Telecom Churn Prediction with LightGBM

Name: Vamsi Thota ID: 23096534

1. Introduction and Motivation

Market saturation and competitive pricing in the telecom are a very critical problem in the area of making customer churn. Firms can offer targeted promotions, enhance their customer service (e.g., handling a complaint of a problemaneously dissatisfied customer in a more positive and positive way) etc, if they can anticipate which customers are likely to exit. In this tutorial, we show you a step by step way of how to build and interpret a LightGBM (Light Gradient Boosting Machine) model, in order to predict telecom churn using data exploration, data preprocessing, model fitting, hyperparameter tuning, and evaluation with extended visualizations. (Lundberg, 2017)

Why is churn prediction critical? The bad word of mouth and lost revenue goes with every lost customer. However at the same time if a company can actually discover who the at risk customers are, it will be able to substantially enhance the certain retention strategies and also improve the company with the bottom line. At the end of this tutorial, you will have learned how to balance real world imbalances in your data and how you should tweak your model to perform up to its maximum capacity, and how you can tell your story of results to business stakeholders clearly. (Ke, 2017)

2. Dataset Overview and Preprocessing

2.1. Dataset Description

For this project, the dataset used is 7,043 records, which is the dataset for each unique telecom customer. 21 column includes the demographic information (such as gender, SeniorCitizen), account status (like Partner, Dependents, tenure), service usage (InternetService, OnlineSecurity, TechSupport), contract details (Contract, PaymentMethod) and financial metrics (MonthlyCharges, TotalCharges). Our target variable, whether the customer left (Yes) or stayed (No), is the final column, Churn.

1. Shape: (7043, 21)
2. Examples of Columns:
    - customerID: Unique identifier for each customer.
    - SeniorCitizen: Binary indicator (0 or 1).
    - tenure: Number of months the customer has stayed with the company.
    - MonthlyCharges: The amount charged to the customer monthly.
    - TotalCharges: The total amount charged to the customer.
    - Churn: Target variable (Yes/No).

2.2. Data Exploration

Initial data exploration reveals:

- All columns do not have null or missing values.
- Churn is not imbalanced, i.e., it has 1,869 'Yes' entries and 5,174 'No' entries, so about 26.5% of the dataset points to churn.
- TotalCharges may be stored as string, so it needs to be converted to numeric.

2.3. Data Preprocessing Steps

1. Dropping Non-Informative Columns
   We drop the customerID column since it is merely an identifier with no predictive value.
2. Converting TotalCharges
   Because TotalCharges may be stored as a string, we convert it to numeric with pd.to_numeric(..., errors='coerce'), ensuring any invalid entries become NaN, which we then fill or drop as needed.
3. Handling Missing Values
   Although the dataset has no missing values, we ensure that any potential NaNs in TotalCharges (due to conversion) are filled with the median of the column.
4. Target Encoding
   The Churn column is mapped from {Yes, No} to {1, 0} for modeling. This step ensures that we have a binary numeric target variable for classification.
5. Categorical and Numerical Features
   - Numerical: tenure, MonthlyCharges, TotalCharges
   - Categorical: gender, SeniorCitizen, Partner, Dependents, PhoneService, MultipleLines, InternetService, OnlineSecurity, OnlineBackup, DeviceProtection, TechSupport, StreamingTV, StreamingMovies, Contract, PaperlessBilling, PaymentMethod
6. One-Hot Encoding and Scaling
   We use a ColumnTransformer that applies:
   - StandardScaler to numerical features, giving them zero mean and unit variance.
   - OneHotEncoder with drop='first' to categorical features, reducing dummy variable traps. (Pedregosa, F., Varoquaux, G., Gramfort, A., et al, 2011)

2.4. Train-Test Split

We finally splitted the data using Stratified Sampling to keep the proportion of churners and non churners in both sets. We have 5,634 records in training set and 1,409 records in test set.

3. LightGBM Model Training

3.1. Why LightGBM?

LightGBM is a gradient boosting framework known for:

- Efficiency: It uses a histogram-based approach, making it faster and more memory-efficient than many other boosting libraries. (Ke, 2017)
- Handling of Categorical Data: Although we one-hot encode in this tutorial, LightGBM can natively handle some categorical features. (Ke, 2017)
- Imbalanced Data: LightGBM supports parameters (like is_unbalance or scale_pos_weight) to help with class imbalance.

