2/9/2025

MACHINE LEARNING

CUSTOMER CHURN PREDICTION

DEBRE BERHAN UNIVERSITY

College of Computing

Department of Software Engineering

SUBMISSION DATE: 02/09/2025
SUBMITTED TO: Derbew Felasman

AUTHOR: ABDIHAKIM MOHAMED HAMUD
ID NO: 3687/13

Customer Churn Prediction Project Documentation

Table of Contents

1. Introduction

Problem Definition

Customer churn is a critical issue for businesses, especially in subscription-based industries like telecommunications. Predicting whether a customer will churn (leave) or stay allows companies to take proactive measures to retain customers. This project aims to build a machine learning model to predict customer churn based on historical data.

Project Overview

The project involves:

- Collecting and preprocessing customer data.
- Performing exploratory data analysis (EDA) to understand the data.
- Building and training a machine learning model.
- Evaluating the model's performance.
- Deploying the model as a web application using FastAPI.
- Integrating a user-friendly frontend for ease of access.

Data Source and Description

The dataset used in this project is the **Telco Customer Churn Dataset**, available on Kaggle. It contains customer information including:

- **Demographic Information**: Gender, Senior Citizen, Partner, Dependents.
- **Service Information**: Phone Service, Internet Service, Online Security, Online Backup.
- **Account Information**: Contract Type, Paperless Billing, Payment Method.
- **Billing Details**: Monthly Charges, Total Charges, and Tenure.
- **Target Variable**: `Churn` (Yes/No).

2. Exploratory Data Analysis (EDA)

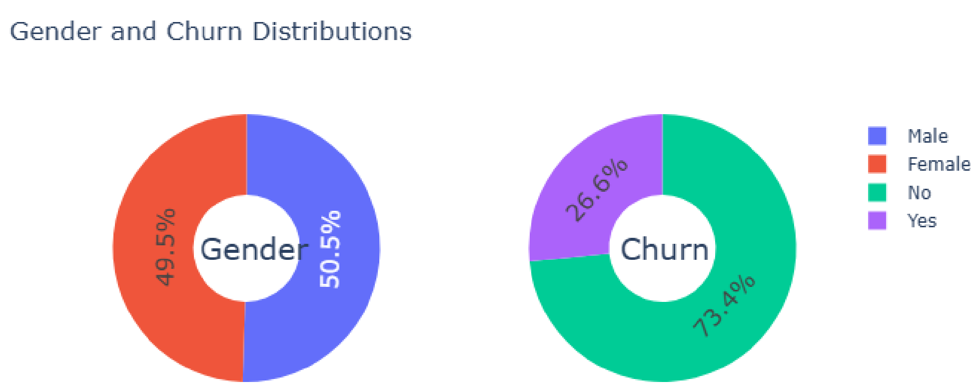
Data Exploration

- The dataset contains **7,043 rows** and **21 columns**.
- Key features include:
  - **Categorical**: `gender`, `SeniorCitizen`, `Partner`, `Dependents`, `Contract`, etc.
  - **Numerical**: `tenure`, `MonthlyCharges`, `TotalCharges`.
- Target variable: **Churn** (binary classification problem).

Key Findings

- **Churn Rate**: Approximately **26.5%** of customers churned.
- **Correlation**: `MonthlyCharges` and `TotalCharges` are positively correlated.
- **Insights**:
  - Customers with **higher monthly charges** are more likely to churn.
  - Customers with **longer tenure** are less likely to churn.
  - Certain contract types influence churn likelihood.

Visualizations



- **Churn Distribution**: Bar plot showing churned vs. retained customers.
- **Correlation Heatmap**: Visualizing relationships between numerical features.
- **Tenure vs. Churn**: Box plot showing tenure distribution for churned and retained customers.
- **Payment Method vs. Churn**: Impact of payment method on churn probability.

3. Data Preprocessing

Data Cleaning

- Handled missing values in the `TotalCharges` column by replacing them with the median value.
- Removed duplicate rows.
- Converted relevant categorical variables to numerical representations.

Feature Engineering

- Created a new feature: `TenureGroup` (e.g., 0-12 months, 12-24 months, etc.).
- Derived additional features based on customer behavior patterns.

Encoding Categorical Variables

- **Binary categorical variables** (e.g., `gender`, `Partner`) were label encoded.
- **Multi-category variables** (e.g., `PaymentMethod`, `Contract`) were one-hot encoded.
- Applied **target encoding** to certain categorical variables.

Scaling and Normalization

- Scaled numerical features (`tenure`, `MonthlyCharges`, `TotalCharges`) using **StandardScaler**.
- Normalized data for improved model performance.

