Matthew Urquhart

Virginia Polytechnic Institute and State University  
[Blacksburg](http://www.blacksburg.gov/), VA 24061-0002

[urquhartmatthew@vt.edu](mailto:urquhartmatthew@vt.edu)

1. Introduction

This project focuses on analyzing and predicting customer churn in a telecommunications company. Customer Churn refers to the loss of clients or customers due to a variety of reasons within the first month of activating service – which poses a significant challenge in the telecom industry. The objective is to develop a machine learning model which can accurately predict which customers are likely to churn, which can guide proactive retention measures. Retention measures include targeted promotions, loyalty discounts, and other perks to incentivize customers to remain loyal.

As of June 2024, Verizon Wireless had a churn rate of 1.59%. Out of the hundreds of  
thousands of added connections per month, this represents a significant portion of customers. The problem is fundamentally a binary classification task, where the model classifies customers into two categories: churners and non-churners. The model uses a set of key account features such as having multiple lines, bundling streaming and internet services, and the bill amount to determine the likelihood of a customer leaving within the first month of service. This prediction capability is crucial for business sustainability and customer retention strategies.

Current approaches to churn prediction often rely on basic demographics and usage metrics with limited consideration of the complex interactions between various customer attributes. Traditional methods typically suffer from reactive rather than proactive approaches to customer retention, limited use of advanced machine learning techniques, insufficient integration of multiple data points, and lack of systematic evaluation of prediction accuracy.

A successful churn prediction model would enable telecom companies to proactively identify at-risk customers before they churn, more targeted and effective prevention strategies, reduced customer acquisition costs through better retention (the cost of acquiring a new customer is significantly more than keeping a current customer), improved customer satisfaction through early intervention, and data-driven decision making in customer relationship management.

1. Approach

Data preprocessing is comparable to preparing ingredients before cooking a meal – each step is crucial for the final result. The preprocessing started with the careful examination of the raw telecom data obtained from IBM, much like a chef inspecting his ingredients. This examination allowed for the removal of data that does not directly contribute to customer churn, such as a customer’s account number or the type of services they had (phone or internet). Removing unnecessary information is crucial to ensure it doesn’t distract the machine learning model – just like a chef removes unwanted parts of vegetables in meal preparation. This is one of many steps in ensuring high accuracy in the model’s predictions.

Imagine organizing a complex puzzle where each piece needs to be perfectly shaped – the cleaning process used in this project involved the surgical removal of non-predictive columns in the dataset, careful handling of categorical variables through one hot encoding, transforming text data into a format the model could understand, and elegant handling of the target variable (churn label) through label encoding and the standardization of numerical features, ensuring all the data was on the same scale.

The feature selection and engineering phase was like crafting a lens through which the model would view customer behavior. I selected features that tells the customer’s story (from demographic details and service usage patterns), transformed categorical features into numerical formats while preserving their meaning, and created the balanced split between training (80%) and testing (20%) data, ensuring the model could learn effectively while still being tested fairly.

The approach used for model development in this project followed a systematic progression from simpler models to more sophisticated algorithms, with each step informing my understanding of the problem space. I explored various algorithms including logistic regression, decision trees, random forest, and XGBoost. My investigation led me to choose the XGBoost classifier as the primary model – a choice that emerged from a careful evaluation of each algorithm’s strengths and limitations in the context of telecom customer churn prediction.

The selection of XGBoost was driven by several compelling factors. First, its gradient boosting framework showcased remarkable abilities to capture complex patterns in customer behavior, particularly the subtle interactions between different service usage patterns and churn probability. Second, the algorithm’s built-in mechanisms for handling imbalanced data proved invaluable, given that churn customers typically represent a minority class in telecommunications datasets. Finally, XGBoost’s superior speed and performance characteristics made it particularly suitable for practical deployment in a business environment, where rapid prediction updates might be necessary.

The model’s configuration process resembled the fine-tuning of a sophisticated instrument. Comprehensive logging metrics were implemented to track the model’s performance during the training process, allowing me to monitor its learning and make necessary adjustments. The implementation of early stopping mechanisms proved crucial in preventing overfitting, ensuring the model would generalize well to new customer data rather than memorizing patterns in the training set.

