.

Each complete pass through the training dataset is called an **epoch**. The **learning rate** controls the size of the updates during training. If it's too large, the model may overshoot the optimal values, and if it's too small, training may converge very slowly or get stuck in suboptimal solutions.

E. Churn (History)

Believe it or not, customer retention has a long and rich history. In the United States, it dates back to the 1790s, when merchants began rewarding customers for their loyalty. One notable example came in the mid to late 1800s, when the Great Atlantic and Pacific Tea Company (A&P) introduced one of the first formal customer loyalty programs [5]. Their A&P premium program offered coupons that could be exchanged for luxury items available in their stores, encouraging repeat business.

Today, customer retention remains a top priority for businesses. Many companies have dedicated teams, composed of professionals from Marketing, Customer Support, and Product Management, all focused on improving the customer experience and driving satisfaction. As a result of these efforts, industry leaders like Apple have achieved customer retention rates as high as 90%. [3]

III. Methods

In this section, we outline the methodology used for our experiments. This includes a discussion of the dataset, including how it was sourced, why it was selected, and key preprocessing steps. We also describe the feature selection process, how we reduced the full feature set down to seven core features used for training, and the rationale behind each choice. Furthermore, we explain the selection of classifiers, the hyperparameters tuned for each, and the procedures used to identify the optimal parameter settings. Lastly, we summarize the evaluation strategy and report the performance of the final models.

A. Data Set

We selected a dataset from Kaggle due to its large size with over 500,000 rows, originally split across separate training and testing CSV files. The dataset contained only a single null row, which was removed. For this project, both files were merged to allow for custom data splitting: approximately 80% for training, 10% for validation, and 10% for testing.

Given the computational cost of training on the entire dataset, especially when performing cross-validation, we limited the working set to 100,000 rows. This subset was further split into 81,000 rows for training, 9,000 for validation, and 10,000 for testing. This approach preserved class balance while maintaining manageable resource usage during model development.

B. *Features*

The features in this data set included **CustomerID** - unique value given to each customer, which is also the row number - int. ***Age*** - Age of customer - int. ***Gender*** - gender of customer - category (male, female). ***Tenure*** - how long the customer has been there - int. ***Usage Frequency*** - how often a customer uses in a certain period - int. ***Support Calls*** - how many calls have been made to customers - int. ***Subscription Type*** - Category - Basic, Standard, Premium. ***Contract Length*** - Category - Monthly, Annual, Quarterly. ***Total Spent*** - how much they have spent to date - int. ***Last Interaction*** - days since last interaction - int. ***Churn*** - TARGET - classification 0/1 for not left, left respectively. Churn is what will be classified in the experiment.

The first step was to drop the customerID column, as it provided no predictive value. For the remaining categorical features, one-hot encoding was applied, expanding the dataset to 16 total columns. To select the most informative features, I calculated the correlation of each feature with the target variable and chose the seven with the highest absolute correlation values. Next, I balanced the dataset by ensuring an equal distribution of the target classes (0 and 1), creating a 50/50 class split. This provided a strong and fair baseline for evaluating model performance, as any model achieving around 50% accuracy would indicate no meaningful learning.

Finally, I generated a box plot to check for outliers in the data. The plot revealed that there were no significant outliers, allowing training to proceed without extra steps.

