Predicting Telecom Customer Churn Using Machine Learning

Yinuo Zhang
Brown University, Department of Data Science
https://github.com/Andyzhang67432/data1030-project

---

1. Introduction

Customer churn is a major problem for telecom companies because acquiring a new customer is much more expensive than keeping an existing one. Accurate churn prediction allows firms to target at-risk customers with retention offers and improve customer lifetime value.

This project uses the IBM Telco Customer Churn dataset with 7,043 customers and about 20 features describing demographics, account information (such as tenure, contract type and billing method), and subscribed services (phone, internet and add-ons). The target variable indicates whether the customer left the company in the last month.

Previous work on this dataset, mainly from Kaggle notebooks and academic case studies, reports test ROC–AUC scores roughly in the 0.83–0.90 range, with tree-based ensembles and support vector machines often performing best. In my experiments, the best model (logistic regression) achieves a test ROC–AUC of 0.84, which is consistent with these published results and suggests that my preprocessing and evaluation are reasonable.

The main research question is: can we predict customer churn with enough accuracy to support cost-effective retention strategies, and which customer characteristics are most predictive of churn?

---

2. Exploratory Data Analysis

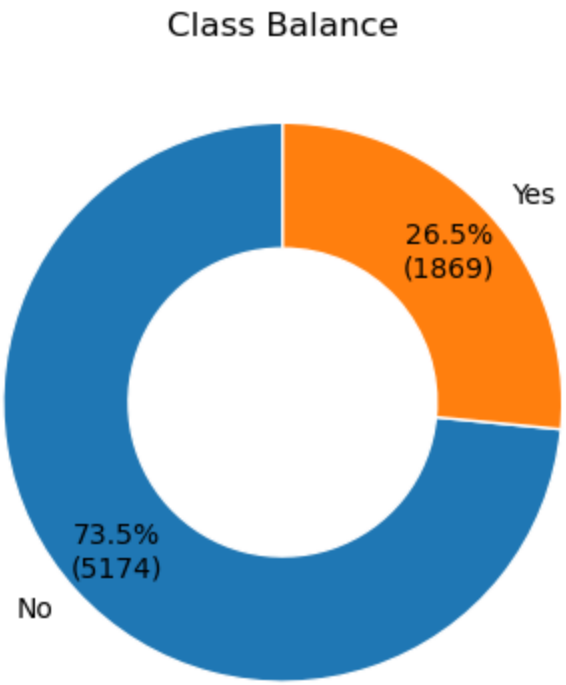

Figure 1. *Class balance of churn vs. non-churn customers in the Telco dataset.*

Class balance in the dataset is shown in Figure 1. Churn is moderately imbalanced: 26.5% of customers churn and 73.5% stay, so a “always predict no churn” model would already achieve about 73% accuracy, which motivates using ROC–AUC and F-scores rather than accuracy alone.

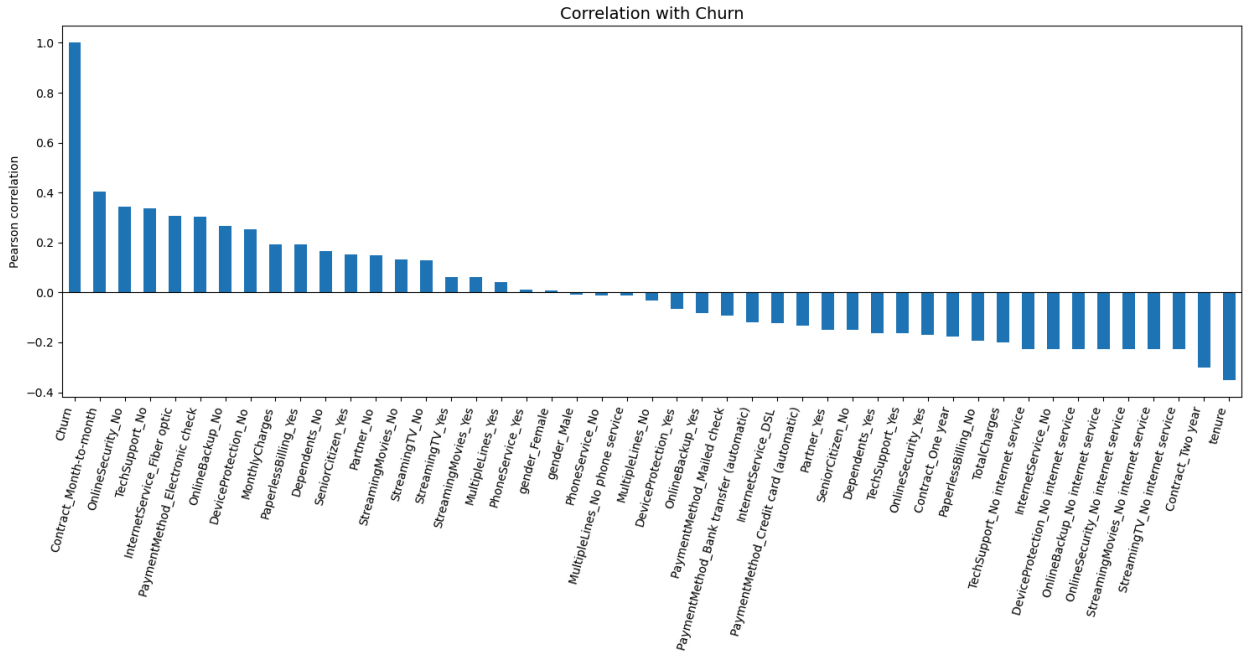

Figure 2. *Pearson correlation between churn and each preprocessed feature.*

To understand which features are most associated with churn, Figure 2 plots the Pearson correlation between the binary churn indicator and each preprocessed feature. Several contract- and service-related variables stand out: month-to-month contracts, lack of online security/backup, paperless billing and paying by electronic check are positively correlated with churn, while one- and two-year contracts and longer tenure are negatively correlated.

