Customer Churn Prediction Project Report

Team:

Mohamed Ahmed Mohamed Reda 21100914

Amr Khaled Gaber 21100834

George Nashaat Mosaed Saad 21100825

Ethar Ahmed Mohamed 23101369

1. Project Overview

The objective of this project was to develop a machine learning-based system to predict customer churn for a telecommunications company using the Telco Customer Churn dataset. The project encompassed data collection, preprocessing, feature extraction, model training, testing and evaluation, and deployment through a Streamlit web application. The final deliverable allows users to input customer data and receive churn predictions with probabilities.

2. Requirements

The project was designed to meet the following requirements:

- **Data Collection**: Obtain and load the Telco Customer Churn dataset.
- **Data Preprocessing**: Clean and prepare the data for modeling.
- **Feature Extraction**: Transform raw data into model-ready features.
- **Model Training**: Train multiple machine learning models with hyperparameter tuning.
- **Testing and Evaluation**: Evaluate model performance using relevant metrics and visualizations.
- **Deployment**: Develop a user-friendly application for churn prediction.

3. Key Steps and Implementation

3.1 Data Collection

- **Dataset**: The Telco Customer Churn dataset (WA_Fn-UseC_-Telco-Customer-Churn.csv) was used, containing 7043 customer records with 21 features, including demographic, service, and billing information, and a binary churn label (Yes/No).

- **Action**: Loaded the dataset using pandas for analysis and modeling.

3.2 Data Preprocessing

- **Cleaning**:
    - Handled missing values in TotalCharges by replacing empty strings with MonthlyCharges and converting to numeric.
    - Dropped the irrelevant customerID column.
    - Converted SeniorCitizen to string type ('0', '1') for consistency.
    - Transformed Churn labels to binary (0 for No, 1 for Yes).
- **Class Imbalance**:
    - Identified imbalance: 5174 non-churn (73%) vs. 1869 churn (27%).
    - Applied Synthetic Minority Oversampling Technique (SMOTE) to balance the training data, ensuring models prioritize the minority class (churn).
- **Data Splitting**:
    - Split data into training (64%), validation (16%), and test (20%) sets using train_test_split with random_state=1 for reproducibility.

3.3 Feature Extraction

- **Categorical Features**: Encoded 16 categorical features (e.g., gender, contract, paymentmethod) using OneHotEncoder with drop='first' to avoid multicollinearity and handle_unknown='ignore' to manage unseen categories during prediction.
- **Numerical Features**: Scaled tenure, monthlycharges, and totalcharges using StandardScaler for consistent model input.
- **Pipeline**: Used make_column_transformer to combine preprocessing steps, ensuring consistent transformation across training, validation, and test sets.

3.4 Model Training

- **Models Trained**:
    - Logistic Regression
    - Random Forest
    - XGBoost

- Support Vector Machine (SVM)

- **Hyperparameter Tuning**:
  - Performed 5-fold cross-validation using GridSearchCV with F1-score as the scoring metric.
  - Tuned parameters:
    - Logistic Regression: C in [0.1, 1, 10]
    - Random Forest: n_estimators in [100, 200], max_depth in [10, 20, None]
    - XGBoost: n_estimators in [100, 200], learning_rate in [0.01, 0.1], max_depth in [3, 5]
    - SVM: C in [0.1, 1, 10], kernel in ['linear', 'rbf']
- **Outcome**: Identified the best model for each algorithm based on F1-score.

3.5 Testing and Evaluation

- **Metrics**:
  - Evaluated models on the test set using accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC).
  - Results:
    - **Logistic Regression**: Accuracy: 0.76, Precision: 0.51, Recall: 0.82, F1-Score: 0.63, AUC: 0.86
    - **Random Forest**: Accuracy: 0.79, Precision: 0.56, Recall: 0.63, F1-Score: 0.59, AUC: 0.84
    - **XGBoost**: Accuracy: 0.78, Precision: 0.54, Recall: 0.71, F1-Score: 0.62, AUC: 0.86
    - **SVM**: Accuracy: 0.77, Precision: 0.52, Recall: 0.71, F1-Score: 0.60, AUC: 0.82
- **Visualizations and Outputs**:
  - Generated confusion matrix heatmaps for each model, saved as PNG files in the result folder.
  - Produced ROC curve plots, saved as PNG files, to visualize AUC performance.