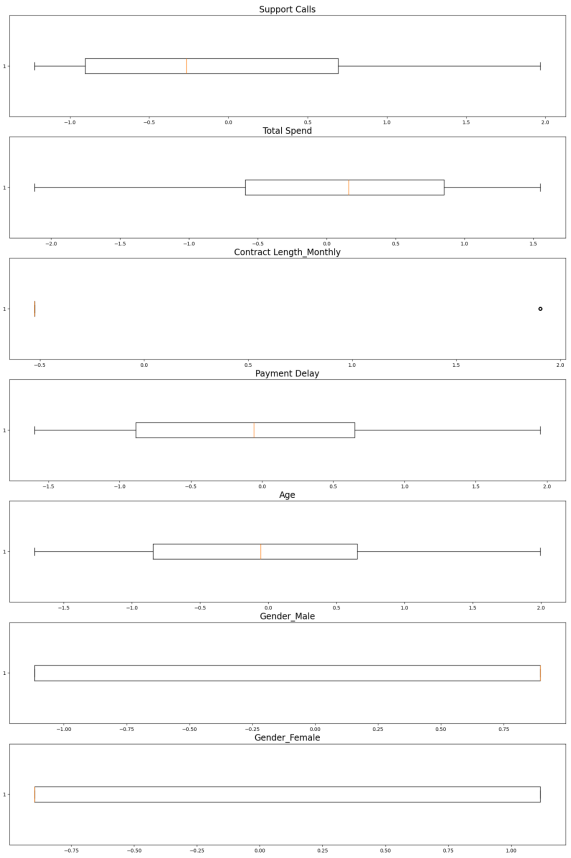

Fig 7. Box plot for selected features

C. Classifiers

I chose the Decision Tree because of its simplicity and ability to handle non-linear data effectively. For this experiment, it has only one tunable parameter, max_depth, making it a strong baseline model to compare others against. I tuned it by fitting models with max_depth values ranging from 1 to 50 and evaluated their accuracy using the validation dataset. The max_depth that yielded the highest validation accuracy was selected as the final setting for the tuned model.

I chose K-Nearest Neighbors (KNN) due to its simplicity, its effectiveness with non-linear data, and the fact that it requires no training phase, making it fast to implement. Similar to the decision tree, KNN only has one tunable parameter for this experiment: K, which determines how many of the closest data points are considered when making predictions. I tuned it by testing integer K values from 1 to 50 and selecting the

one that achieved the highest accuracy on the validation set.

Support Vector Machine (SVM) was selected for its flexibility in handling different types of data through kernel functions and its ability to handle outliers. The parameters tuned for this model include C, which controls the trade-off between margin width and misclassification; gamma, which defines the influence of individual training points; and the kernel function itself, which transforms the data into a higher-dimensional space. Since this model involves multiple parameters, I used scikit-learn's GridSearchCV to perform a grid search across the combinations. The values I tested were C ∈ {0.1, 1, 10, 100}, gamma ∈ {1, 0.01, 0.001}, and kernels including 'linear' and 'rbf'. Grid search uses 5-fold cross-validation, where the training data is split into five subsets. The model is trained on four subsets and validated on the fifth, rotating through all combinations. The parameters that achieved the highest average validation accuracy were selected for the final model.

Finally, I chose the Multilayer Perceptron (MLP) due to its strength in modeling complex relationships and ease of implementation. For this experiment, I tuned the hidden_layer_sizes using several configurations, including (50), (100, 100), (1025, 1025, 1025, 250), and (245, 256, 256). I also tested different activation functions such as tanh, relu, and logistic (sigmoid), along with alpha, which acts as a regularization term and controls the step size of updates during training. The MLP was trained using stochastic gradient descent (SGD) to accelerate convergence and used an adaptive learning rate to reduce the risk of overshooting the global minimum. I also enabled early stopping to terminate training when performance stopped improving, which helped reduce training time. Like the SVM, the MLP model was tuned using grid search with 5-fold cross-validation, and the parameter combination with the highest average validation accuracy was chosen for the final model.

D. Performance (F1 Score)

After tuning each model, we evaluate the best-performing versions using their **F1 scores** to determine which model performs best overall. The **F1 score**, also known as the F-measure, is the harmonic mean of a model's **precision** and **recall**. These two metrics contribute equally to the final score, making the F1 scores a balanced measure of a model's reliability—especially in scenarios with class imbalance.

