Customer Churn Prediction

Project report

Project Overview

Project Objective

In this project, our objective is to develop a predictive model that can identify customers who are likely to churn. By accurately predicting churn, the business can proactively engage at-risk customers with targeted retention strategies, thereby reducing churn rates and improving overall business performance.

By leveraging machine learning techniques, we can analyze historical customer data to uncover patterns and signals associated with churn. The predictive model will enable the business to.

- **Identify high-risk customers early**
- **Prioritize retention efforts and allocate resources efficiently**
- **Reduce revenue loss due to customer attrition**
- **Improve customer satisfaction by addressing issues proactively**

Dataset Information

- **Source**: Telco Customer Churn dataset
- **Size**: 7,043 customers
- **Features**: 21 columns including demographics, services, and account information
- **Target Variable**: Customer churn (binary classification)

Key Steps

1. Data Analysis

2. Data Preprocessing

3. Model Development

4. Model Performance

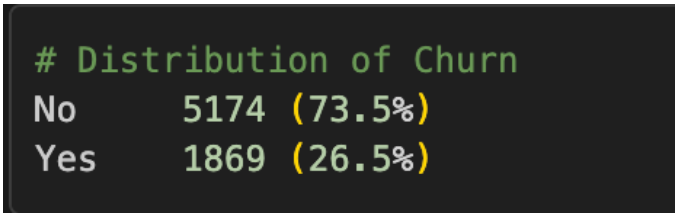
5.Model Deployment

Data Analysis

1. Initial Data Exploration

- Dataset contains 7,043 customer records with 21 features
- Mix of categorical and numerical variables
- Target variable: Churn (binary - Yes/No)
- Basic checks revealed minimal missing values

2. Target Variable Analysis



Insight: Class imbalance identified - only ~26.5% customers churned

3. Data Type Handling

- Converted TotalCharges to numeric format
- Transformed SeniorCitizen to categorical
- Created binary target column from Churn

4. Categorical Features Analysis

- Analyzed distributions of:
  - Demographics (gender, senior citizen status)
  - Services subscribed
  - Contract types
  - Payment methods **Insights**:
- Month-to-month contracts show higher churn
- Fiber optic service users more likely to churn
- Electronic payment methods associated with lower churn

5. Numerical Features Analysis

Examined:

- Monthly charges
- Total charges
- Tenure **Insights**:
- Strong negative correlation between tenure and churn
- Higher monthly charges associated with increased churn
- Total charges show positive correlation with customer retention

6. Feature Relationships

Used two methods:

1. **Correlation Analysis for Numerical Features**:
   - Monthly and total charges highly correlated
   - Tenure strongly related to total charges
2. **Cramér's V Analysis for Categorical Features**:
   - Strong association between InternetService and online services
   - Contract type strongly related to payment method
   - Services often bundled together

7. Outlier Analysis

- Used box plots for numerical features
- No significant outliers detected beyond IQR
- Data relatively clean in terms of extreme values

8. Key Business Insights

1. **Customer Profile Risk Factors**:
   - Short tenure customers
   - Month-to-month contracts
   - High monthly charges
   - Fiber optic service subscribers
2. **Retention Indicators**:
   - Long-term contracts
   - Bundled services
   - Paperless billing
   - Electronic payment methods
3. **Service Impact**:
   - Technical support importance
   - Security services correlation with retention
   - Online backup services value

9. Data Quality Insights

- Overall high-quality data
- Minimal missing values
- Well-structured features
- Good balance of categorical and numerical variables
- No significant data cleaning required

Data Preprocessing Steps in Customer Churn Project

1. Initial Data Assessment and Cleaning

- The dataset consisted of 7,043 customers with 21 features.
- Missing values were checked and identified.
- The target variable (“Churn”) was found to be imbalanced, with about 26.5% of customers having churned.
- The “customerID” column, which served no predictive purpose, was dropped.