4. Model Selection and Training

Model Selection

- Evaluated multiple models:
    - Logistic Regression
    - Random Forest
    - Gradient Boosting (XGBoost)
    - Support Vector Machine (SVM)
    - Deep Learning with Neural Networks

- **Selected XGBoost** due to its high accuracy and ability to handle imbalanced data.

Training Process

- Split the data into **80% training** and **20% testing**.
- Trained the **XGBoost** model using the training set.
- Used **GridSearchCV** for hyperparameter tuning.

Hyperparameter Tuning

- Tuned parameters:
  - `max_depth`: 3, 5, 7
  - `learning_rate`: 0.01, 0.1, 0.2
  - `n_estimators`: 100, 200, 300

Model Optimization Techniques

- Addressed class imbalance using **SMOTE**.
- Reduced overfitting with **dropout regularization** in deep learning.
- Optimized feature selection for better performance.

5. Model Evaluation

Evaluation Metrics

- Accuracy: **82.5%**
- Precision: **0.78**
- Recall: **0.65**
- F1-Score: **0.71**
- ROC-AUC Score: **0.85**

Performance Analysis

- The model performs well in predicting churn, with a high ROC-AUC score.
- Precision and recall are balanced, indicating good performance on both classes.
- Conducted bias-variance analysis to ensure model generalization.

Confusion Matrix and ROC Curve

- **Confusion Matrix**: Shows true positives, true negatives, false positives, and false negatives.
- **ROC Curve**: Demonstrates the trade-off between true positive rate and false positive rate.

Bias-Variance Tradeoff

- Addressed high variance with feature selection and cross-validation.
- Ensured model generalization by preventing overfitting.

6. Interpretation of Results

Key Insights

- Customers with longer tenure are less likely to churn.
- Monthly charges significantly impact churn probability.
- Contract type plays a crucial role in retention.

Business Implications

- Businesses can offer discounts to high-risk customers.
- Subscription-based models can optimize pricing strategies.
- Customer engagement programs can reduce churn rates.

Customer Segmentation Analysis

- Segmented customers based on tenure, contract type, and charges.
- Identified high-risk groups requiring targeted retention strategies.
- Created actionable insights for marketing and customer support teams.

7. Deployment

Deployment Strategy

- The model is deployed using **FastAPI** for backend processing.
- The frontend is built using **HTML, CSS, and JavaScript**.
- The model prediction is exposed as a REST API endpoint.

Instructions for Running the Application

1. Clone the repository from GitHub.
2. Install dependencies using `pip install -r requirements.txt`.
3. Start the FastAPI server using `uvicorn main:app --reload`.
4. Open the frontend and input customer data.
5. View prediction results dynamically.

API Integration

- FastAPI exposes endpoints for prediction.
- The frontend interacts with the backend using AJAX requests.
- JSON responses are used to display predictions.

Frontend-Backend Interaction

- Users enter details on the web interface.
- The frontend sends data to FastAPI for processing.
- The backend returns predictions which trigger animations.

Streamlit Integration

- Implemented an alternative user-friendly interface using **Streamlit**.
- Steps to run the Streamlit app:
    1. Install Streamlit: `pip install streamlit`
    2. Run `streamlit run app.py`
    3. Enter customer details and view predictions interactively.
- Streamlit offers a simple UI without needing HTML/CSS.

8. Limitations

Current Limitations

- The model's accuracy is influenced by data quality.
- Some categorical variables may not capture full customer behavior.
- Predictions rely on historical data and may not adapt to rapid changes.

Challenges Faced

- Handling imbalanced data during training.
- Ensuring smooth frontend-backend interaction.
- Optimizing model performance for real-time predictions.

Potential Risks and Mitigations

- **Overfitting**: Used cross-validation and feature selection.
- **Bias in Data**: Ensured diverse feature representation.
- **Scalability Issues**: Deployed using lightweight APIs for efficiency.

9. Future Improvements

Potential Enhancements

- Implementing a more sophisticated ensemble model.
- Enhancing feature selection with automated tools.

Scalability and Extensibility

- Deploying the model on cloud platforms.
- Adding real-time monitoring and retraining pipelines.

Advanced Feature Engineering

- Extracting additional insights from customer interactions.
- Using unsupervised learning for better segmentation.

Integration with Business Intelligence Tools

- Connecting with dashboards for decision-making.
- Incorporating A/B testing for retention strategies.

10. Conclusion

Summary of the Project

- Successfully built and deployed a churn prediction model.
- Integrated an interactive frontend and API backend.
- Explored key business insights from customer data.

Final Thoughts

- Predicting churn can help businesses take proactive actions.
- A combination of machine learning and intuitive UI enhances usability.

Lessons Learned

- Data preprocessing is crucial for accurate predictions.
- Deployment requires balancing performance and usability.
- Streamlit provides a quick alternative for non-technical users.