1. Experiments and Results

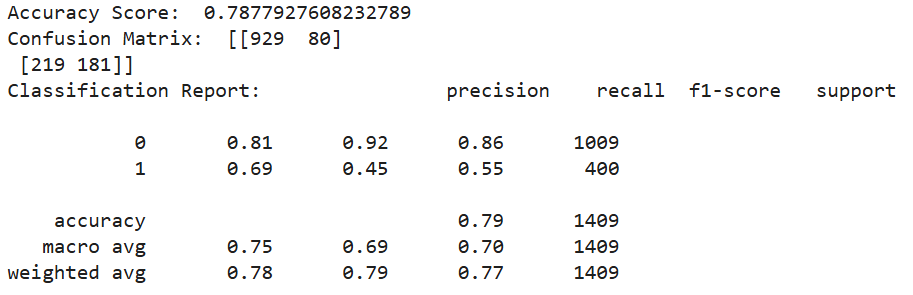
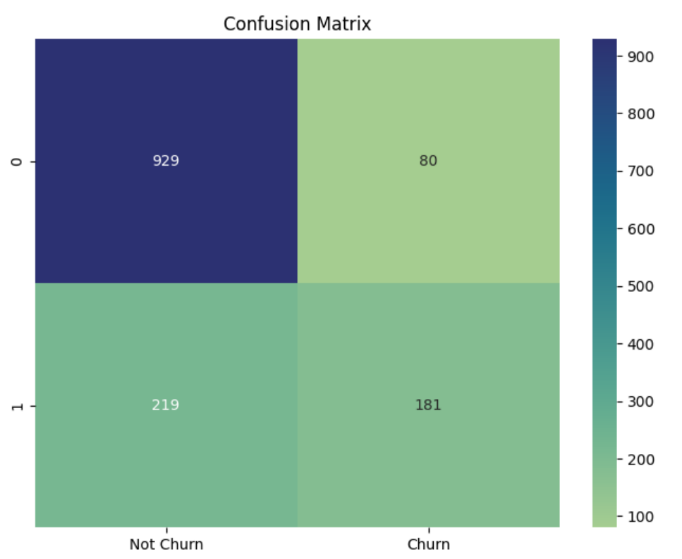
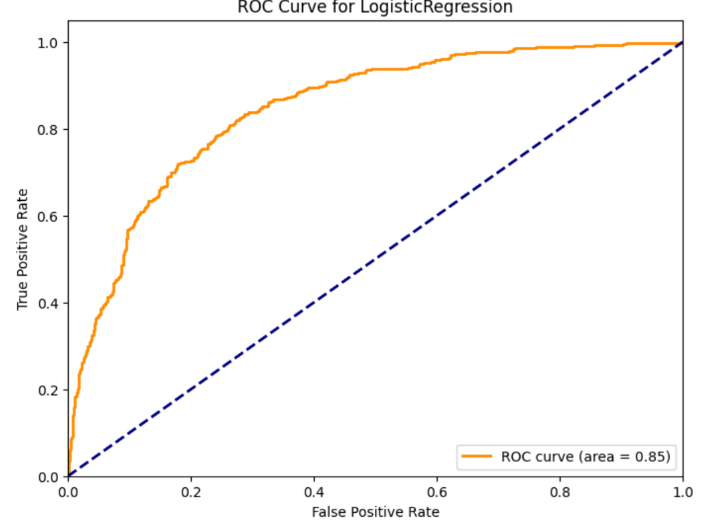
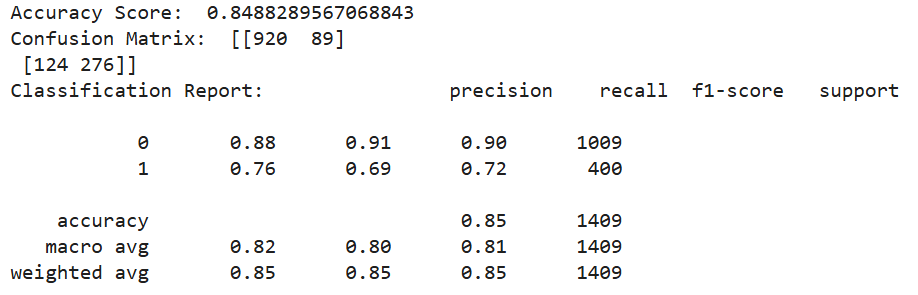
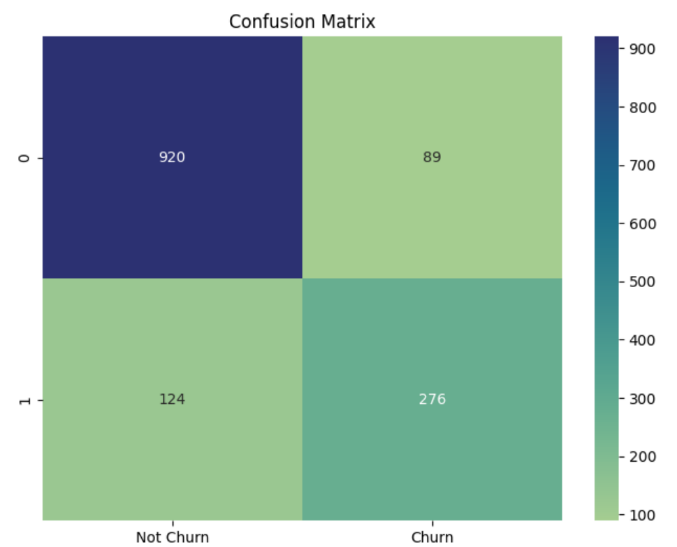
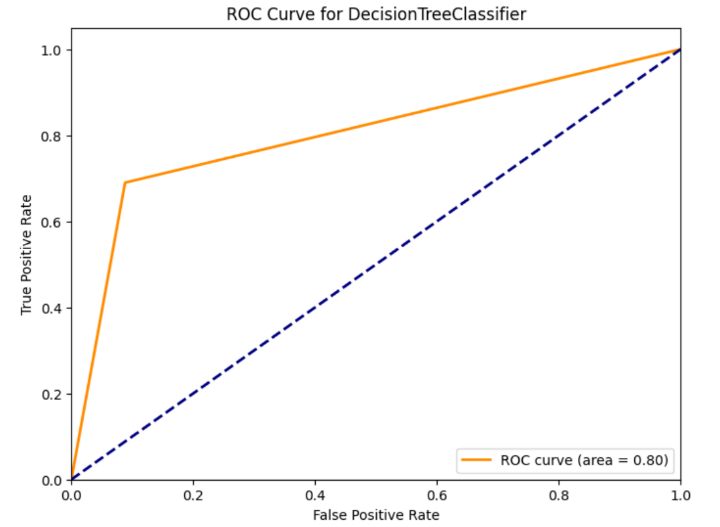
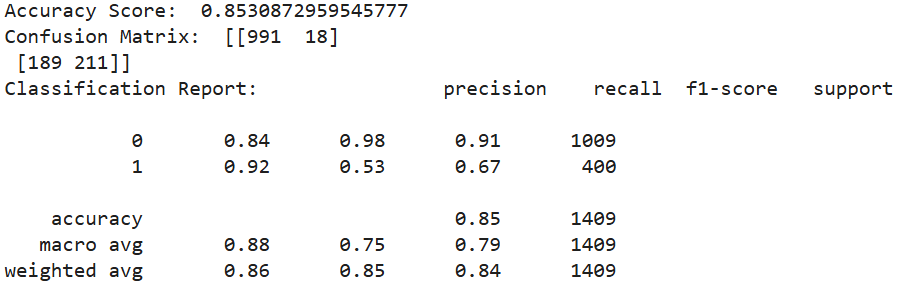
The logistic regression model was used as a starting point, mainly for its interpretability and established track record in binary classification tasks. The implementation used scikit-learn’s logistic regression class with carefully tuned parameters. The model achieved an accuracy of 78.77%, providing a solid baseline but highlighting the need for more sophisticated approaches. Accuracy score measures the proportion of correct predictions (both true positives and true negatives) out of all predictions. Logistic regression received an ROC value of .85, meaning that the model has an 85% probability of correctly ranking a randomly chosen positive instance (a customer who churns) higher than a randomly chosen negative instance (a customer who does not churn).

