Data Mining Project Report

Telco Customer Churn Dataset

Zhiqi Yang, 2110635
Elise Wolf, 1828204
Xi Liu, 2113820
Shiqi Zhou, 2036915
Yenan Chen, 2113612

December 7, 2024

Submitted to
Data and Web Science Group
Prof. Dr. Sven Hertling
University of Mannheim

Contents

1 Business and Data Understanding

This project addresses customer churn in the telecommunications industry, aiming to develop a predictive model to identify at-risk customers. By leveraging machine learning, telecom companies can implement targeted retention strategies to reduce churn rates. The binary target variable, *Churn*, helps optimize resource allocation, improve customer satisfaction, and enhance business outcomes through data-driven decisions, ultimately boosting profitability and customer loyalty.

The dataset combines six sources, including demographics, location, services, status, and churn data, resulting in 7,000 customer records with approximately 20 features. Key features include demographic details (e.g., *Gender*, *Senior Citizen*), account information (e.g., *Tenure*, *Contract Type*), and usage metrics (e.g., *Monthly Charges*, *Total Charges*). The target variable, *Churn*, is imbalanced, with 26.5% of customers labeled as churned. This imbalance requires careful handling during model training to ensure predictive fairness. The dataset provides valuable insights into customer demographics, behaviors, and their relationship with the telecom company.

2 Preprocessing

To prepare the Telco Customer Churn dataset for modeling, we followed a structured preprocessing pipeline. Multiple datasets were integrated, missing values imputed, and columns converted for consistency. Categorical features were encoded, and numerical features scaled. Exploratory data analysis (EDA) provided insights into the dataset's structure, while feature engineering added new features to improve performance. The dataset was split into training and testing subsets, ensuring test set validity by avoiding distribution rebalancing during preprocessing. Outliers were addressed, and clustering identified common customer profiles, ensuring data quality and relevance for modeling.

2.1 Data Loading and Dataset Integration

To enhance our analysis, we sourced additional datasets from a newer version on the IBM website, enriching our study with granular customer information, including demographics, location, service subscriptions, and customer status. We verified the uniqueness and consistency of customer IDs across datasets, excluding the population dataset due to missing IDs to ensure accurate merging. The final combined dataset simplifies structure, reduces redundancy, and enhances readability, making it efficient for visualization and reporting.

2.2 Handling missing value

Key variables in the dataset, including Offer, Internet Type, Churn Category, and Churn Reason, contained missing values critical for analyzing customer behavior and churn patterns. Systematic patterns were identified: missing entries in Offer reflected customers without promotional offers, while missing values in Internet Type indicated the absence

of active internet services. For Churn Category and Churn Reason, missing data occurred exclusively for non-churned customers, aligning logically with their status. To address this, missing values were imputed as follows: "No Offer" for Offer, "No Internet Type" for Internet Type, and "No Churn" for the churn-related columns. Post-imputation validation ensured the dataset was complete and ready for analysis, preserving consistency and analytical integrity.

2.3 Data Type Conversion and Categorical Variable Encoding

Most of our machine learning algorithms require numerical input. Using Pandas, we first identified columns as `int64`, `float64`, or `object`. Object columns with fewer than 50 unique values were converted to the `category` type to optimize memory usage and explicitly define categorical features. We also removed columns with entirely unique entries (e.g., IDs) to retain only meaningful features for modeling.

Categorical variables were encoded using different methods: Label Encoding was applied to binary categorical variables. Using `LabelEncoder`, these variables were efficiently transformed into numerical values, preserving their relationships without increasing dimensionality. For features with multiple categories, we employed One-Hot Encoding, creating separate binary columns for each category and avoiding any unintended ordinal assumptions. This classification based on the data types ensured the most appropriate encoding method was applied to each feature, optimizing efficiency and interpretability.

To ensure compatibility with machine learning algorithms that require numerical input, we verified that all columns were numeric (`int64` or `float64`) after encoding. This process not only ensured efficient data structuring and enhanced interpretability but also laid a solid foundation for predictive modeling by transforming the dataset into a fully numerical format, ready for subsequent analysis.

2.4 Feature Engineering

To enhance the predictive performance and interpretability of our machine learning models, we employed feature engineering, generating new variables that captured complex relationships, transforming features to align with domain-specific insights.

