VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

Report

Churn Prediction with Apache Spark

Source Code GitHub Repository:https://github.com/Danh1905/Scalable-Project/blob/main/scalableProj.ipynb

Part 1: Understanding the Data

First Step: Understanding What I Have

Data Source:

https://www.kaggle.com/datasets/ranasarkar15/customerchurndatasets

It contains information on **7,043 customers**, including demographic attributes (e.g., gender, senior citizen), relationship status (partners, dependents), service subscriptions (phone service, internet service), and account-related features such as tenure and contract type.

The target variable is **customer churn**, indicating if a customer has discontinued the service.

However, raw tabular data alone is meaningless.

Before building any model, it is necessary to understand what patterns, imbalances, and assumptions are hidden.

Part 2: Logistic Regression & First Failure

2.1. Baseline Approach

I approached customer churn as a standard binary classification problem. The goal was straightforward: given customer attributes such as tenure, service usage, and contract information, predict whether a customer would churn.

To establish a baseline, I chose Logistic Regression, a simple and interpretable classification model.

2.2. Initial Results

At first glance, the model appeared to perform reasonably well.
The overall accuracy was relatively high (80.74%), suggesting that the model was able to classify a large proportion of customers correctly.

However, relying solely on accuracy can be misleading, especially in datasets where the target variable is imbalanced.
To gain a clearer understanding of the model's behavior, I examined the **confusion matrix**.

2.3. What the Confusion Matrix Reveals

The confusion matrix reveals a critical weakness of the baseline model. While the model correctly identifies most non-churn customers, it fails to capture a lot of actual churners, resulting in a miss of 173 churners. This indicates that the model favors the majority class, exploiting the imbalance in the dataset.

2.4. Why This Failure Is Not Surprising

This outcome is not entirely unexpected. Customer churn datasets typically exhibit class imbalance, where non-churn customers significantly outnumber churned customers. In such contexts, a model can achieve high accuracy by simply predicting the majority class. However, from a business perspective, this is problematic, as missing a churner (false negative) often incurs a higher cost than incorrectly flagging a loyal customer (false positive).

2.5. Lesson from the First Attempt

The failure of the baseline model highlights an important insight: the problem is not merely about accurately predicting churn but about identifying customers who are at risk of churning. This suggests that the issue lies more in how the problem is framed and evaluated rather than in the choice of model.

Part 3: Reframe the model

3.1 Why the First Model Failed

The initial Logistic Regression model achieved a reasonable overall accuracy.
However, a closer inspection of the confusion matrix revealed a critical issue: **the model failed to correctly identify churned customers**.

Despite performing well on the majority class (non-churn), the model predicted very few positive churn cases. This behavior is expected in customer churn prediction due to **class imbalance** and the asymmetric cost of misclassification.

In practice, failing to detect a churned customer (false negative) is far more costly than incorrectly flagging a loyal customer as at risk (false positive).

3.2 Logic over Model Metrics

Using accuracy as the primary evaluation metric proved misleading.
Because most customers do not churn, a model can achieve high accuracy by simply predicting "no churn" for the majority of cases.

Therefore, instead of optimizing accuracy, the focus was shifted to **recall for the churn class** (**label** = **1**), which measures the model's ability to correctly identify customers who actually churn.

By default, Logistic Regression uses a decision threshold of 0.5 to convert predicted probabilities into class labels.

At this threshold, the model showed:

- High accuracy

- Low recall for churn
- A confusion matrix dominated by false negatives

This confirmed that the model was overly conservative and biased toward predicting non-churn.

3.3 The fix

Rather than changing the model immediately, the decision boundary was adjusted to better align with the business objective.

The predicted churn probability was used directly, and the classification threshold was lowered from 0.5 to 0.3.
This change encourages the model to flag more customers as potential churners, increasing sensitivity at the cost of lower overall accuracy.

3.4 Result after adjustment

- Churn recall jumped from 53% to 76%
- More churned customers were correctly identified
- Overall accuracy decreased to 77.8%, but this trade-off is acceptable in a churn prediction context.

Part 4: Random Forest – When Model Complexity Fails

After improving churn detection by reframing the evaluation strategy in Part 3, a more complex, non-linear model was explored.
Random Forest was selected due to its ability to capture non-linear interactions between features such as tenure, contract type, and service usage.

The objective of this step was not to blindly outperform Logistic Regression, but to examine whether increased model complexity could further improve churn recall under the same evaluation framework.

4.1Model Performance Analysis

Contrary to expectations, **Random Forest** did not outperform **Logistic Regression**.
In fact, the confusion matrix revealed that Random Forest performed worse in identifying churned customers.

Specifically:

- The model strongly favored the majority class (non-churn)

- Churn recall remained low

- A large number of churned customers were still misclassified as non-churn

This behavior indicates that Random Forest collapsed toward predicting the dominant class, despite its non-linear modeling capability.

