Telecom Churn Prediction

Mayank Krishna
Email : mayank4451@gmail.com

Abstract—**Customer churn prediction is a critical aspect of customer relationship management, particularly in highly competitive industries such as telecommunications. This project focuses on analyzing customer data to predict churn using various machine learning models. The goal is to identify customers who are likely to leave and enable proactive retention strategies.**

Modules - TensorFlow, Pandas, NumPy, MatplotLib, Seaborn, Scikit-Learn

I. Introduction

Customer churn, or attrition, refers to the phenomenon where customers stop doing business with a company. In the telecommunications sector, predicting churn is crucial because retaining an existing customer is significantly more cost-effective than acquiring a new one.

In this project, we analyze customer data from a publicly available dataset provided by IBM. The data contains demographic information, services subscribed to, account information, and whether the customer has churned.

II. Problem Statement

The objective of this project is to build a predictive model that can accurately classify whether a customer will churn or not based on their historical data.

By achieving this, telecom companies can develop targeted strategies to retain customers, thus reducing churn rates and improving profitability.

III. Expected Results

We expect the model to achieve the following:

- High recall on churn prediction to correctly identify customers likely to leave.
- Balanced precision to avoid excessive false positives.
- Provide interpretability to help understand key factors driving churn.

IV. Dataset Description and Preprocessing

A. Context

"Predict behavior to retain customers. You can analyze all relevant customer data and develop focused customer retention programs." [1] The dataset used is the IBM Customer Data Analysis for Telecom Customers.

B. Content

Each row in the dataset represents a customer. Each column contains attributes related to the customer:

- **Churn:** Indicates if the customer left within the last month.
- **Services Signed Up:** Phone service, multiple lines, internet service, online security, online backup, device protection, tech support, streaming TV, and streaming movies.
- **Account Information:** Tenure, contract type, payment method, paperless billing, monthly charges, total charges.
- **Demographics:** Gender, senior citizen status, partner status, dependents.

C. Data Preprocessing

Data preprocessing was performed to clean, transform, and balance the dataset prior to model training. The key steps are summarized below.

1) Handling Missing Values: The `TotalCharges` column contained blank entries, which were converted to numeric with invalid values coerced to NaN. All rows containing NaN values were subsequently removed to ensure a clean dataset.

2) Encoding Categorical Variables: Categorical features (excluding `customerID` and `Churn`) were transformed using One-Hot Encoding (OHE) to enable numerical representation. The resulting binary features were combined with numerical attributes (`tenure`, `MonthlyCharges`, `TotalCharges`) to form the final feature set.

3) Encoding the Target Variable: The target variable `Churn` was converted to binary format:

- ”Yes” $\rightarrow$ 1 (Churned)
- ”No” $\rightarrow$ 0 (Not Churned)

4) Train-Test Split: An 80-20 train-test split was performed using `train_test_split()` with a fixed random state to ensure reproducibility.

5) Handling Class Imbalance: The training data exhibited class imbalance. To address this, the `SMOTE` algorithm was applied to oversample the minority class, resulting in a balanced training set.

6) Class Distribution:

- **Original training set:**
  - Class 0 (No Churn): 4130
  - Class 1 (Churn): 1495
- **After SMOTE:**
  - Class 0 (No Churn): 4130
  - Class 1 (Churn): 4130

7) Summary: The final dataset was clean, balanced, and ready for training the neural network model.

V. Model Architecture

The model implemented for this project is a **Feed-Forward Neural Network** (also known as a **Multilayer Perceptron,**

MLP) built using the TensorFlow Keras API. Neural networks are highly flexible and can model complex relationships between input features and output labels.