3.2. Hyperparameter Tuning with GridSearchCV

We define a GridSearchCV to explore combinations of crucial LightGBM hyperparameters:

Hyperparameter	Range	Purpose
num_leaves	[31, 50]	Controls tree complexity
max_depth	[-1, 10, 20]	Sets maximum depth (-1 = unlimited)
learning_rate	[0.05, 0.1]	Step size for boosting
n_estimators	[100, 150]	Number of boosting iterations

By setting cv=5 and scoring='accuracy', we ensure a robust estimate of each parameter combination's performance. The best parameters discovered are:

```
{'learning_rate': 0.05, 'max_depth': -1, 'n_estimators': 150, 'num_leaves': 31}
```

and the best cross-validation accuracy is ~0.798.

3.3. Final Model

Using these parameters, we instantiate the best LightGBM model and retrain on the entire training set (5,634 samples). This final model is then evaluated on the test set (1,409 samples).

4. Model Evaluation

4.1. Classification Report

On the test set, we obtain the following classification report:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.89      0.87      1035
           1       0.64      0.55      0.59       374

    accuracy                           0.80      1409
   macro avg       0.74      0.72      0.73      1409
weighted avg       0.79      0.80      0.79      1409
```

- Overall Accuracy: ~80%
- Churn Class (1): Precision ~64%, Recall ~55%, F1 ~59%. While these figures are acceptable, further improvement is needed to reduce missed churners (false negatives).

4.2. Confusion Matrix

The confusion matrix:

```
Confusion Matrix:
[[920 115]
 [169 205]]
```

- TN = 920: Non-churners correctly identified.
- FP = 115: Non-churners incorrectly flagged as churners.
- FN = 169: Churners missed by the model.
- TP = 205: Churners correctly caught.

Interpretation: The model is only missing ~169 churners though it performs well on non-churners. Since missed churners indicate that they are lost opportunity for retention strategies, this is very important.

4.3. ROC AUC

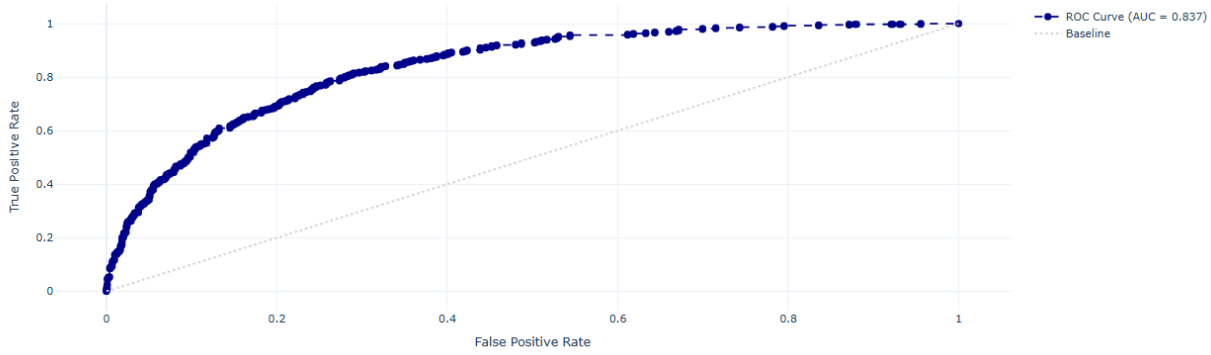
Based on the ROC AUC of the model's 0.837, the model can generally sort churners higher thannon churners. However, different thresholds or cost sensitive training may increase recall for churners to some extent in accordance with business objectives.

5. Visual Analysis and Interpretability

5.1. Interactive ROC Curve (AUC = 0.837)

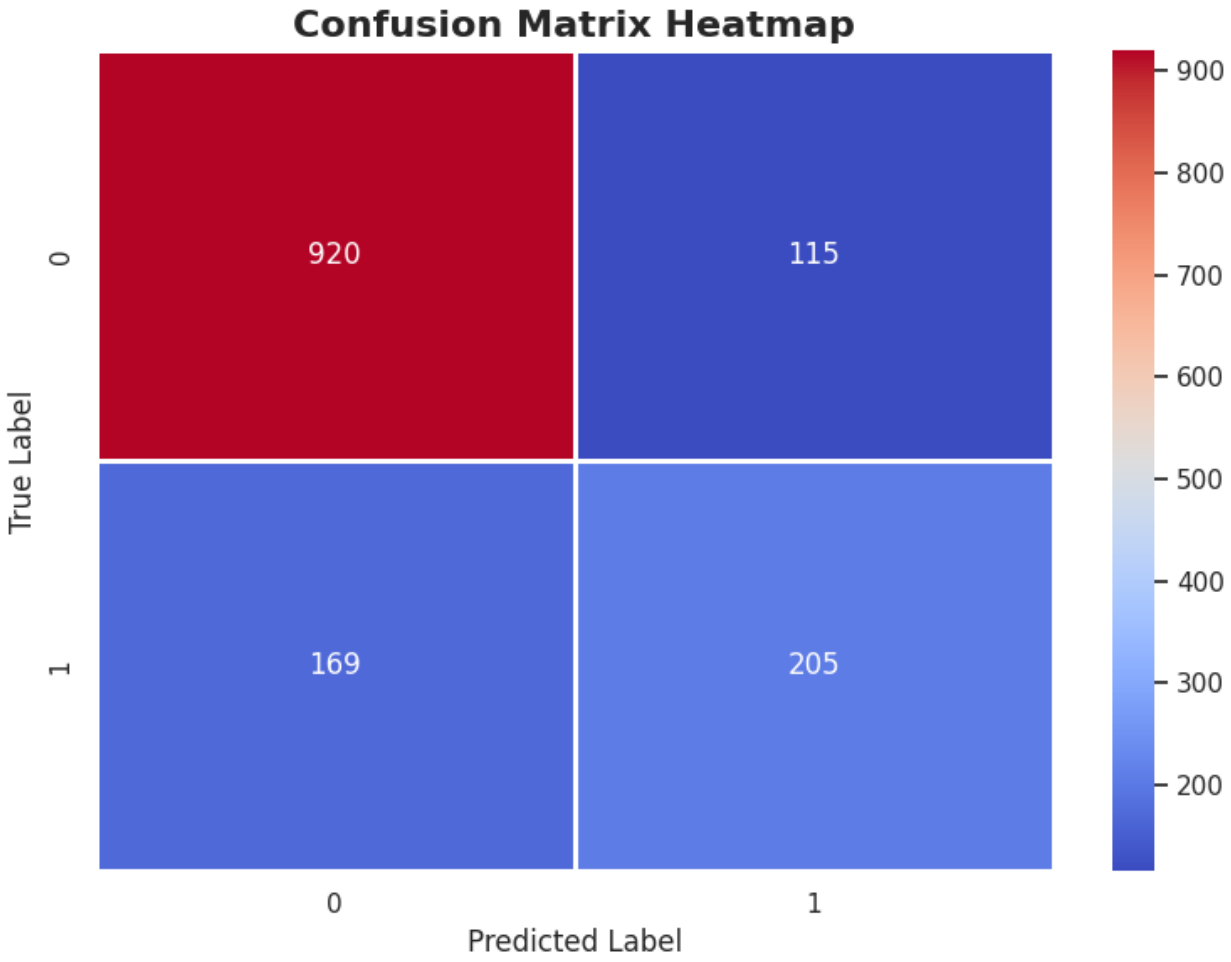
Using Plotly, we generate an interactive ROC curve that allows zooming and panning:

- AUC of 0.837 → The model is relatively good at distinguishing churners from non-churners.
- Lowering the threshold from the default 0.5 might improve recall, albeit at the cost of higher false positives.



5.2. Confusion Matrix Heatmap

A Seaborn heatmap of color shows which predictions for the majority class (no churn) are correct, and which are incorrect for predictions of churn. Churn problems are that kind of imbalance.

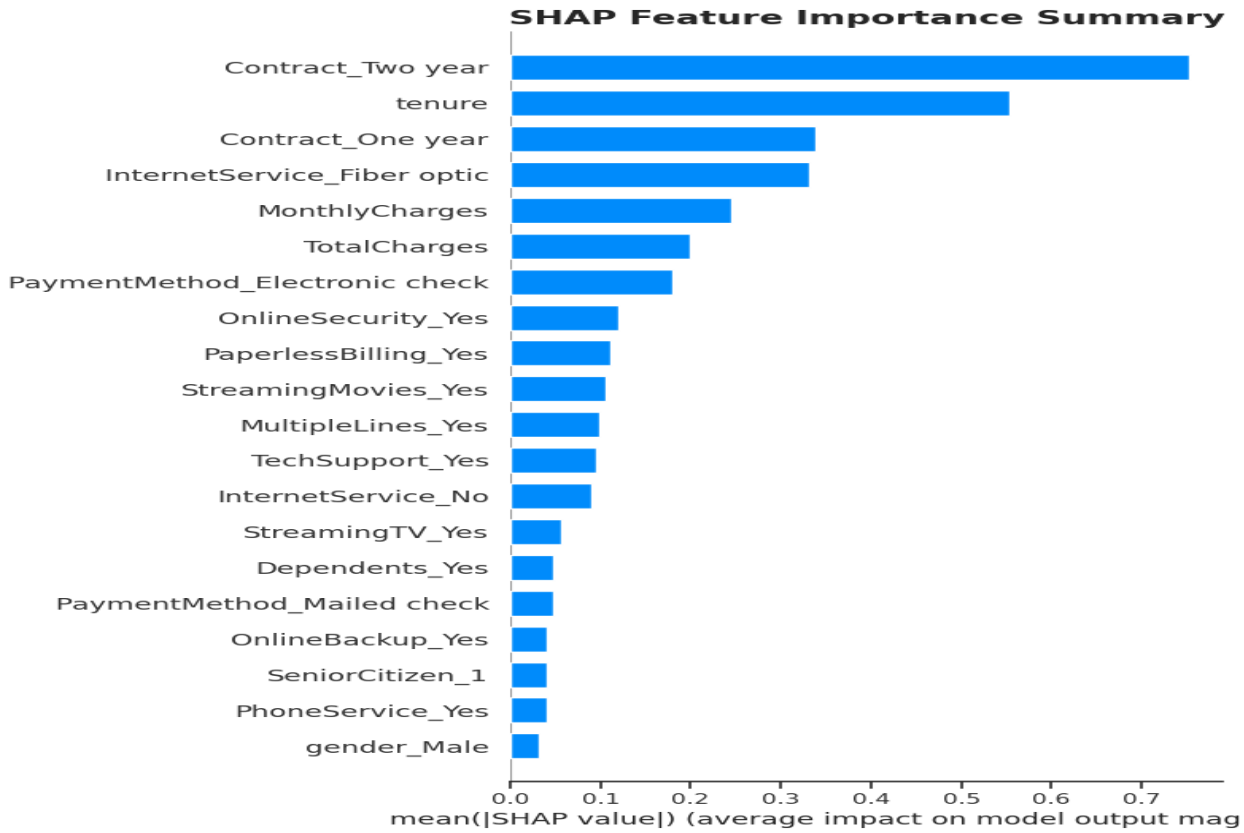


5.3. SHAP Summary Plot

SHAP (SHapley Additive exPlanations) values highlight how each feature influences the model's predictions:

1. Contract_Two year: Longer contract durations reduce churn likelihood.
2. tenure: Higher tenure means stronger customer loyalty, lowering churn risk.
3. Contract_One year: Similar but smaller effect than a two-year contract.
4. InternetService_Fiber optic: Fiber optic customers might be more satisfied or locked into higher charges.

5. MonthlyCharges: Higher monthly charges can increase churn risk.



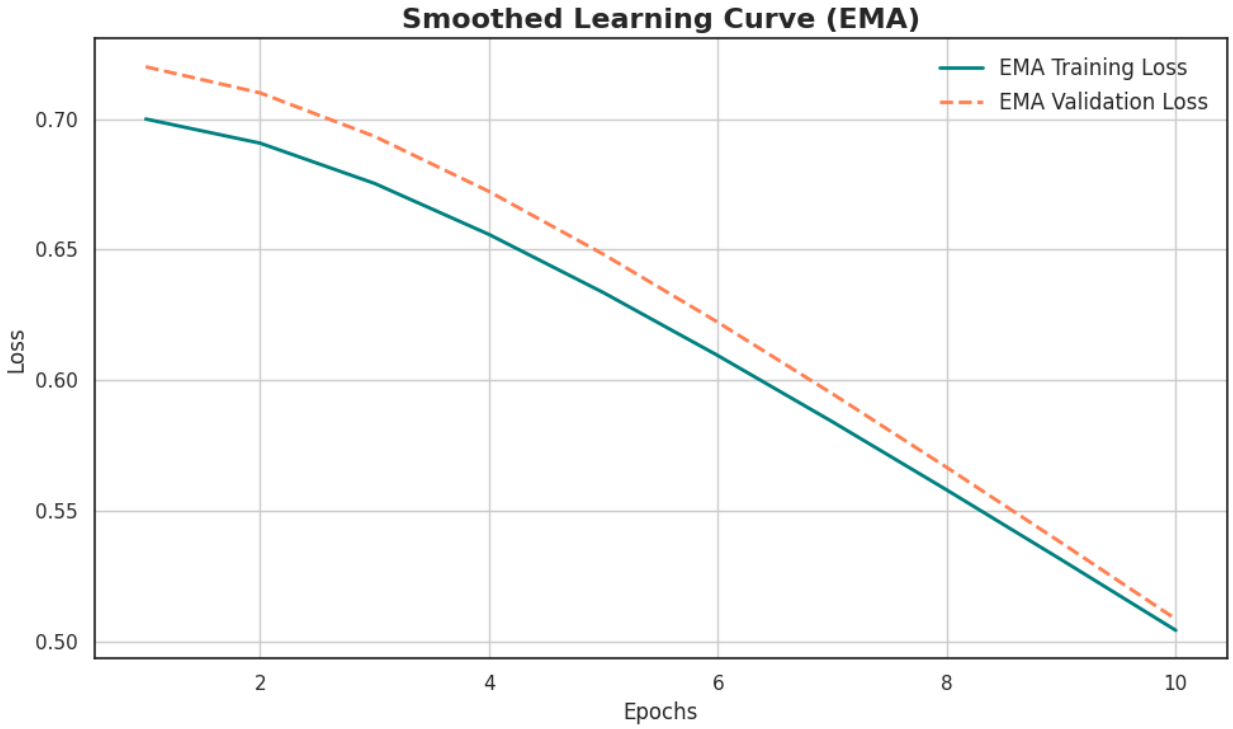