Figure Set One – Linear Regression Model Results

Decision Trees was implemented next using the classification and regression tree algorithm through the scikit-learn’s decision tree classifier class. The model was initially configured with default parameters to establish a pure tree-based baseline. The decision tree achieved an accuracy of 84.88%, and an ROC score of 80%. These results demonstrate a better handling of non-linear relationships in the dataset compared to logistic regression.

Figure Set 2- Decision Tree Model Results

The Random Forest classifier was another step forward in model sophistication. Using the scikit-learn random forest classifier class with 100 estimators, the power of ensemble learning was leveraged. The implementation included n\_estimators = 100, random\_state = 42 (for reproducibility) and balanced class weights to handle class imbalances. The Random Forests achieved slightly better results over the decision trees, with an accuracy of 85.31% an ROC score of 94%, showing improved stability compared to the single decision trees. The ensemble nature of random forests helped mitigate the overfitting observed in individual decision trees, while capturing complex feature interactions.



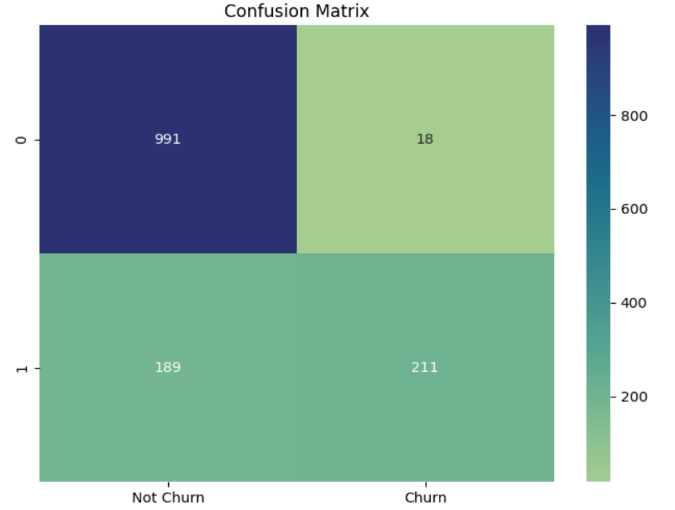
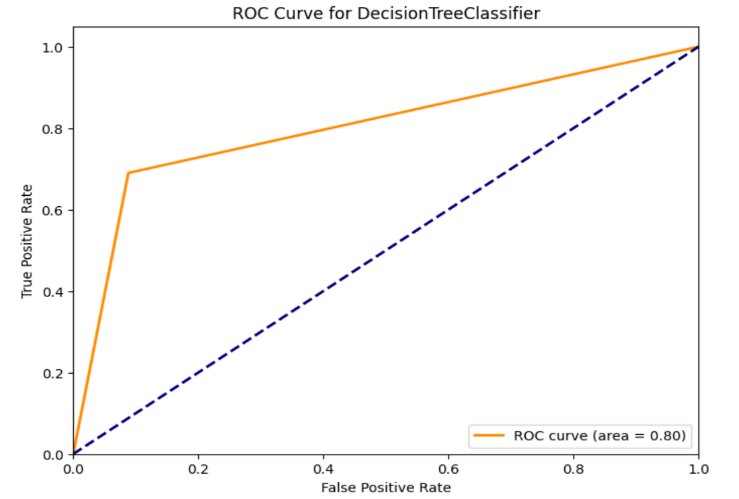