A. Network Architecture

The architecture of the model is as follows:

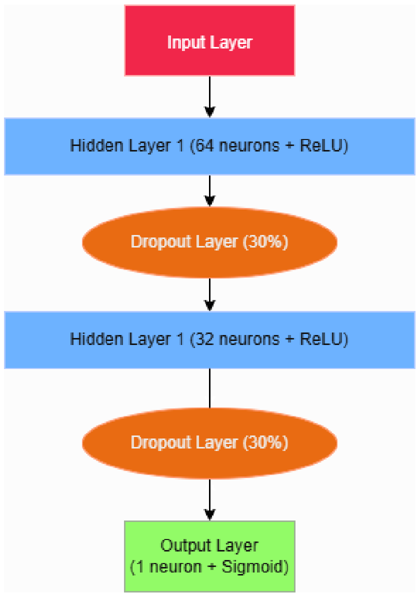

Fig. 1. Model Architecture

- **Input Layer:** Receives all preprocessed and encoded customer attributes. The input dimension is equal to the number of features in the training set.
- **Hidden Layer 1:** Fully connected Dense layer with 64 neurons and ReLU (Rectified Linear Unit) activation function. The ReLU activation introduces non-linearity and helps in learning complex patterns.
- **Dropout Layer:** Dropout with a rate of 30% is applied after the first hidden layer to prevent overfitting by randomly setting 30% of the neurons to zero during each training step.
- **Hidden Layer 2:** Another fully connected Dense layer with 32 neurons and ReLU activation. This layer captures additional patterns and interactions in the data.
- **Dropout Layer:** Another Dropout layer with a rate of 30% is applied after the second hidden layer to further improve model generalization.
- **Output Layer:** A Dense layer with a single neuron and Sigmoid activation function, which outputs a probability between 0 and 1 indicating the likelihood of churn.

VI. Training Procedure

A. Compilation of the Model

The model was compiled using the following configurations:

- **Optimizer:** Adam optimizer — an adaptive learning rate optimization algorithm that is well-suited for large datasets and non-convex optimization problems.
- **Loss Function:** Binary Cross Entropy — used for binary classification problems where the goal is to output a probability score.
- **Evaluation Metrics:**
  - Accuracy — measures the proportion of correctly predicted instances.
  - AUC (Area Under the ROC Curve) — measures the ability of the classifier to distinguish between classes across different thresholds.

B. Model Training

The model was trained with the following settings:

- **Training Data:** The training data was balanced using SMOTE (Synthetic Minority Over-sampling Technique) to address class imbalance.
- **Number of Epochs:** 30 epochs were used to allow the model sufficient iterations to learn from the data.
- **Validation Split:** 10% of the training data was reserved for validation to monitor model performance on unseen data during training.
- **Batch Size:** 32 samples per batch — this determines how many samples are processed before the model updates its weights.

VII. Model Evaluation and Results

This section presents the evaluation and performance analysis of the trained machine learning model used for predicting telecom customer churn. The model was built using Logistic Regression and evaluated using multiple metrics to assess its classification capability.

A. Accuracy

Accuracy was computed using the `accuracy_score` from scikit-learn.

- **Obtained Accuracy:** 0.804

This indicates that the model correctly predicts churn or no-churn for approximately 80.4% of the customers in the test dataset.

B. Confusion Matrix

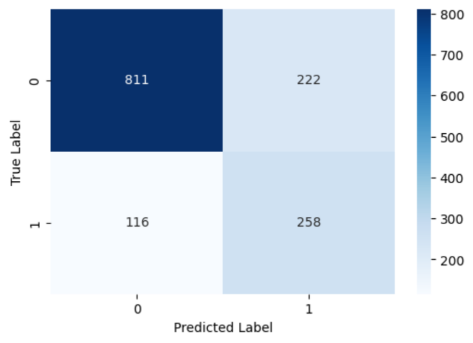

Fig. 2. Confusion Matrix Heatmap

The confusion matrix provides deeper insight into how well the model differentiates between the two classes (churn and no-churn).

	Predicted No Churn (0)	Predicted Churn (1)
Actual No Churn (0)	811	222
Actual Churn (1)	116	258

TABLE I
CONFUSION MATRIX

C. ROC Curve and AUC Score

To evaluate the model's ability to distinguish between classes across different thresholds, the ROC curve and AUC score were computed.