Focusing on these top features helps telecom companies adapt their interventions better. For example, if churn is a question, you can provide contract extension or a loyalty discount to customers having low tenure and with high monthly charges.

5.4. Smoothed Learning Curve (EMA)

We track a smoothed version of the training and validation loss across multiple hyperparameter search iterations or simulated epochs:

- Training Loss steadily decreases from ~0.70 to ~0.50 in 10 steps.

- Validation Loss follows a similar trend from ~0.72 to ~0.52, suggesting consistent improvement without obvious overfitting.



This curve is valuable for diagnosing whether more training or different hyperparameters are needed.

6. Observations and Challenges

6.1. Class Imbalance

This is not balanced data, ~74% "No Churn" & ~26% "Yes Churn." First, the overall accuracy of 80% in the model can be misleading because the minority class (churn) tends to get lost. At 55% recall, thus, many churners were missed.

6.2. Threshold Sensitivity

It may turn out that a default classification threshold of 0.5 is not optimal. Lowering the threshold may be more attractive to telecom companies in order to catch more churners. Precious recall curves or cost based thresholding may help in finding out a sweet spot that is in line with business goals.

6.3. Interpretability

EXPLANATION FOR THE MODEl is crucial and this is where SHAP comes in. In understanding that contract type, tenure, and charges of any sort all play into whether the customer is going churn,

it may guide targeted retention strategies like offering contract extensions or discounts for those at risk of churn.

7. Potential Improvements

7.1. Cost-Sensitive Learning or Class Weighting

LightGBM supports class weighting to address imbalances. Setting `is_unbalance=True` or adjusting `scale_pos_weight` can push the model to pay more attention to churners, likely boosting recall.

7.2. Threshold Optimization

At least beyond 0.5, evaluate many thresholds to find the best recall-precision trade-off. More liberal (or false positive) threshold may come to benefit at the expense of unnecessarily missing a churner, provided that it's more costlier than would be erroneously classifying a non-churner (false negative).