2. Data Type Transformations

- The “TotalCharges” column, initially read as an object due to formatting issues, was converted into a numeric format for accurate analysis.
- The “SeniorCitizen” column, originally stored as a numerical type (0 or 1), was converted into a categorical variable to reflect its binary nature more appropriately.
- A new binary target variable was created from the “Churn” column, where ‘Yes’ was mapped to 1 and ‘No’ to 0.

3. Feature Separation

- Numerical features like tenure, MonthlyCharges, and TotalCharges were identified.
- Categorical features included demographic information (e.g., gender, SeniorCitizen) and service usage data (e.g., PhoneService, InternetService).
- Features were grouped into numerical and categorical categories to enable tailored preprocessing.

4. Pipeline Construction

- **Numerical Transformer**: Since the numerical features were already in usable form, they were passed through directly without modification initially.
- **Categorical Transformer**: Missing values were imputed using the most frequent category for each feature. Then, categorical features were transformed using one-hot encoding, allowing the model to handle categorical data effectively. Any unknown categories during inference were ignored to prevent errors.

5. Combined Preprocessing Pipeline

- A unified preprocessing pipeline was created that applied the numerical transformations to the numerical features and the categorical transformations to the categorical features. This ensured all features were prepared appropriately before feeding into the model.

6. Train-Test Split

- The dataset was split into training and testing sets using an 80:20 ratio.
- Stratified sampling was applied to preserve the original class distribution of the churn variable across both sets.
- A fixed random state ensured reproducibility.

7. Class Imbalance Handling

- SMOTEENN, a combination of oversampling (SMOTE) and undersampling (Edited Nearest Neighbors), was used to handle the class imbalance.
- This method helped increase the proportion of churn cases in the training set from 26.5% to nearly 50%, allowing the model to learn better from minority class examples.

8. Final Pipeline Integration

- The complete modeling pipeline included:
  - The preprocessing step for cleaning and transforming data.

    - Standard scaling to normalize numerical features.
    - Resampling using SMOTEENN to address class imbalance.
    - The final machine learning classifier.

This structured pipeline ensured:

- Clean and consistent data inputs.
- Balanced representation of target classes.
- Prevention of data leakage by applying transformations only to training data.

Model Building and Optimization Process

1. Initial Model Setup (Logistic Regression)

- Created preprocessing pipeline with:
    - Numerical features: Passthrough transformer
    - Categorical features: SimpleImputer + OneHotEncoder
    - StandardScaler for overall scaling

2. Model Optimization with GridSearchCV

A. Parameter Grid Setup

- Defined parameter grids for each model.
- Cross-validation: 5-fold
- Scoring metric: Recall (focus on identifying churners)
- Parallel processing: n_jobs=-1

B. Models Evaluated

1. **Base Models**:
    - Logistic Regression
    - Decision Tree
    - Random Forest
    - SVM
    - Gradient Boosting
    - XGBoost
    - LightGBM

3. Performance Metrics (Before SMOTEENN)

- Best performing models:
    - Gradient Boosting: 81% accuracy, 53% recall
    - Random Forest: 80% accuracy, 53% recall
    - XGBoost: 80% accuracy, 52% recall

4. Class Imbalance Handling

- Implemented SMOTEENN:

```
resampled_pipeline = ImbPipeline([
    ('preprocessor', preprocessor),
    ('resampler', SMOTEENN()),
    ('classifier', model)
])
```

5. Final Results (After SMOTEENN)

- **Logistic Regression**:
  - Highest recall: 84%
  - Lower precision: 45%
- **LightGBM (Best Overall)**:
  - Balanced recall: 78%
  - Better precision: 53%
  - Highest F1-score: 0.63

6. Model Selection Criteria

- Prioritized recall while maintaining acceptable precision
- Considered overall accuracy and F1-score
- Selected LightGBM for final deployment due to:
  - Balanced performance metrics
  - Good generalization
  - Efficient handling of both numerical and categorical features

Model Deployment

- Saved best performing LightGBM model
- Implemented complete pipeline including preprocessing
- Model ready for production use via joblib