Project Report: Customer Churn Prediction System

Author: Ashwin
Technologies: Python, XGBoost, Scikit-Learn, Pandas

---

1. Abstract

This project focuses on developing a robust Machine Learning solution to predict customer churn in a telecommunications dataset. By leveraging the XGBoost algorithm and implementing advanced techniques such as **Stratified K-Fold Cross-Validation**, **Hyperparameter Tuning (GridSearchCV)**, and **Class Weighting**, the final model achieves a strong balance between accuracy (76%) and recall (76%), specifically optimizing for the identification of at-risk customers while mitigating overfitting.

---

2. Introduction

2.1 Background

Customer churn—when customers stop doing business with a company—is a critical metric for subscription-based businesses. Acquiring a new customer is significantly more expensive than retaining an existing one. Therefore, predicting which customers are likely to leave allows companies to take proactive retention measures.

2.2 Problem Statement

The dataset presents a typical real-world challenge: **Class Imbalance**. Approximately 73% of customers stay, while only 27% churn. Standard machine learning models trained on such data tend to be biased towards the majority class, achieving high accuracy by simply predicting “No Churn” for everyone, but failing to identify the actual churners.

2.3 Objectives

- To build a predictive model using **XGBoost**.
- To address class imbalance without losing valuable data (avoiding excessive downsampling).
- To rigorously tune hyperparameters to prevent **Overfitting**.
- To ensure the model generalizes well using **Stratified Cross-Validation**.

---

3. Methodology

3.1 Data Preprocessing

The raw data contained categorical variables (Gender, Payment Method, etc.) and numerical variables (Tenure, Charges). - **Data Cleaning:** Handled missing values in `TotalCharges` by converting empty strings to numeric zeros. - **Encoding:** Utilized `LabelEncoder` to transform categorical text into machine-readable integers. - **Splitting:** Employed `train_test_split` with stratification to ensure the test set mirrored the real-world distribution of churners.

3.2 Advanced Techniques Implemented

To ensure a production-grade solution, the following five key strategies were implemented:

1. **Model Selection (XGBoost):**
   - Moved from Random Forest to **XGBoost** (Extreme Gradient Boosting) due to its superior performance on tabular data and built-in L1/L2 regularization capabilities.
2. **Addressing Class Imbalance (Class Weighting):**
   - Instead of using SMOTE (which generated synthetic data causing overfitting) or Downsampling (which discarded 50% of data), we implemented **`scale_pos_weight`**.
   - Calculated a weight ratio of approx. **2.77**, penalizing the model 2.77x more for missing a churner than a non-churner.
3. **Hyperparameter Tuning (GridSearchCV):**
   - Exhaustively searched a predefined grid of 36 parameter combinations.
   - Optimized for: `learning_rate`, `max_depth`, `n_estimators`, `subsample`, and `colsample_bytree`.
4. **Addressing Overfitting:**
   - Restricted `max_depth` to **5** (preventing the model from memorizing noise).
   - Used `subsample=0.8` to introduce randomness during training.
   - **Result:** Reduced the generalization gap (Train Accuracy - Test Accuracy) from a dangerous **19%** to a stable **~7%**.
5. **Stratified K-Fold Cross-Validation:**
   - Used `StratifiedKFold(n_splits=3)` during tuning.
   - Ensured every validation fold contained exactly the same proportion of churners (27%) as the full dataset, preventing biased evaluation scores.

---

4. Results & Evaluation

4.1 Performance Metrics

The final optimized model yielded the following results on the Test Set:

Metric	Value	Interpretation
Accuracy	**76%**	The model is correct 3 out of 4 times overall.
Recall (Churn)	**76%**	**Critical Success.** The model correctly identifies 76% of all customers who are actually leaving.
Precision (Churn)	**53%**	While acceptable, the model trades some precision (more false alarms) to ensure high recall.
F1-Score	**0.62**	Indicates a balanced trade-off between Precision and Recall.

4.2 Overfitting Analysis

- **Training Accuracy:** ~83%
- **Test Accuracy:** ~76%
- **Gap:** ~7%
- **Conclusion:** The small gap indicates the model has learned general patterns rather than memorizing the training data.

---

5. Conclusion

The project successfully delivered a highly effective Customer Churn Prediction system. By moving beyond basic modeling to implement Class Weighting and rigorous Hyperparameter Tuning, we solved the "Accuracy Paradox" where models look good but fail to catch churners. The final XGBoost model is robust, generalized, and ready for deployment to assist methodically in customer retention efforts.

6. Future Scope

- **Feature Engineering:** Creating composite features like “Tenure Years” or “Family Bundle” flags.
- **deployment:** Wrapping the model in a Flask/FastAPI service for real-time predictions.
- **Threshold Tuning:** Adjusting the probability threshold (e.g., from 0.5 to 0.4) to further increase Recall if business needs dictate.