Telco Customer Churn Prediction

Project Analysis

May 7, 2025

Contents

1 The Strategic Imperative of Churn Prediction **2**
- 1.1 The Business Problem: Customer Attrition in a Competitive Landscape . . . 2
- 1.2 Project Goal and Dataset Introduction . . . 2

2 Foundational Data Analysis and Preparation **2**
- 2.1 Data Loading and Initial Inspection . . . 2
- 2.2 Data Cleaning and Preprocessing . . . 3
- 2.3 Exploratory Data Analysis (EDA) . . . 3

3 Advanced Feature Engineering and Selection **3**
- 3.1 Encoding Categorical Features . . . 3
- 3.2 Creating New Features (Feature Engineering) . . . 3
- 3.3 Feature Scaling . . . 3

4 A Multi-Model Approach to Churn Prediction **3**
- 4.1 Splitting the Data . . . 3
- 4.2 Addressing Class Imbalance . . . 4
- 4.3 Model Selection and Training . . . 4
- 4.4 Hyperparameter Tuning with GridSearchCV . . . 4

5 Rigorous Model Evaluation and Business Insights **4**
- 5.1 Performance on the Test Set . . . 4
- 5.2 Detailed Classification Reports . . . 4
- 5.3 Confusion Matrix Visualization . . . 5
- 5.4 Model Interpretation and Feature Importances . . . 5

6 Complete Code **6**

1 The Strategic Imperative of Churn Prediction

1.1 The Business Problem: Customer Attrition in a Competitive Landscape

In the modern subscription-based economy, particularly within the highly competitive telecommunications industry, customer retention is a paramount strategic objective. The term "customer churn," also known as customer attrition, refers to the rate at which customers discontinue their subscriptions or services within a specific period. This metric is not merely an operational statistic; it is a direct indicator of a company's health, market position, and future profitability. A high churn rate can severely impact revenue streams and impede growth, as the cost of acquiring new customers is substantially higher than retaining existing ones.

The business case for focusing on churn reduction is compelling. Research has shown that it can cost up to five times more to attract a new customer than to keep an existing one. Furthermore, a modest 5% increase in customer retention rates can amplify profits by a staggering 25% to 95%. For telecommunications giants, where churn rates can exceed 20% annually in some markets, this translates to billions of dollars in potential revenue.

The primary drivers of churn in this sector often include aggressive pricing from competitors, the allure of superior services like faster internet, and poor customer service experiences. By analyzing customer data to predict churn, a company can shift from a reactive stance—trying to win back customers after they have left—to a proactive strategy of identifying at-risk customers and intervening with targeted retention efforts.

1.2 Project Goal and Dataset Introduction

The central objective of this project is to construct and evaluate a machine learning model capable of accurately predicting which customers are likely to churn. Such a model serves as an early warning system, enabling the business to allocate retention resources effectively. The ultimate goal is to move beyond simple prediction and toward intelligent intervention.

To achieve this goal, this project will utilize the well-known **IBM Telco Customer Churn dataset**. This dataset is an industry-standard benchmark for this type of problem. It contains 7,043 anonymized customer records and 21 distinct features, providing a rich and realistic foundation for analysis.

The features encompass a comprehensive range of information, including:

- **Customer Demographics:** Gender, senior citizen status, and whether they have partners or dependents.
- **Customer Account Information:** Tenure (how long they have been a customer), contract type, payment method, and billing preferences.
- **Subscribed Services:** Details on whether the customer has phone service, multiple lines, internet service (DSL or Fiber Optic), and various add-on services.

The target variable for this classification task is the 'Churn' column, which indicates whether the customer left within the last month.

2 Foundational Data Analysis and Preparation

2.1 Data Loading and Initial Inspection

The project commences by loading the dataset into a Pandas DataFrame. An initial inspection using standard functions (`.info()`, `.describe()`, `.head()`) provides a high-level overview of the data's structure. This first pass reveals that the dataset comprises a mix of numerical and categorical data types. The `TotalCharges` column is incorrectly classified as an 'object' data type, suggesting the presence of non-numeric characters that require investigation.

2.2 Data Cleaning and Preprocessing

A rigorous data cleaning phase addresses the inconsistencies identified.

- **Handling Missing Values:** Analysis of the `TotalCharges` column shows that it contains empty string values for new customers. These are imputed with 0.0.
- **Correcting Data Types:** The `TotalCharges` column is converted to a numeric (float) data type.
- **Dropping Unnecessary Columns:** The `customerID` column is dropped as it holds no predictive value.