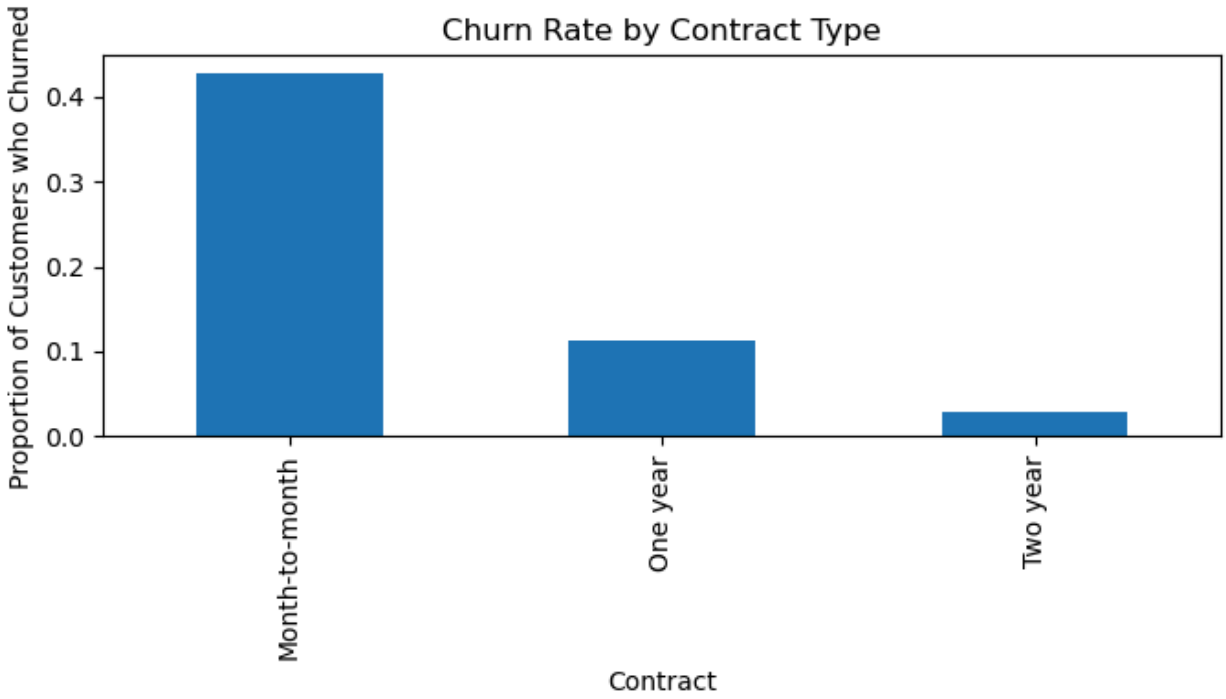

Figure 3. *Churn rate by contract type (month-to-month vs. one- and two-year contracts).*