To better understand the F1 score, it is helpful to understand the **confusion matrix**, which summarizes a classification model's performance. The matrix has two rows and two columns: The top-left cell represents **True Positives (TP)**, cases where the model correctly predicts a positive class (1). The bottom-right cell shows **True Negatives (TN)**, cases where the model correctly predicts a negative class (0). The top-right cell shows **False Positives (FP)**, cases where the model incorrectly predicts a 1 when the true label is 0. The bottom-left cell represents **False Negatives (FN)**, cases where the model incorrectly predicts a 0 when the true label is 1.

Precision measures the proportion of positive predictions that are actually correct, and is calculated as:

$$\textbf{Precision} = \frac{TP}{TP + FP}$$

Recall measures the proportion of actual positive cases that were correctly predicted, and is calculated as:

$$\textbf{Recall} = \frac{TP}{TP + FN}$$

The **F1 score** combines both metrics into a single value using the formula:

$$\textbf{F1} = \frac{2}{(\frac{1}{PRECISION} + \frac{1}{RECALL})}$$

This formulation ensures that both precision and recall are taken into account, and that a model with high values in both will achieve a strong F1 score. This makes it a robust metric for comparing the overall effectiveness of our tuned models.

IV. EXPERIMENTS

In this section we will go over the experiments we conducted and go over the results that I found. First split 100,000 rows into 80,000 training rows, 10,000 validation rows and 10,000 test rows. For decision trees and K-nearest neighbors the experiment will be simpler and thus differ from SVM and MLP.

For decision tree and KNN,

1) Select learner {decision tree, knn}
2) Select hyper parameters to tune for;
    a) **Max depth**
    b) **K values**
3) Train 50 models, one with each parameter value from 1 to 50 against X_train and y_train
4) Predict values using X_val and test accuracy against y_vals
5) Select the model with the highest validation accuracy
6) Predict values using X_test
7) Print classification report from this using y_pred and y_test

For SVM and MLP

1) Select learner {SVM, MLP}
2) Select grid of parameter to tune for
    a) **SVM :**
        i) **C** ∈ {0.1, 1, 10, 100}
        ii) **Gamma** ∈ {1, 0.01, 0.001}
        iii) **Kernel** ∈ {linear, rbf}
    b) **MLP**
        i) **Hidden layer size** ∈ {(50), (25,25,25) (100, 250, 250, 100)}
        ii) **Activation function** ∈ {tanh, relu, logistics (sigmoid)}
        iii) **Alpha** ∈ {0.0001, 0.001, 0.01}
3) Use grid search with 5 cross fold validations and select the parameters that give the highest accuracy
4) Train model with these optimal parameters over 250 epochs and find most optimal max epoch by finding corresponding epoch with highest validation accuracy
5) Predict values using X_test
6) Print classification report from this using y_pred and y_test

V. RESULTS

The most optimal max depth for a decision tree is 7, resulting in a validation accuracy of 92% and test accuracy of 91.75%. The overall F1score was 0.92

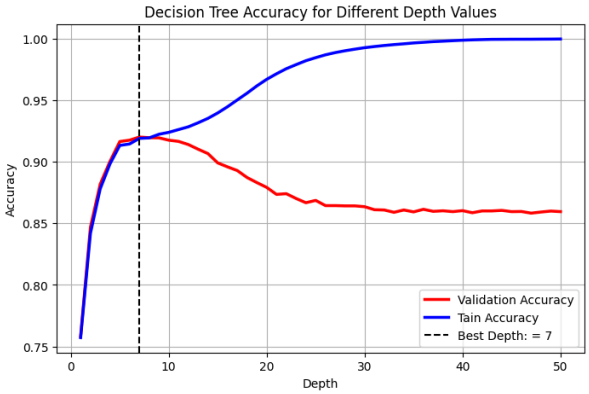

Fig 8. Decision Tree Accuracy for each Depth Values

The most optimal K values for a knn model is 13, resulting in a validation accuracy of 90.66% and test accuracy of 90.52%. The overall F1score was 0.91