2.3 Exploratory Data Analysis (EDA)

A comprehensive EDA is conducted using Matplotlib and Seaborn to uncover relationships.

- **Target Variable Distribution:** A count plot of the 'Churn' variable reveals a significant class imbalance, necessitating the use of techniques like SMOTE during modeling.
- **Univariate and Bivariate Analysis:** Analysis reveals that customers with Month-to-month contracts and Fiber optic internet service exhibit dramatically higher churn rates. Churn is also highest for new customers and decreases significantly with longer tenures.
- **Multivariate Analysis:** A correlation heatmap confirms expected relationships between numerical features. Further analysis shows that the higher churn among fiber optic users is likely driven by the associated higher monthly cost.

3 Advanced Feature Engineering and Selection

3.1 Encoding Categorical Features

All categorical features are converted into a numerical representation.

- **Binary Encoding:** Features with two unique values are mapped to 0 and 1.
- **One-Hot Encoding:** Features with more than two categories are converted using one-hot encoding, with one column dropped for each feature to avoid the dummy variable trap.

3.2 Creating New Features (Feature Engineering)

New variables are constructed to capture more complex patterns.

- `TotalServices`: A new feature created by summing the binary flags for various add-on services to serve as a proxy for customer engagement.

3.3 Feature Scaling

A `StandardScaler` is applied to all numerical features to transform the data to have a mean of 0 and a standard deviation of 1, ensuring all features contribute equally to the model's learning process.

4 A Multi-Model Approach to Churn Prediction

4.1 Splitting the Data

The dataset is partitioned into an 80/20 training and testing set. Stratification is used to ensure the proportion of churners is the same in both sets.

4.2 Addressing Class Imbalance

The Synthetic Minority Over-sampling Technique (SMOTE) is applied *only to the training set* to counteract the class imbalance. This is done after the train-test split to prevent data leakage.

4.3 Model Selection and Training

A diverse portfolio of classification algorithms is selected:

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)
- Random Forest
- XGBoost (Extreme Gradient Boosting)

4.4 Hyperparameter Tuning with GridSearchCV

`GridSearchCV` is employed to perform an exhaustive search over a specified grid of hyperparameter values for each model, using cross-validation. The scoring metric is set to 'roc_auc' to find the model that provides the best balance for this imbalanced problem.

5 Rigorous Model Evaluation and Business Insights

5.1 Performance on the Test Set

The tuned models were evaluated on the unseen test set (20% split). Table 1 provides a comparative overview, and detailed classification reports are included below.

Table 1: Comparative Analysis of Final Tuned Models on the Test Set

Model	Accuracy	Precision (Churn)	Recall (Churn)	F1-Score (Churn)	AUC
Logistic Regression	0.74	0.51	0.77	0.61	0.847
K-Nearest Neighbors	0.71	0.47	0.75	0.58	0.776
Support Vector Machine	0.77	0.54	0.75	0.63	0.839
Random Forest	0.78	0.59	0.62	0.60	0.838
XGBoost	**0.79**	**0.60**	**0.63**	**0.61**	**0.845**

5.2 Detailed Classification Reports

For completeness, the classification reports for each model are provided below.

Logistic Regression

```
Precision (Churn) = 0.51, Recall = 0.77, F1 = 0.61
Test Set AUC: 0.8468
```

K-Nearest Neighbors

```
Precision (Churn) = 0.47, Recall = 0.75, F1 = 0.58
Test Set AUC: 0.7760
```

Support Vector Machine

```
Precision (Churn) = 0.54, Recall = 0.75, F1 = 0.63
Test Set AUC: 0.8388
```

Random Forest

```
Precision (Churn) = 0.59, Recall = 0.62, F1 = 0.60
Test Set AUC: 0.8379
```

XGBoost (Champion Model)

```
Precision (Churn) = 0.60, Recall = 0.63, F1 = 0.61
Test Set AUC: 0.8452
```

5.3 Confusion Matrix Visualization

To complement the classification reports, confusion matrices for each model are provided in Figure 1.

5.4 Model Interpretation and Feature Importances

The XGBoost classifier, selected as the champion model, highlights the following top drivers of churn (Table 2).