A central aspect of our approach was creating **interaction features** to reveal latent relationships between variables. For instance, combining tenure and age allowed us to capture how customers' proportional engagement evolves over time. Similarly, features such as cumulative data usage relative to streaming subscriptions provided insights into customer behavior trends. We also implemented **aggregation techniques** to summarize binary features into composite indicators, such as the total number of services subscribed by a customer. This not only reduced dimensionality but also highlighted overall customer engagement levels. To improve interpretability, we designed **group-based features**, such as revenue tiers derived from customer charges using quantile-based thresholds. These clusters provided actionable insights into customer value segments, facilitating strategic decision-making. Lastly, domain-inspired transfor-

mations, such as scaling monthly charges to an annualized basis or creating ratios like refund-to-charges, ensured that the dataset was both interpretable and well-suited for model input. These transformations reflected key business metrics, allowing the models to align more closely with real-world dynamics.

2.5 Split the Data

To prevent data leakage, we removed columns containing "Churn" in their names, except for the target column `Churn`, ensuring the model is trained with valid features only. The dataset was then split into features (`X`) and the target variable (`y`), with `X` excluding unnecessary identifiers (e.g., `Unnamed: 0`) and `y` representing customer churn. Using stratified sampling with `StratifiedShuffleSplit`. We combined the train and test sets for preprocessing steps before the splitting such as scaling because these operations do not influence the test set data. This ensures that the preprocessing is consistent and correctly applied across both sets before splitting.

After the split, we verified the dataset shapes to ensure consistency: 7043 total records were divided into 5634 training and 1409 testing samples. For the removed outlier dataset there are 3885 training examples. The resulting datasets were saved as CSV files. For subsequent analysis as well as two specifically designed datasets for our dashboard which also contain the Customer ID as a unique identifier. Finally, we validated the training set's feature integrity, ensuring no crucial features were missing or incorrectly removed, resulting in a clean, reliable dataset optimized for machine learning tasks.

2.6 Outlier detection

Outlier detection was performed on the training set to preserve the test set's integrity. Three tilt features, Lifetime Value per Month, Refund to Charges Ratio, and Extra Data Usage Cost Proportion, were flagged for their extreme values and skewed distributions. Significant deviations were identified using interquartile range (IQR) analysis and visualizations such as boxplots and histograms. During the process, the data was appropriately scaled for consistency across methods.

2.7 Clustering Analysis

In the preprocessing phase, clustering was used to segment customers into meaningful groups. Principal Component Analysis (PCA) helped reduce dimensionality and visualize the data, revealing clear separations. Using K-means and DBSCAN, we identified three customer types: Loyal Customers have high revenue and low churn risk, reflecting satisfaction and stability. Moderate Users generate moderate revenue with steady engagement and minimal churn risk. At-Risk Customers have low revenue and high churn likelihood, indicating dissatisfaction.

3 Model selection

3.1 Gaussian Naive Bayes

Our project applies the Gaussian Naive Bayes (GNB) model to classify data effectively, with a focus on feature preparation, parameter tuning, and handling class imbalance. Continuous features were selected, standardized, and evaluated for Gaussian assumptions using visualization and statistical tests. Highly correlated features were identified through a correlation matrix for further consideration.

To enhance model performance, the `var_smoothing` parameter was optimized using GridSearchCV with 5-fold cross-validation, maximizing the F1-Score. Class imbalance was addressed through oversampling (SMOTE) and undersampling techniques, with SMOTE showing the best results in balancing the dataset and improving minority class predictions. The final model, trained on SMOTE-enhanced data and fine-tuned parameters, demonstrated significant improvements in Recall and F1-Score, highlighting the effectiveness of combining data augmentation and parameter optimization.

3.2 Multinomial Naive Bayes

Multinomial Naive Bayes (MNB) effectively classifies whether a customer will churn by handling discretized input data, such as segmented monthly charges. It performs exceptionally well in high-dimensional scenarios, with numerous feature dimensions and sparse samples. Its robustness allows it to handle data noise and feature redundancy effectively. However, MNB has limitations, primarily stemming from its core assumption of feature independence, which often does not hold in real-world scenarios. It has limited support for continuous features, typically requiring preprocessing (e.g., discretization), which may result in information loss. Moreover, as a linear classifier, MNB cannot model complex nonlinear decision boundaries and may struggle in scenarios with highly correlated features or intricate feature interactions.