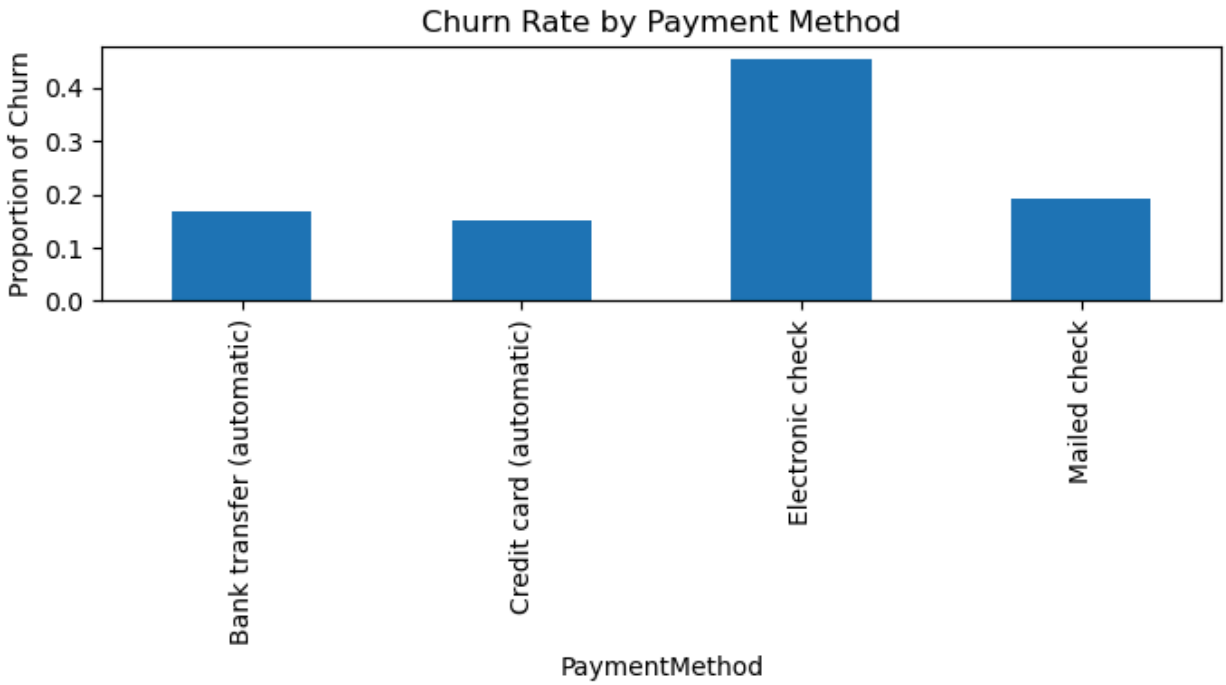

Figure 4. *Churn rate by payment method (automatic transfers, credit card, electronic check, mailed check).*

Figures 3 and 4 zoom in on two of these drivers. Customers on month-to-month contracts churn at more than four times the rate of two-year contract customers (Figure 3), and customers who pay by electronic check churn substantially more often than those on automatic bank transfer or credit-card payments (Figure 4). Together, these patterns suggest that contractual commitment and payment convenience are key behavioral signals of churn risk.

---

3. Methods

3.1 Data Splitting Strategy

The data were split into training (60%), validation (20%), and test (20%) sets using stratified sampling on the churn label so that each split preserves the overall churn rate. The training set is used to fit the preprocessing pipeline and models, the validation set is used for hyperparameter selection, and the test set is held out for the final performance evaluation.

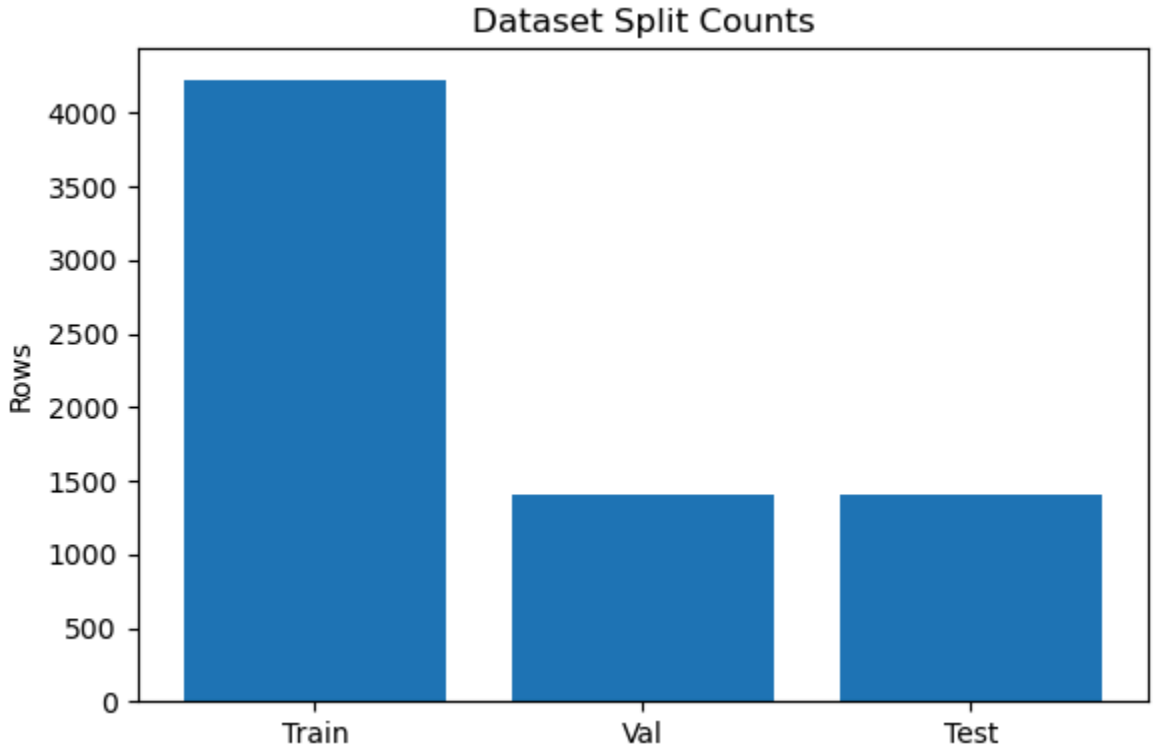

Figure 5. *Dataset split into 60% training, 20% validation, and 20% test sets; the bar heights show the number of rows in each split, confirming the intended stratified partition of the data.*

3.2 Data Preprocessing Pipeline

Preprocessing was implemented with a scikit-learn ColumnTransformer that applies separate pipelines to numerical and categorical features.

For numerical variables, missing values are imputed with the mean and then standardized with StandardScaler.