- Saved classification reports as text files for detailed per-class metrics.
- Exported feature importance for Random Forest and XGBoost as CSV files to identify key churn drivers.
- Saved model comparison metrics as a CSV file and best hyperparameters as a JSON file.

- **Analysis**:
  - Logistic Regression excelled in recall (0.82), ideal for identifying churners.
  - Random Forest had the highest accuracy (0.79) but lower recall.
  - XGBoost and SVM offered balanced performance, with XGBoost matching Logistic Regression's AUC (0.86).
  - The results were deemed sufficient for deployment due to strong AUC values and reasonable F1-scores.

3.6 Deployment

- **Model Saving**:
  - Saved the best model (highest F1-score, likely Logistic Regression or XGBoost) as result/best_churn_model.pkl.
  - Saved the preprocessing transformer as result/transformer.pkl for consistent input transformation.
- **Streamlit Application**:
  - Developed a Streamlit app (app.py) to provide a user-friendly interface for churn prediction.
  - Features:
    - Input fields for all 19 features (16 categorical, 3 numerical) organized in two columns.
    - Slider for tenure and number inputs for monthlycharges and totalcharges.
    - Displays prediction ("Churn" or "Not Churn") and churn probability.
    - Visual indicators: red warning for churn, green success for non-churn.
  - Tested with a high-risk customer profile (e.g., tenure=2, contract=Month-to-month, internetservice=Fiber optic) to confirm churn prediction.

    - Verified model loading by adding a debug line to display the model type (e.g., LogisticRegression).
- **Outcome**: The app successfully loads the saved model, processes user inputs, and delivers accurate predictions, as confirmed by testing.

4. Challenges and Solutions

- **Class Imbalance**:
    - **Challenge**: The dataset was imbalanced (73% non-churn vs. 27% churn), risking biased models.
    - **Solution**: Applied SMOTE to oversample the churn class in the training set, improving recall for churn predictions.
- **Pandas FutureWarning**:
    - **Challenge**: Chained assignment warning for df['TotalCharges'].fillna(df['MonthlyCharges'], inplace=True).
    - **Solution**: Replaced with direct assignment: df['TotalCharges'] = df['TotalCharges'].fillna(df['MonthlyCharges']).
- **OneHotEncoder Error**:
    - **Challenge**: ValueError: Found unknown categories ['0'] in column 1 during transform due to inconsistent SeniorCitizen categories.
    - **Solution**: Set SeniorCitizen to string type ('0', '1') and added handle_unknown='ignore' to OneHotEncoder.
- **Scikit-learn Warning**:
    - **Challenge**: BaseEstimator._validate_data deprecation warning.
    - **Solution**: Noted as harmless; recommended updating scikit-learn to suppress in future versions.
- **Streamlit Integration**:
    - **Challenge**: Ensuring the app correctly used the saved model and transformer.
    - **Solution**: Verified model loading with debug output and tested with churn-prone data to confirm predictions.

5. Outcomes and Results

- **Model Performance**: Achieved AUC values of 0.82–0.86, with Logistic Regression and XGBoost leading in F1-score (0.63 and 0.62, respectively). High recall (0.82 for Logistic Regression) ensures effective churn identification.
- **Comprehensive Outputs**: Saved all relevant results in the result folder, including metrics, visualizations, and model artifacts, facilitating analysis and reporting.
- **User-Friendly App**: The Streamlit app provides an intuitive interface for stakeholders to input customer data and receive churn predictions, successfully tested with high-risk profiles.
- **Project Success**: All requirements (data collection, preprocessing, feature extraction, model training, evaluation, and deployment) were fully met, with robust solutions to technical challenges.

6. Future Improvements

- **Model Enhancement**: Explore additional algorithms (e.g., neural networks) or feature engineering to boost precision without sacrificing recall.
- **App Features**: Add visualizations (e.g., feature importance, confusion matrix) or support for batch predictions via CSV upload.
- **Deployment**: Host the Streamlit app on a cloud platform (e.g., Streamlit Cloud, AWS) for broader access.
- **Monitoring**: Implement model performance monitoring to detect data drift or degradation over time.

7. Conclusion

The customer churn prediction project successfully delivered a robust machine learning solution, from data processing to a deployable web application. The use of SMOTE, multiple models, and comprehensive evaluation ensured reliable predictions, while the Streamlit app provided an accessible interface for end-users. The project addressed all challenges effectively and lays a strong foundation for future enhancements.