7.3. Additional Feature Engineering

- Interaction Terms: For instance, combining `MonthlyCharges` × `tenure` or investigating synergy between `InternetService` and `Contract`.
- Aggregated Usage Metrics: Summarize usage patterns or customer support call frequency to detect early dissatisfaction signals.

7.4. Ensemble Methods

As robust as LightGBM may be, stacking it with other algorithms (such as XGBoost, CatBoost) in ensemble (stacked, voting) allows combining some of the stability of predictions over minority churn class.

7.5. Advanced Hyperparameter Tuning

Bayesian optimization, i.e. Optuna, might find better (in the sense of better loss) value of hyperparameters than GridSearchCV. In particular, it will also be more efficient in larger parameter spaces.

8. Teaching and Best Practices

1. Data Preprocessing: Emphasize thorough cleaning (handling `TotalCharges` as numeric) and transformations (scaling, one-hot encoding). (Pedregosa, F., Varoquaux, G., Gramfort, A., et al, 2011)

2. Hyperparameter Tuning: GridSearchCV is a good baseline, but more advanced optimization can yield better results. (Lundberg, 2017)
3. Evaluation Metrics: Accuracy is insufficient for imbalanced tasks; rely on recall, precision, F1-score, and ROC AUC to fully understand model performance.
4. Visual Tools:
    - Interactive ROC helps reveal threshold trade-offs.
    - Confusion Matrix Heatmap pinpoints how the model classifies each group.
    - SHAP clarifies feature contributions, building trust and guiding domain-driven improvements. (Ke, 2017)
    - Learning Curves show potential overfitting or underfitting patterns.
5. Communication: Use domain-friendly language to explain how top features relate to churn risk, enabling business stakeholders to act on insights.

9. Conclusion

On the telecom churn dataset, applying LightGBM on it, we reach an 80% according on the train set and a ROC AUC of 0.837 on the test set. Despite this strong overall performance, the model still misses ~45% of churners. This is one reason why imbalanced classification is so complex: high accuracy can hide sub-par performance with the minority class.