For categorical variables, missing values are imputed with the most frequent category and then one-hot encoded using OneHotEncoder with handle_unknown="ignore" so that unseen categories at test time do not cause errors.

The preprocessing transformer is fit only on the training data and then applied to the validation and test sets to avoid data leakage.

3.3 Model Selection and Hyperparameter Tuning

I evaluated four supervised learning algorithms: logistic regression, a single decision tree, random forest, and gradient boosting. Logistic regression with L2 regularization serves as a linear baseline and provides interpretable coefficients. The decision tree captures simple non-linear interactions and offers an intuitive rule-based model. Random forest and gradient boosting are ensemble methods that combine many decision trees; the former reduces variance through bagging, while the latter improves performance by sequentially correcting errors of previous trees.Most previous work on this dataset uses XGBoost as the main ensemble model. In this project I instead use scikit-learn's Gradient Boosting as a closely related but simpler boosting method, because I wanted to compare a less specialized ensemble model against the XGBoost results reported in prior studies.

The main hyperparameters for each model and the ranges explored are summarized in Table 1. For each combination of hyperparameters, the classifier was trained on the training set and evaluated on the validation set using ROC–AUC. The configuration with the highest validation ROC–AUC was selected as the final best version of that model.

Table 1. ML Models and Tuned Hyperparameters

ML Model	Hyperparameter	Values
Logistic Regression	C	0.01, 0.1, 1.0, 10.0
Logistic Regression	solver (fixed)	lbfgs
Decision Tree	max_depth	5, 10, 15, None
Decision Tree	min_samples_split	10, 20
Random Forest	n_estimators	50, 100, 200
Random Forest	max_depth	10, 15, 20
Random Forest	min_samples_split	5, 10, 20
Gradient Boosting	n_estimators	50, 100
Gradient Boosting	learning_rate	0.1, 0.05
Gradient Boosting	max_depth	3, 5

3.4 Evaluation Metrics

The primary performance metric is the area under the ROC curve, because it is threshold-independent and less sensitive to class imbalance than accuracy. After selecting hyperparameters based on validation ROC–AUC, I report ROC–AUC on the held-out test set. To provide a more complete picture of performance at the default 0.5 decision threshold, I also report test accuracy, precision, recall and F1-score for each model.

3.5 Uncertainty Quantification

Two sources of uncertainty were quantified. First, to measure variability due to different data splits, I computed 5-fold cross-validation scores (ROC–AUC) on the training set for each best model and summarized them as mean +- standard deviation. Second, to capture stochasticity in non-deterministic algorithms, I refit the Random Forest and Gradient Boosting models five times with different random seeds and evaluated each run on the test set; the resulting test ROC–AUC values are again summarized as mean +- standard deviation.

---

4. Results

4.1 Overall model performance

Table 2 summarizes the test performance of the four classifiers in terms of ROC–AUC, accuracy, precision, recall, and F1-score. All models outperform the naïve “always predict no churn” baseline accuracy of about 73%, confirming that they learn signal beyond class

imbalance. Among them, Logistic Regression achieves the highest test ROC–AUC of **0.84**, with the tree-based ensembles performing slightly worse but still competitive, and the single Decision Tree clearly underperforming the ensembles.

Figure 6 visualizes these results. The left panel shows the test ROC–AUC of each model compared to the random-guess reference line at 0.5. All models lie well above this line, with Logistic Regression at the top and the Decision Tree at the bottom. The right panel shows metrics of accuracy, precision, recall, F1, AUC for the best model only. At the default 0.5 decision threshold, Logistic Regression attains high accuracy with reasonably balanced precision and recall, indicating that it does not achieve performance by simply predicting the majority class.

Because model performance can vary with different train/validation splits and, for ensembles, with different random seeds, I report uncertainties as described in Section 3.5. The 5-fold cross-validation ROC–AUC scores on the training set show small standard deviations for all models, suggesting that the ranking of models is stable across different splits. For Random Forest and Gradient Boosting, refitting the models with five different random seeds leads to similarly small variation in test ROC–AUC, so the reported scores are not driven by a single lucky initialization. Overall, Logistic Regression remains the most predictive and most stable model under both sources of uncertainty.

```
MODEL PERFORMANCE SUMMARY
              Model                                                Best Params        CV AUC  Test Accuracy  Test Precision  Test Recall  Test F1  Te
st AUC Test AUC(mean,std)
Logistic Regression                                {'C': 10.0, 'solver': 'lbfgs'} 0.8406±0.0134       0.798439        0.650000     0.521390 0.578635  0.
843724      0.8437±0.0000
      Decision Tree                    {'max_depth': 5, 'min_samples_split': 20} 0.8214±0.0097       0.791341        0.613636     0.577540 0.595041  0.
818321      0.8183±0.0000
      Random Forest {'n_estimators': 50, 'max_depth': 10, 'min_samples_split': 20} 0.8415±0.0157       0.798439        0.659574     0.497326 0.567073  0.
839583      0.8390±0.0011
  Gradient Boosting     {'n_estimators': 50, 'learning_rate': 0.1, 'max_depth': 3} 0.8433±0.0142       0.801987        0.664360     0.513369 0.579186  0.
842808      0.8428±0.0000
Best Model: Logistic Regression (Test AUC: 0.8437)
Best model is 0.14 standard deviations above baseline
```