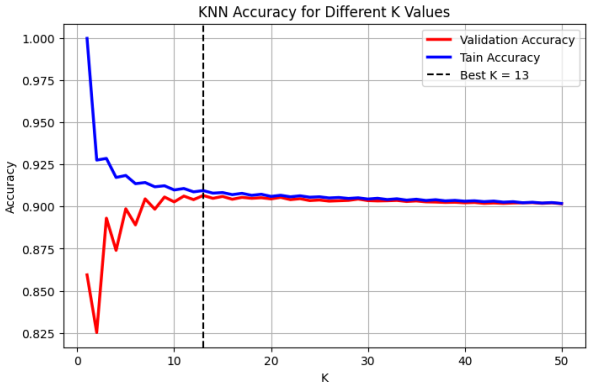

Fig 9. KNN Accuracy for each K value

The most optimal set of parameters for the svm model after running grid search are, C = 10, gamma = 1, kernel function rbf. This resulted in validation accuracy of 91.41% and a test accuracy of 91.19% and a F1 score of 0.91.

The most optimal set of parameters for the MLP model after running grid search are, activation = relu, alpha = 0.0001, hidden_layer_size = (100,250,250,100). Trained over 124 or move epochs. This resulted in a validation accuracy of 91.54 % and test accuracy of 91.70% and a F1 score of 0.92

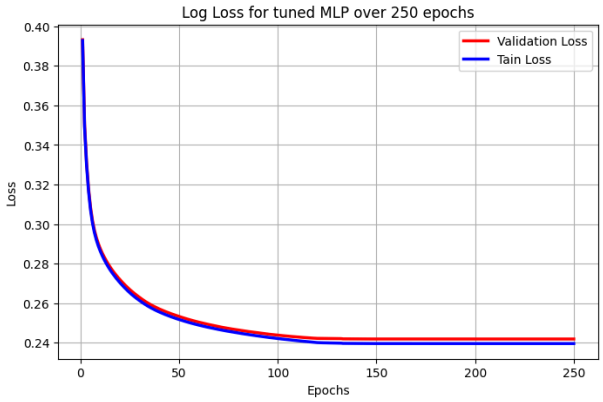

Fig 10 Log loss for tuned MLP model over 250 epoch

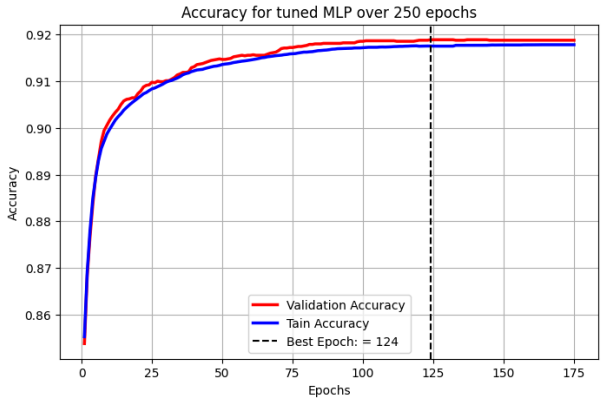

Fig 11. Accuracy for tuned MLP model over 250 epochs

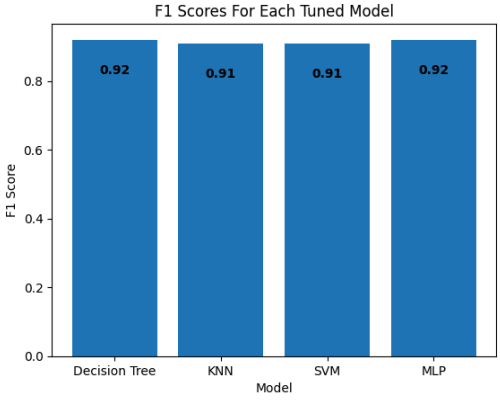

Fig 12 F1 Scores for each tuned model

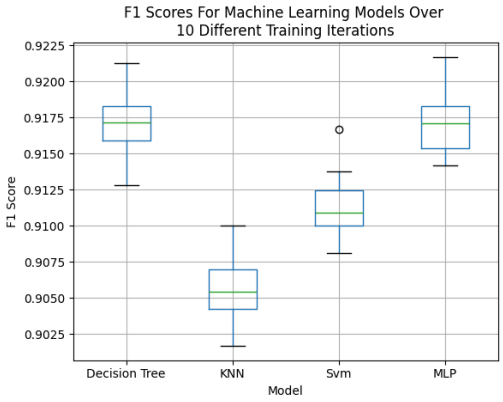

Fig 13. Box plot of F1 scores for each model over 10 iterations of training and testing