4.2 Why Random Forest Underperformed

This outcome highlights an important limitation of Random Forest in this context:

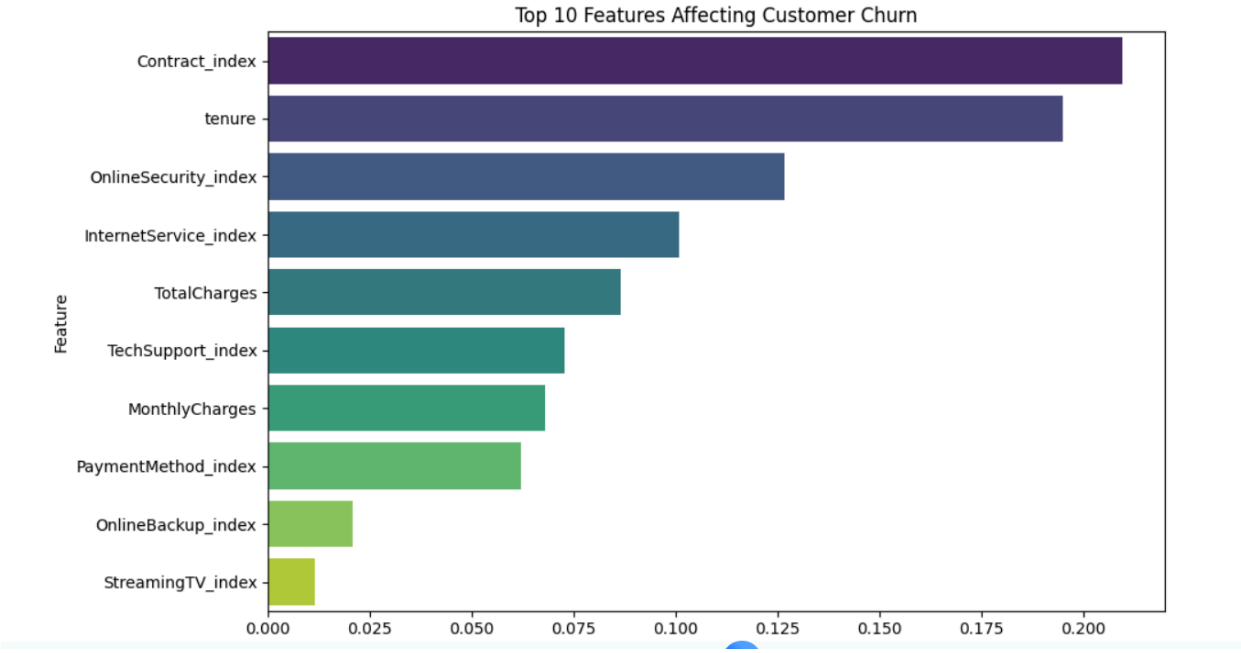
1. **Class Imbalance Sensitivity**
   Random Forest in Spark optimizes impurity-based metrics and does not explicitly account for class imbalance or asymmetric misclassification costs. As a result, it tends to favor conservative predictions that minimize overall error rather than maximizing churn detection.

2. **Weak Predictive Signal**
   Customer churn data often contains weak and noisy signals. In such cases, more complex models may overfit local patterns while failing to generalize meaningful decision boundaries for minority classes.

3. **Evaluation Objective Misalignment**
   While Random Forest excels at overall classification accuracy, it is

not inherently optimized for recall-driven objectives unless additional cost-sensitive mechanisms are introduced.

4.3 Feature Importance Analysis



4.4 Hyperparameter Tuning with Cross-Validation

To further investigate whether the underperformance of Random Forest was caused by suboptimal hyperparameters, a grid search with cross-validation was conducted.

The tuning process explored different combinations of tree depth and the number of trees, with weighted recall used as the evaluation metric to align with the churn detection objective.

Despite applying cross-validation and selecting the best-performing configuration, the tuned Random Forest model did not achieve a significant

improvement in churn recall compared to the baseline Random Forest model.

This result suggests that the limitation lies not primarily in hyperparameter selection, but in the model's inherent sensitivity to class imbalance and the weak predictive signal present in the dataset.

The failure of Random Forest demonstrates that **model complexity alone does not guarantee better performance**.
In this project, the primary performance gains came from redefining the evaluation strategy rather than introducing more sophisticated models.

Logistic Regression, when paired with a business-aligned decision threshold, proved to be a more reliable and interpretable solution than a complex ensemble model.

Part 5: Conclusion

1. Problem Framing Matters More Than Model Choice

The most significant improvement in this project did not come from switching models, but from redefining how model performance was evaluated.
Focusing on churn recall instead of accuracy transformed an apparently weak baseline model into a useful predictive tool.

2. Accuracy Can Be Misleading in Imbalanced Problems

High accuracy masked poor churn detection performance in the early stages of the project.
This reinforces the importance of selecting evaluation metrics that reflect real-world business objectives, especially in imbalanced classification tasks.

3. Simpler Models Can Be More Effective

Despite its simplicity, Logistic Regression combined with threshold adjustment outperformed Random Forest in terms of churn recall.
This result highlights the practical value of interpretable, well-aligned models over more complex alternatives that lack cost-awareness.

4. More Complex Models Do Not Always Add Value

Random Forest did not improve performance and, in some cases, degraded it.
Rather than treating this as a failure, it provided valuable insight into the limitations of ensemble methods under class imbalance and weak signal conditions.

To conclude

The best-performing solution in this project was not the most complex model, but the one that best aligned with the business objective of identifying at-risk customers.

This project demonstrates that effective machine learning is not about chasing sophisticated algorithms, but about understanding the data, defining the right objective, and making informed trade-offs between performance and interpretability.

References:

Nguyen, Q. (2026). Iterative machine learning modeling in practice. Substack.

Rana Sarkar. (2020). Customer churn dataset. Kaggle.