Table 2. *Test ROC–AUC, accuracy, precision, recall, and F1-score for all models, with mean standard deviation where applicable.*

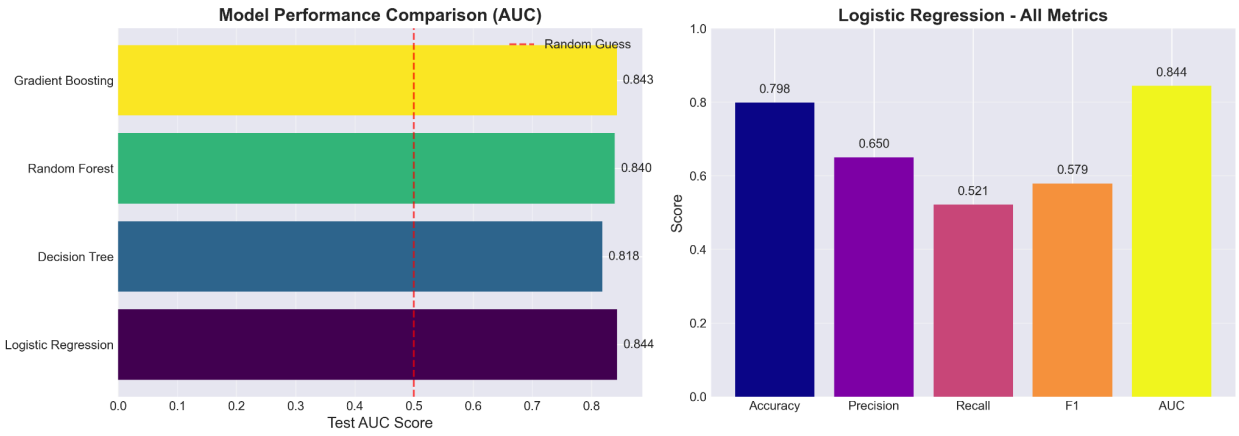

Figure 6. *Left: test ROC–AUC of all models compared to a random-guess baseline (0.5). Right: accuracy, precision, recall, F1-score, and ROC–AUC for the best model (Logistic Regression) on the test set.*

4.2 Global feature importance

To understand which customer characteristics drive churn predictions, I computed three complementary global feature importance measures for the best model: permutation importance, logistic-regression coefficient magnitude, and SHAP-based global importance.

Figure 7 shows the top features according to permutation importance on the test set. The most influential variables are contract-related and service-related: having a month-to-month contract, shorter tenure, higher monthly charges, lack of online security/backup/tech support, and paying by electronic check all cause the largest drops in ROC–AUC when shuffled. These patterns are consistent with the EDA: flexible contracts and low security add-ons are associated with substantially higher churn rates.

Figure 8 ranks features by the absolute value of their Logistic Regression coefficients. The same groups of variables appear near the top, with month-to-month contract, absence of Internet security add-ons, and electronic check payment pushing the model toward predicting churn, while long-term contracts, automatic payment methods, and longer tenure push toward retention. The agreement between permutation importance and coefficient magnitude suggests that the model is not overly reliant on a single artifact of preprocessing.