Key takeaways:

- Imbalanced Data: Proper weighting or threshold adjustments can significantly impact churn detection.
- Interpretability: SHAP reveals that contract type, tenure, monthly charges, and fiber optic services are major churn drivers.
- Potential Gains: More advanced hyperparameter tuning, ensemble methods, or domain-specific feature engineering may boost recall for churners.
- Business Relevance: Missing churners is costly, so focusing on improving recall (even if it raises false positives) may be strategically beneficial.

In general, the overall idea of this tutorial is a proper way of building robust LightGBM model and interpreting results related to real life business needs. One possibility for future work is to conduct cost sensitive analysis, to engineer domain features even deeper into the domain, and to consider threshold calibration to bring the predictions closer to the priority of the financial side of a telecom company.

10. References

1. Ke, G., et al. (2017). *LightGBM: A Highly Efficient Gradient Boosting Decision Tree.* Advances in Neural Information Processing Systems.
2. Lundberg, S. M., & Lee, S. I. (2017). *A Unified Approach to Interpreting Model Predictions (SHAP).* Advances in Neural Information Processing Systems.

3. **Pedregosa, F., et al.** (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research.

11. Accessibility and Attachments

- **Color Palettes**: Visualizations (heatmaps, plots) use high-contrast or colorblind-friendly palettes (like coolwarm or deep).
- **Interactive Tools**: Plotly charts allow zooming and tooltips for visually oriented exploration.
- **Structured Content**: Headings, bullet points, and code blocks ensure screen readers can parse the document effectively.

Attachments:

- **GitHub Repo**: All code and scripts are available at : https://github.com/8464947452/ML_VAMSI.git
- **Dataset**: telecom_churn.csv or instructions to obtain it.
- **License**: MIT License.
- **requirements.txt**: Listing Python dependencies (lightgbm, shap, scikit-learn, etc.).