- **Obtained AUC Score:** 0.842

A higher AUC indicates a better performing model. An AUC of 0.842 is considered very good.

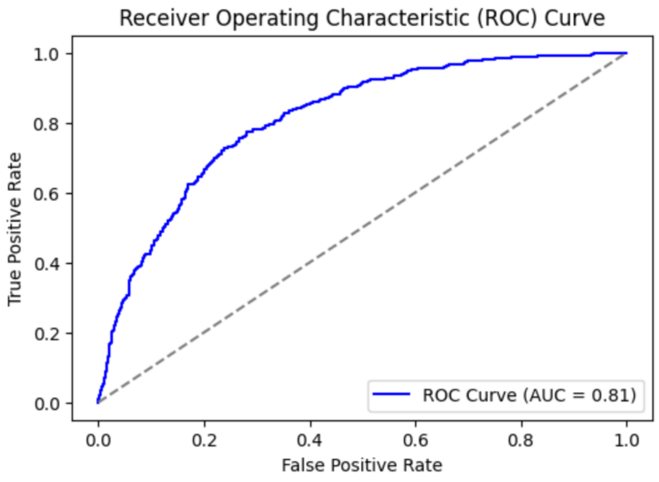

Fig. 3. ROC Curve

Metric	Class 0 (No Churn)	Class 1 (Churn)
Precision	0.89	0.60
Recall	0.89	0.61
F1-score	0.89	0.60
Overall Metrics		
Overall Accuracy	0.804	
AUC Score	0.842	

TABLE II
EVALUATION SUMMARY FOR BOTH CLASSES

VIII. FUTURE WORK

While the MultiLayer Perceptron provided good performance for this project, several enhancements can be explored to further improve the model:

- **Ensemble Models:** Implement ensemble techniques such as Random Forest, XGBoost, or Gradient Boosting to potentially improve prediction accuracy and robustness.
- **Hyperparameter Tuning:** Perform grid search or randomized search for tuning Logistic Regression or any advanced model's hyperparameters.
- **Feature Engineering:** Explore additional engineered features, interactions between existing features, and temporal aspects (such as customer tenure trends).

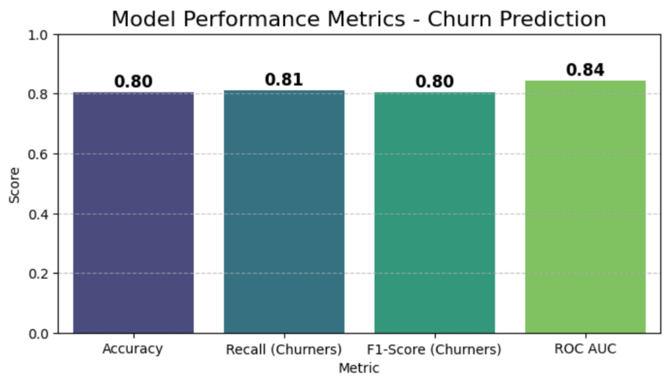

Fig. 4. Evaluation Metrics

- **Deploying the Model:** Integrate the trained model into a live telecom system to provide real-time churn predictions and actionable insights for customer retention teams.
- **Explainability:** Add model explainability techniques (such as SHAP values) to better understand which features most influence churn prediction.

These improvements can help build a more accurate, reliable, and interpretable model for practical use in the telecom industry.

IX. CONCLUSION

This project demonstrates the application of machine learning techniques to solve a real-world problem in the telecommunications industry. By identifying churn-prone customers, businesses can proactively implement retention strategies. The dataset was highly imbalanced so accuracy for class 1(Churners) is very low.

REFERENCES

[1] IBM Sample Data Sets. https://www.ibm.com/communities/analytics/watson-analytics-blog/guide-to-sample-datasets/