Figure Set 3 – Random Forest Model Results

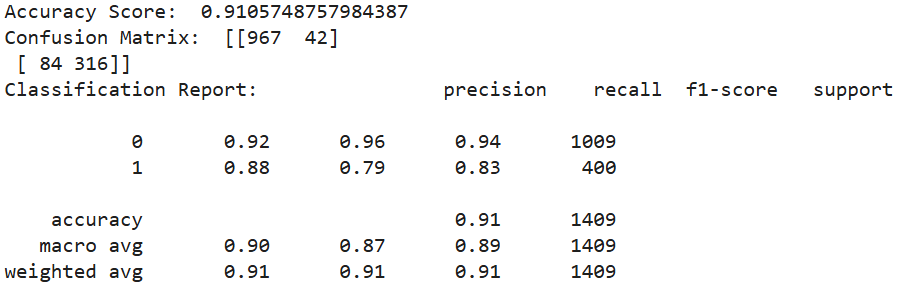
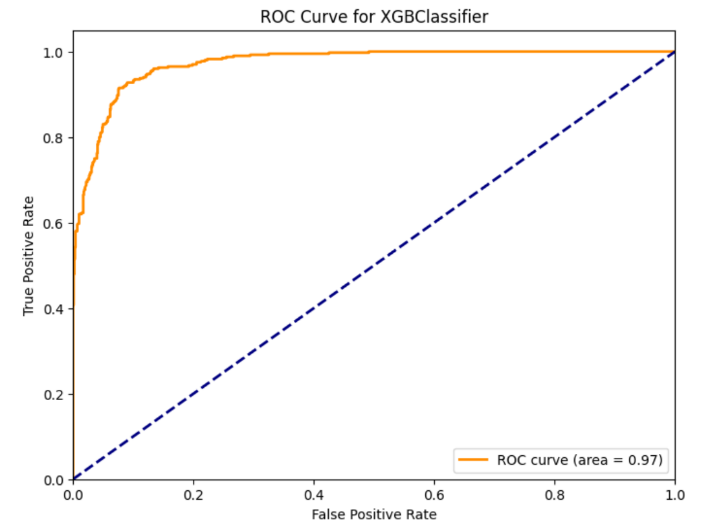
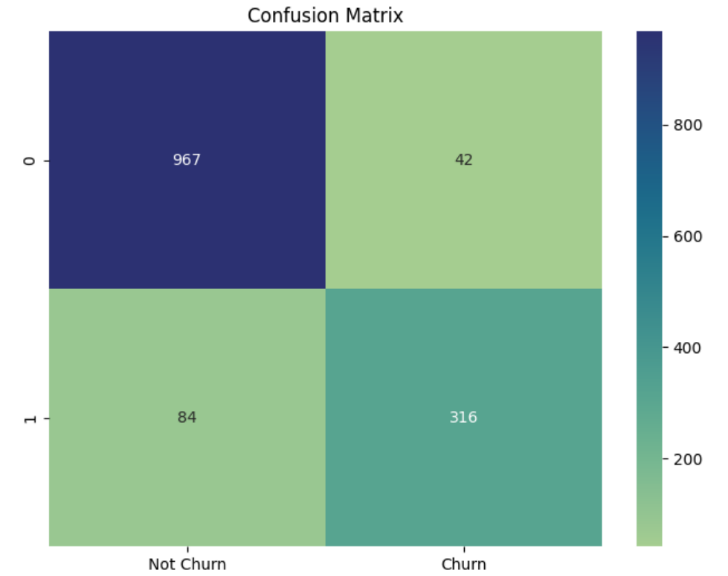
XGBoost emerged as the best performing model, showing superior capabilities across all metrics. Key configurations in this implementation include eval\_metric = ‘logloss’ (for training optimization), early stopping to prevent overfitting, learning rate of 0.1, and maximum depth of 6. The results were compelling, with an overall accuracy of 91.06%, precision of 92% for non-churners, and 88% for churners, a recall of 96% for non-churners, and 79% for churners. The model achieved an astounding ROC score of 97%. The XGBoost sequential tree building process proved especially effective at capturing subtle patterns in customer behavior. The algorithm’s built in feature importance calculation helped identify and leverage the most predictive attributes of customer churn. Also, the XGBoost native handling of missing values proved advantageous given the real-world nature of this telecommunications dataset. It’s built in regularization techniques effectively prevented overfitting, while maintaining strong predictive powers. XGBoost’s superior performance justifies its selection as my final model for deployment.

Figure Set 4 – XGBoost Model Results

1. Availability

My code is available through the public GitHub repository linked below. I recommend running the code on Google Colab for compatibility and efficiency. The dataset is also included in the repository.

Link: <https://github.com/MU386997/Telco-Customer-Churn.git>

1. Reproducibility

The results of this project can be reproduced easily. The Python code constructing the model along with the dataset is contained in the public GitHub repository linked below. Download the code from the repository and paste it into a Google Colab project (preferred) or any IDE of your choice. If using Google Colab, upload the dataset to the project files and run the code. If using another IDE, modify the file path to the stored location of the dataset on your machine (filename is the variable name, and it is at the very bottom of the project), and then run the code. You may choose to run all 4 different models attempted at once (logistic regression, decision trees, random forests, and XGBoost), or you may comment out the models you don’t wish to run in the “models” array in the ”test\_model” function. By default, only XGBoost is enabled, since that was the optimal model and the one chosen for the final project. The exact graphs shown in this document will be displayed automatically. This report is also contained in the repository as a description of the project.

1. Conclusion

The development process through this churn prediction project revealed not just technical success, but practical insights into customer behavior. The model’s robust performance (91.06% accuracy) isn’t just a statistical achievement – it’s a foundation for more proactive customer retention strategies. In this project, we developed a highly accurate prediction system, created balanced performance across customer categories, built a robust and reproducible modeling framework, and established clear visualization protocols. Future horizons include exploring more sophisticated feature engineering (feature evolution), combining multiple models for even better performance (model ensemble development), moving towards dynamic prediction systems (real-time implementation), incorporating business cost into model’s decision-making (cost sensitive enhancement), and developing more interactive and insightful visual tools. This project isn’t just about predicting churn – it’s about understanding and improving customer relationships through the power of AI and data science.

References

[1] “Telco customer churn: IBM dataset,” *www.kaggle.com*. https://www.kaggle.com/datasets/yeanzc/telco-customer-churn-ibm-dataset

[2] “Telco customer churn,” *www.ibm.com*. https://www.ibm.com/docs/en/cognos-analytics/11.1.0?topic=samples-telco-customer-churn

THE END