3.3 Logistic Regression

For our customer churn analysis, we use Logistic Regression as a baseline model due to its simplicity and efficiency. With 73.5% churned and 26.5% non-churned customers, it provides probabilistic predictions that rank churn risk and uses feature coefficients to clearly identify key factors influencing churn. Despite assuming linearity, it's a practical starting point for benchmarking performance.

3.4 KNN

In our customer churn analysis, k-Nearest Neighbors (KNN) is selected as an additional baseline model due to its intuitive and effective approach. With $k = 15$ yielding the best results, KNN predicts class labels based on the majority vote of the closest neighbors. Its simplicity and performance make KNN a reliable tool for assessing model effectiveness in churn analysis.

3.5 Decision Tree

Decision Tree is a versatile and interpretable classification algorithm that splits data into subsets based on feature values, forming a tree-like structure. It is particularly effective for datasets where interpretability and ease of implementation are priorities. It reads training and testing datasets, sets parameters for the decision tree (e.g., maximum depth and minimum sample count), trains the model, and performs cross-validation to evaluate performance (including accuracy, precision, recall, etc.). Then, it makes predictions on the test set, outputs the evaluation metrics, and saves the model and results for future use.

3.6 Random Forest

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their outputs for robust classification. It is particularly effective for datasets with high-dimensionality, noise, or imbalanced classes. We train and evaluate a random forest model. First, it loads the training and testing datasets (including data with outliers removed for the training) and reshapes the target variable into a 1D array. Then, it initializes the RandomForestClassifier model with parameters and trains it using the training dataset. Next, it performs 10-fold cross-validation to compute performance metrics, saving the evaluation metrics.

3.7 Support Vector Machines

Support Vector Machines (SVM) were applied to predict customer churn by identifying key features and leveraging the kernel trick to handle non-linear separability. After cleaning the dataset by removing low-variance and irrelevant features, the remaining features were standardized to ensure equal contributions to the model, given SVM's sensitivity to feature scales. The model was trained using the RBF kernel, which is well-suited for non-linear classification tasks, and its performance was evaluated through cross-validation and on the test set, using metrics such as accuracy, precision, recall, F1-Score, and ROC-AUC.

To optimize model performance, hyperparameters were fine-tuned using GridSearchCV and RandomizedSearchCV. Additionally, for linear kernels, feature importance was analyzed using the coefficients of the decision boundary. The final model achieved a high recall rate, minimizing false negatives in churn predictions, while also balancing precision and F1-Score.

3.8 Multilayer Perceptron Neural Networks (MLP)

The MLP model was selected for this study based on the following key considerations:
Non-Linear Modeling Capability. The dataset used in this research involves non-linear relationships between features, which cannot be effectively captured by linear models. The MLP's non-linear activation functions allow it to identify and model these complex interactions.

Flexibility and Adaptability. The MLP architecture can be customized in terms of the number of hidden layers and neurons to match the complexity of the data. This flexibility ensures that the model has sufficient capacity to learn intricate patterns.
Proven Track Record in Classification Tasks.MLPs have consistently demonstrated high performance in classification problems, aligning well with the study's objective of predicting.

3.9 XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful ensemble learning method based on gradient boosting that is widely used for classification and regression tasks. It combines the predictive power of decision trees with boosting techniques to improve model accuracy and robustness.

XGBoost was applied to this dataset for its ability to handle complex feature interactions and high-dimensional data effectively. It iteratively builds decision trees, focusing on reducing residual errors to capture intricate patterns while maintaining flexibility. Cross-validation was used to evaluate and fine-tune the model, ensuring generalization to unseen data. XGBoost's strengths include handling mixed feature types, robustness to overfitting through L1 and L2 regularization, and efficient parallel processing, making it a strong baseline for classification tasks despite requiring careful hyperparameter tuning.

3.10 Nearest Centroid

The Nearest Centroid classifier is a simple and efficient algorithm that assigns new data points to the class with the nearest centroid (mean) from the training data. It works well for datasets with well-separated classes and linear decision boundaries. The Nearest Centroid algorithm was used to predict customer churn, leveraging boolean features without requiring scaling. Preprocessing involved removing highly correlated features to retain only the most informative ones. The classifier is computationally efficient, calculating class centroids during training and distances during prediction. Unlike k-NN, it is robust to noise by aggregating information from all instances in a class.