Overall, the F1 scores across all models are very similar. While the Decision Tree and MLP models slightly outperformed the others, the differences are not statistically significant. Therefore, no model can be said to outperform the others in predicting customer churn in a meaningful way (see Figure 12). To better evaluate model stability and generalization, each tuned model was trained on ten different random training splits and evaluated using F1 Score. These results are shown in Figure 13 and summarized in Table 1 below:

	Decision tree	**KNN**	**SVM**	**MLP**
Mean F1 Score	0.91717	0.90559	0.91146	0.91713
Standard deviation	0.002353	0.02453	0.002515	0.002221

Table 1: Mean and Standard Deviation of F1 Scores for each model across 10 training/ test

As shown in Table 1, no single model consistently outperforms the others in a significant way. Additionally, the very low standard deviations (around 0.002 for most models) indicate that these models are highly stable across different data splits.

VI. CONCLUSION

In conclusion, none of the models Decision Tree, KNN, SVM, or MLP demonstrated a significant ability to predict customer churn better than the others . Their F1 scores were closely aligned, and differences fell within a negligible range. However, if a model had to be selected, the Decision Tree would be a strong choice due to its interpretability and simplicity. Alternatively, the MLP offers greater flexibility and potential for improvement through additional training and hyperparameter tuning. The consistent performance across all models may be attributed to limitations

within the dataset.

VII. References

[1] *"3 Engagement Strategies You Can Use to Retain Customers."* Business Insights Blog, 26 Mar. 2024, online.hbs.edu/blog/post/how-to-retain-customers. Accessed 26 May 2025.

[2] *"The Average Consumer Pays Nearly $1,000 a Year for Subscriptions - What Do You Pay?"* Yahoo! Finance, Yahoo!, finance.yahoo.com/news/average-consumer-pays-nearly-1-190009547.html?guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAAxrcAAkRAYD1sj6IVadpkTuxlZzt6NmzUv0GcNKbeLurEI_KfNK_9D-CIxyoxMxQym9bvgaaJrY0ah4KO3uxAw0igyl_8VJ0ffexsGtn5twHibbg7cHVEHA99IHhvHJMo7GHq_TPqLwfCzAPAREx_45GjOObKIjKc3IAgdZLbwz&guccounter=2. Accessed 26 May 2025.

[3] *Customer Churn 101: What Is It, Types of Churn, and What to Do about It,* www.paddle.com/resources/customer-churn. Accessed 26 May 2025.

[4] *O'Brien, Keith, and Amanda Downie. "What Is Customer Churn?*" IBM, 16 Apr. 2025, www.ibm.com/think/topics/customer-churn#churn-rate. Accessed 26 May 2025.

[5] *"The Surprising History of Loyalty Programs." White Label Loyalty,* whitelabel-loyalty.com/blog/loyalty/the-surprising-history-of-loyalty-programs/. Accessed 26 May 2025.

[Fig 1] *"Decision Tree Learning." Wikipedia*, Wikimedia Foundation, 6 May 2025, en.wikipedia.org/wiki/Decision_tree_learning. Accessed 26 May 2025.

[Fig 2] *"Decision Tree Learning." Wikipedia*, Wikimedia Foundation, 6 May 2025, en.wikipedia.org/wiki/Decision_tree_learning. Accessed 26 May 2025.

[Fig 3] *"Euclidean Distance." Wikipedia*, Wikimedia Foundation, 30 Apr. 2025, en.wikipedia.org/wiki/Euclidean_distance. Accessed 26 May 2025.