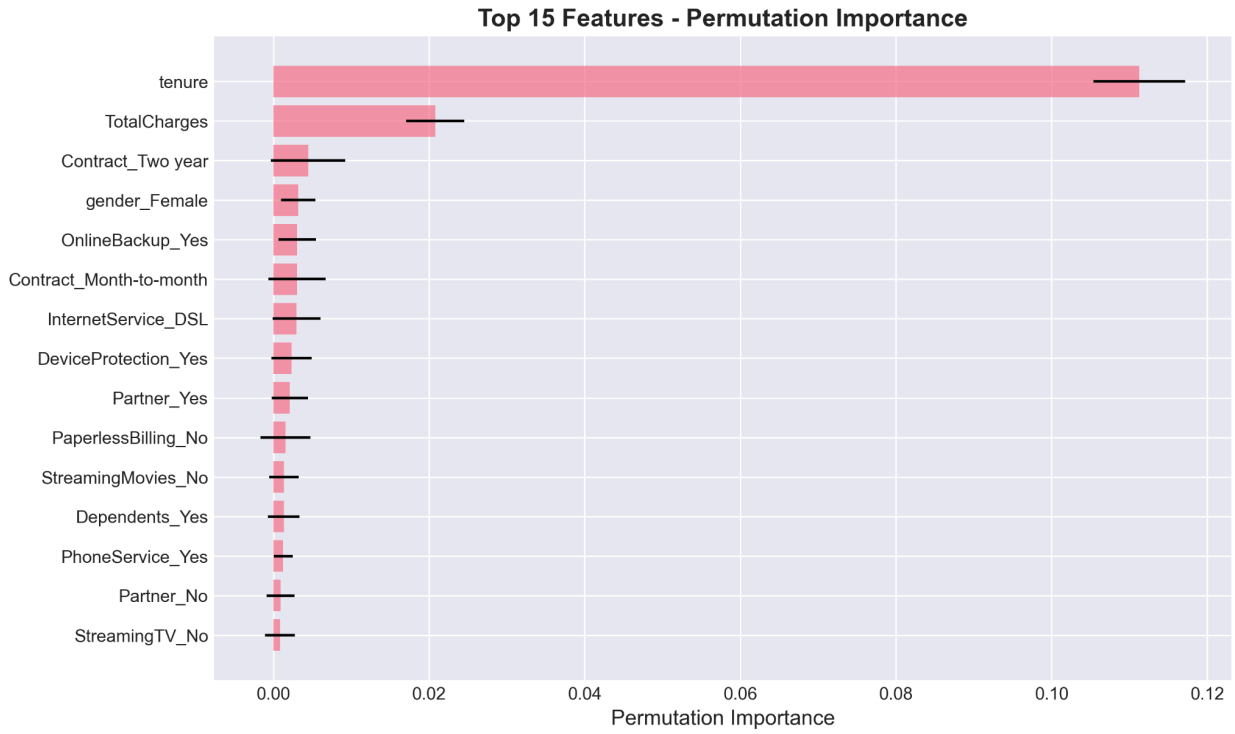

Figure 7. *Top 15 features by permutation importance for the best model; bars show mean decrease in ROC–AUC with error bars for standard deviation over repetitions.*

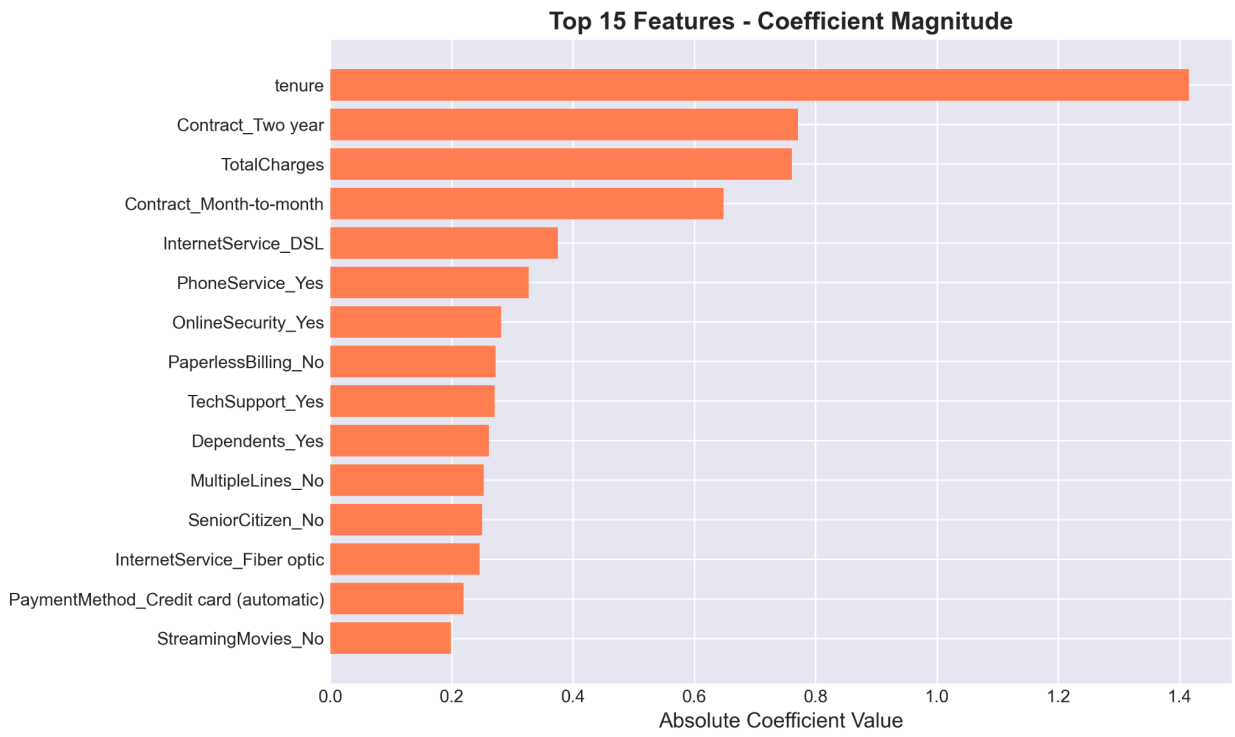

Figure 8. *Top 15 features ranked by absolute Logistic Regression coefficient; positive coefficients increase churn probability and negative coefficients decrease it.*

4.3 Local explanations with SHAP

While global importance scores summarize average behavior, business decisions often depend on understanding individual customers. To provide local explanations, I computed SHAP values for the best Logistic Regression model on the test set.

Figure 9 presents the SHAP summary plot, which shows both the magnitude and direction of each feature's contribution across all test customers. Red points on the positive side of the x-axis indicate conditions that push the model toward predicting churn; for example, high monthly charges and month-to-month contracts are strongly associated with higher churn risk. Blue points on the negative side indicate protective conditions such as two-year contracts, automatic bank/credit-card payments, and existing security services.