4 Evaluation Approach

Our evaluation ensured reliable and valid results through systematic testing of predictive models, using metrics like accuracy, precision, recall, F1-score, and ROC-AUC. Performance was validated through cross-validation (CV) on training data and test set predictions, including an additional set with outliers removed on the training set.

To maintain class distributions, the dataset was stratified into training and test sets. Model selection prioritized minimizing false negatives, given the business impact of missed churners, while balancing false positives. The F1-score was chosen as the primary metric for its balance between precision and recall.

Baselines, including a majority class predictor, random predictor, and a domain-informed rule-based model, provided reference points. Ultimately, the Decision Tree emerged as the best-performing model due to its high F1-score and interpretability, making it well-suited for real-world deployment.

4.1 Baseline

To establish a benchmark for our models, we developed and evaluated three baseline approaches.

The simplest baseline was the **majority class predictor**, which always predicted the most frequent class based on the training dataset. Next, we implemented a **rule-based classifier** leveraging domain knowledge. This model applied a hierarchy of rules, prioritizing conditions strongly associated with churn. Finally, the **random predictor** assigned churn probabilities uniformly, maintaining the class distribution of the training dataset.

4.2 Model Evaluation and Results

The detailed evaluation metrics for each model and all training and testing results can be found in the Github Repository Telco Customer Churn. Since the models performed best in the CV on the training set with outliers being included, we decided to make our final prediction on the test set, which also included outliers. An overview of our evaluation metrics for all models on the test set can be found in 1.

Model	**Accuracy**	**Precision**	**Recall**	**F1**	**ROC AUC**
Decision Tree	0.963804	0.942466	0.919786	0.930988	0.949748
XGBoost	0.963804	0.965418	0.895722	0.929265	0.942064
SVM	0.963804	0.968116	0.893048	0.929068	0.941210
MLP	0.961675	0.930108	0.925134	0.927614	0.992586
Random Forest	0.948900	0.968944	0.834225	0.896552	0.912281
MNB	0.803407	0.835809	0.803407	0.811735	0.807271
Nearest Centroid	0.800568	0.836163	0.800568	0.809397	0.807900
GNB	0.851668	0.676660	0.844920	0.751486	0.849513
KNN	0.876508	0.818471	0.687166	0.747093	0.816047
Baseline	0.734564	0.539584	0.734564	0.622155	0.500000
Baseline Rule Based	0.598297	0.792817	0.598297	0.614801	0.698394
Baseline Random	0.618169	0.611537	0.618169	0.614751	0.501885
Logistic Regression	0.797019	0.689655	0.427807	0.528053	0.679121

Table 1: Performance Metrics for Various Models

The Decision Tree model emerged as the top-performing algorithm, achieving an F1-score of 93.10%. Its hierarchical structure allowed it to effectively handle the dataset's interactions and identify key patterns. This model's robustness to noisy data and its ability to maintain interpretability make it particularly suitable for real-world applications

where decision transparency is critical. Its balanced F1-score highlights its reliability across metrics, validating its selection as the most suitable model for deployment.

XGBoost demonstrated exceptional performance, achieving an F1-score of 92.93%. Its ability to model complex interactions between features, combined with effective handling of class imbalance through boosting, allowed it to maintain high precision (96.54%). The discrepancy in its recall (89.57%) compared to Decision Tree (91.97%) reflects XGBoost's tendency to optimize for overall accuracy, sometimes at the expense of recall.

The SVM model achieved an F1-score of 92.91%, closely matching that of XGBoost. Its kernel-based approach enabled it to capture nonlinear patterns effectively, resulting in the highest precision among all models (96.81%). However, its recall (89.30%) was lower than that of the Decision Tree and MLP. The addition of outliers significantly impacted SVM's recall, dropping it to 68.18% in the outlier-removed dataset. This suggests that the SVM relied heavily on edge cases for defining class separations and struggled with generalizing to subtler churn patterns without such examples. Its computational complexity, especially during cross-validation, further complicates its practical application despite its theoretical strengths.

The MLP model achieved a balanced F1-score of 92.76%, with high recall (92.51%) and precision (93.01%). Its neural network architecture allowed it to capture intricate patterns and nonlinear relationships in the data, contributing to its competitive performance.

Random Forest achieved an F1-score of 89.66%, with a high precision of 96.89% but relatively lower recall (83.42%). Its ensemble structure, based on averaging decisions across multiple trees, provided robustness against overfitting but limited its ability to capture nuanced patterns. Despite these shortcomings, Random Forest's strong precision (100% in the outlier-removed dataset) underscores its strength in minimizing false positives, making it suitable for use cases where high confidence in positive predictions is paramount.