Figure 10 aggregates these SHAP values into a bar chart of mean absolute impact, providing a third global importance ranking. Again, contract type, tenure emerge as the dominant drivers, reinforcing the conclusions from permutation importance and coefficients.

Finally, Figure 11 shows a SHAP waterfall plot for one representative test customer. The base value corresponds to the average churn probability in the test data. Each bar then shows how individual features push this customer's prediction up or down relative to the average. For this example, a month-to-month contract, absence of certain security add-onst push the log-odds toward churn, while long tenure and lower monthly charges partially offset this effect. This kind of explanation can be used by a retention team to justify targeted interventions for specific customers.

Figure 9. *SHAP summary plot for the best model; each point shows a SHAP value for one customer–feature pair, colored by feature value.*

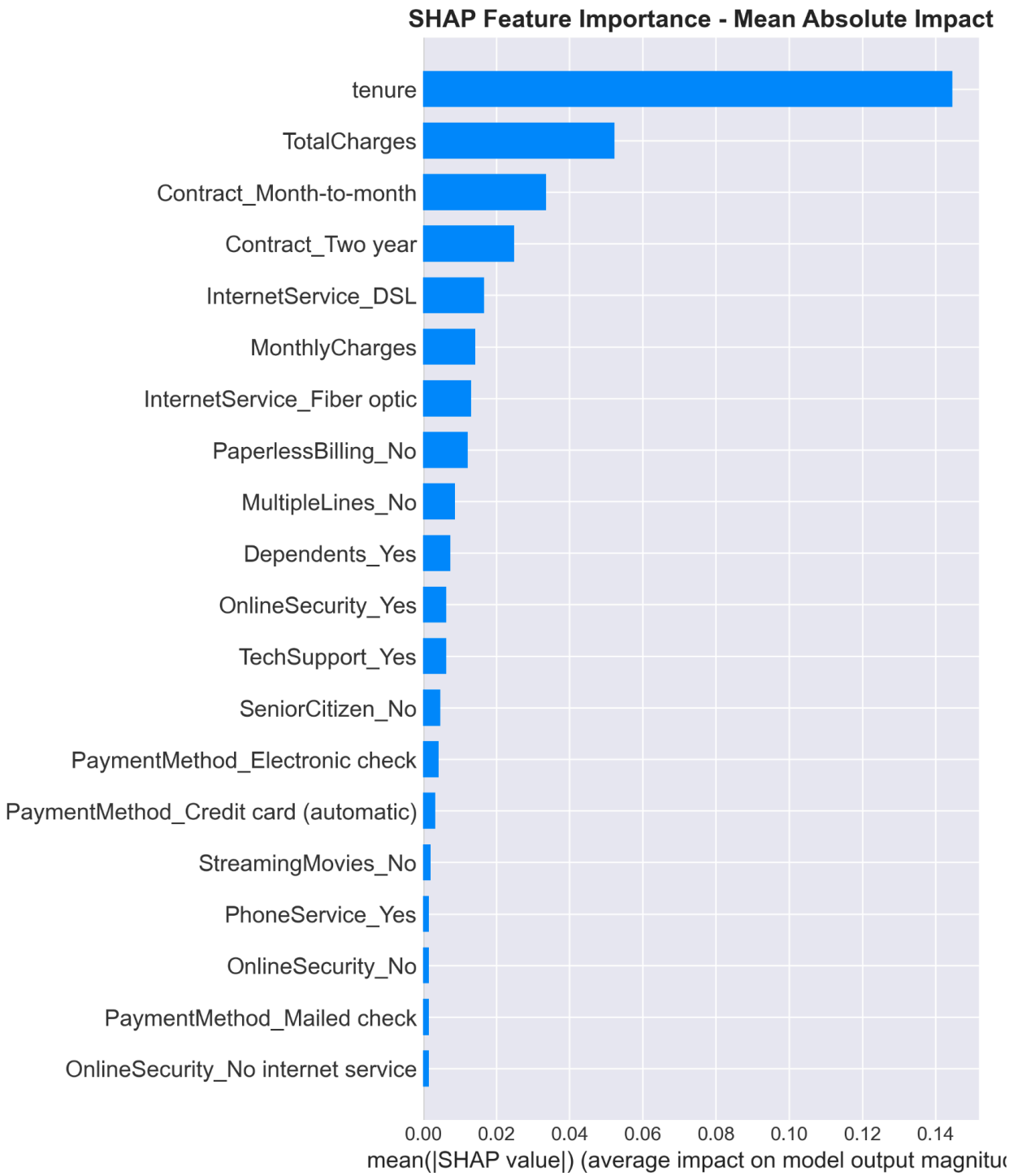

Figure 10. *Mean absolute SHAP value for each feature, summarizing its overall contribution to churn predictions.*

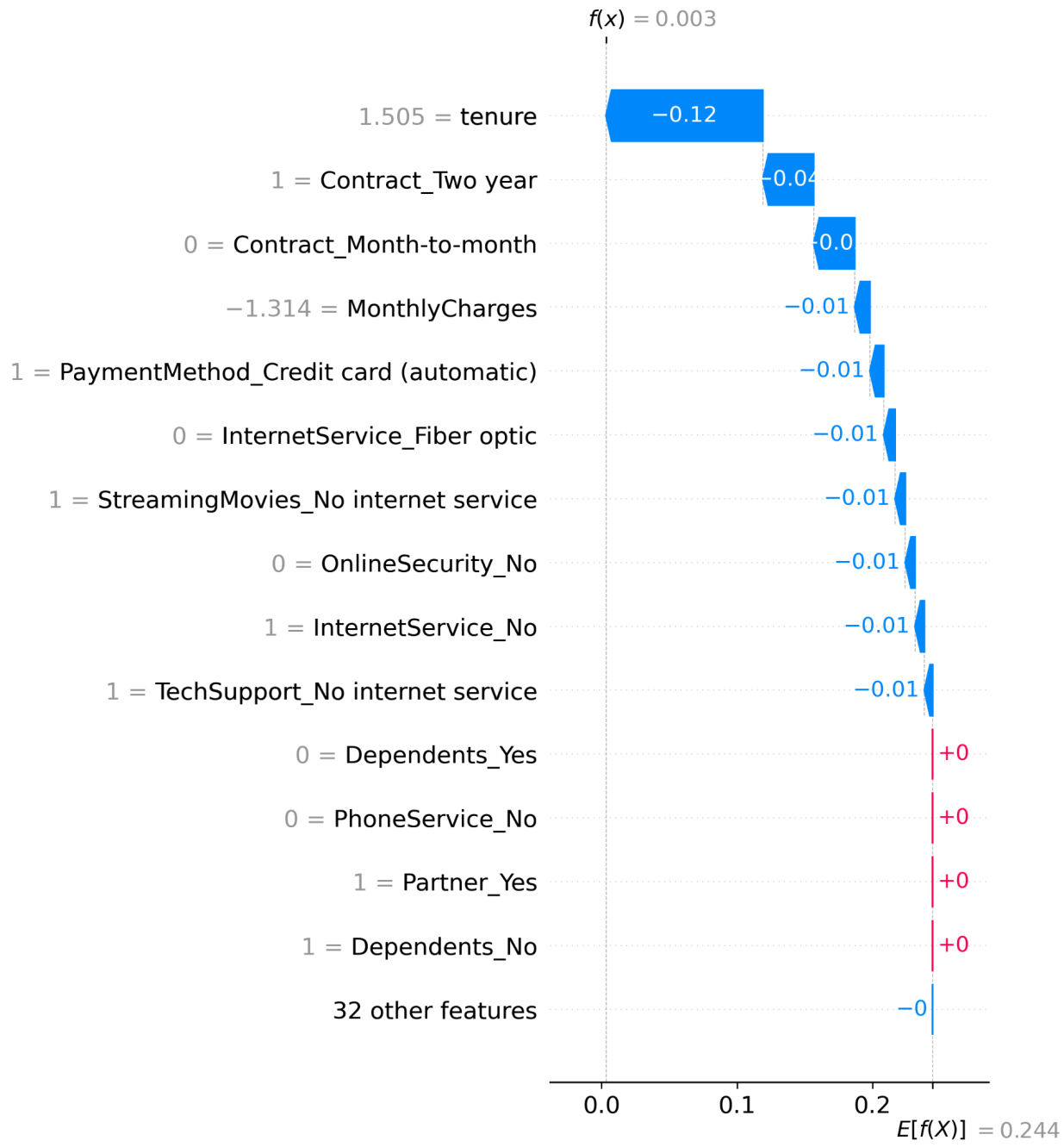

Figure 11. *SHAP waterfall plot for a single test customer, showing how each feature moves the prediction away from the average churn probability.*

---

5. Outlook

The current model works reasonably well, but there is room to improve it. I only used fairly simple features and a coarse hyperparameter search. In future work, I could add interaction features like tenure * monthly charges, try stronger models such as XGBoost or LightGBM, since they can handle missing values natively, and use better tuning methods, like random search with early stopping.

The dataset itself is a limitation: it is a single snapshot of each customer. Adding richer data—such as call-center and complaint history, app or internet usage, late payments, or information about local competitors—would likely make churn patterns clearer.

Finally, the model is not yet business-aware or dynamic. It treats all errors the same, does not model *when* churn will happen, and does not update over time. Future work could include choosing the decision threshold based on churn intervention costs, building time-to-churn models, segmenting customers and training separate models per segment, and regularly retraining or updating the model as new data arrives.

References

1. IBM Telco Customer Churn dataset. Kaggle. Available at: https://www.kaggle.com/datasets/blastchar/telco-customer-churn (accessed November 2025).

2. Bharti Prasad. “Customer Churn Prediction.” Kaggle Notebook. Available at: https://www.kaggle.com/code/bhartiprasad17/customer-churn-prediction (accessed November 2025).

3. Atindra Bandi. “Telecom Churn Prediction.” Kaggle Notebook. Available at: https://www.kaggle.com/code/bandiatindra/telecom-churn-prediction (accessed November 2025).
4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, É. (2011). “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research*, 12, 2825–2830.

5. Chen, T., & Guestrin, C. (2016). “XGBoost: A Scalable Tree Boosting System.” In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’16)*, 785–794.

6. Lundberg, S. M., & Lee, S.-I. (2017). “A Unified Approach to Interpreting Model Predictions.” In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*.