Both MNB and Nearest Centroid models delivered comparable F1-scores of 81.17% and 80.94%, respectively. While these models maintained high recall (80.34% and 80.06%), their performance was hindered by lower precision. The simplicity of their underlying algorithms limits their ability to capture complex patterns, explaining their underperformance compared to more sophisticated models like XGBoost and Decision Trees.

The GNB model displayed an interesting trade-off, achieving a high recall (84.49%) but a low precision (67.66%), resulting in an F1-score of 75.15%. Its strong recall highlights its potential for identifying churners broadly, but its inability to model feature dependencies led to frequent misclassifications.

The KNN model underperformed with an F1-score of 74.71%. Its reliance on proximity-based decisions made it less effective in high-dimensional and imbalanced datasets. The relatively low recall (68.72%) indicates that KNN struggled to generalize effectively, making it unsuitable for churn prediction in its current form.

Baseline models provided critical reference points for evaluating the advanced models. The rule-based predictor achieved an F1-score of 61.48%, demonstrating the utility of domain knowledge in crafting simple but effective models. However, other baselines,

including random and majority class predictors, performed poorly, highlighting the limitations of naive approaches in addressing the complexities of churn prediction.

Decision Tree emerged as the most versatile model, balancing recall and precision effectively across all conditions. In contrast, models like SVM offered interpretability and models like XGBoost offered robustness, but fell short in classifying unseen data.

4.3 Feature Importance Analysis

To assess the significance of each feature in the model's decision-making process, the feature importance was extracted directly from the trained Decision Tree model. This was done using the feature_importances_ attribute, which provides a measure of how each feature contributes to the model's prediction.

The **Satisfaction Score** was by far the most critical predictor, with a significant importance value (0.839), underscoring its centrality in churn prediction. Following this, **Contract Type (Month-to-Month)** and **Online Security** were identified as moderately influential, reflecting the impact of contract flexibility and service add-ons on customer retention.

5 Discussion

This project highlights the importance of aligning machine learning models with business needs, particularly for customer churn prediction. The Decision Tree model was selected for its balance of strong F1-score and interpretability, effectively minimizing false negatives shown through its high recall to identify at-risk customers.

Feature importance analysis confirmed **Satisfaction Score** as the key predictor, while misclassification analysis revealed gaps in the feature set, suggesting the potential value of adding temporal or qualitative data. The Decision Tree also demonstrated resilience to outliers, unlike models such as SVM, which were more sensitive to data variations. These findings emphasize the need for tailored preprocessing strategies for different models.

Finally, the deployment of a churn prediction dashboard showcased the practical application of these insights. Future work could enhance the system by integrating time-series features and real-time data for dynamic customer behavior predictions.

6 Conclusion

This project demonstrated the effective application of machine learning to the challenge of customer churn prediction, with the Decision Tree model standing out as the most effective approach. Its high recall, interpretability, and robust performance were critical factors in its selection. The development of a user-friendly churn prediction dashboard further emphasized the practical value of these insights, enabling targeted retention strategies that improve customer satisfaction and reduce churn rates.

By addressing current limitations, such as feature enrichment with temporal and qualitative data, and integrating real-time predictive capabilities, the system can evolve into

a more dynamic and responsive tool. These advancements will ensure sustained business impact and adaptability to changing customer behaviors, laying the groundwork for future improvements.

References

Macko, Steven (2019). *Telco Customer Churn Dataset*. Available at: https://community.ibm.com/community/user/businessanalytics/blogs/steven-macko/2019/07/11/telco-customer-churn-1113 [Accessed: 2024-12-07].

Declaration of Honor

I hereby declare that I have written the enclosed project report without the help of third parties and without the use of other sources and aids other than those listed in the table below and that I have identified the passages taken from the sources used verbatim or in terms of content as such. or content taken from the sources used. This work has not been submitted in the same or a similar form been submitted to any examination authority. I am aware that a false declaration declaration will have legal consequences.

Declaration of Used AI Tools

Tool	Purpose	Where?	Useful?
ChatGPT	Rephrasing	Throughout	+
DeepL	Translation	Throughout	+
Github Copilot	Code generation	Github Repository	+

Signatures
Mannheim, 7. December 2024