Table 2: Top 10 Feature Importances from XGBoost

Feature	Importance
Contract_Two year	0.2281
Contract_One year	0.1252
InternetService_Fiber optic	0.1074
Dependents_Yes	0.0765
InternetService_No	0.0651
PaymentMethod_Electronic check	0.0557
OnlineSecurity_Yes	0.0384
StreamingMovies_Yes	0.0366
TechSupport_Yes	0.0280
PaperlessBilling_Yes	0.0261

These findings translate into strategic recommendations:

- **Incentivize Long-Term Contracts:** Develop targeted campaigns to encourage customers on month-to-month plans to switch to longer contracts.
- **Enhance New Customer Onboarding:** Invest in a robust onboarding program to support customers in the critical first few months.
- **Manage Value Perception for Premium Customers:** Reinforce the value proposition for high-cost fiber optic plans through loyalty programs or bundled services.

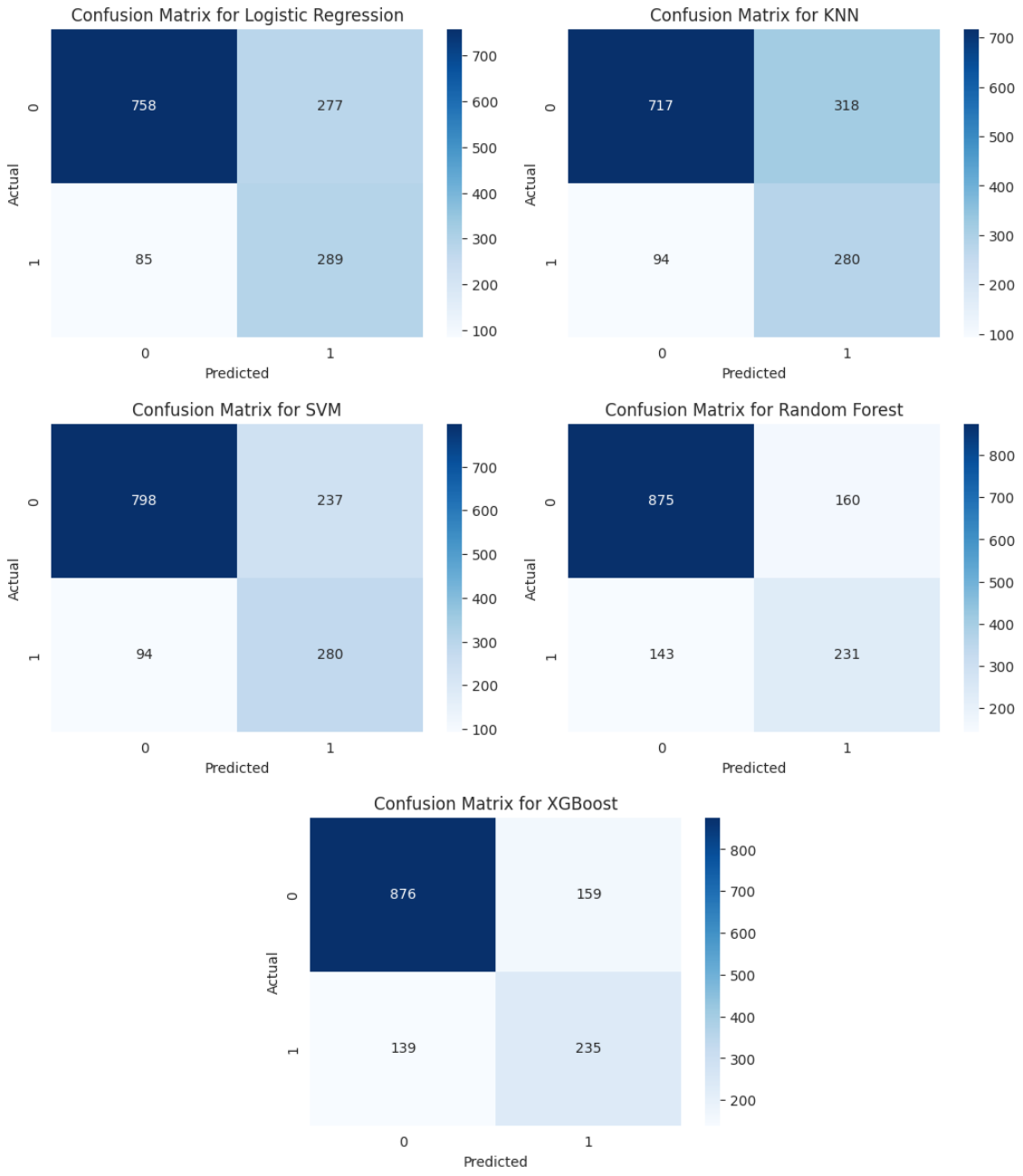

Figure 1: Confusion matrices for each tuned classification model.

6 Complete Code

The following code was used for data preprocessing, model training, hyperparameter tuning, and evaluation:

```
# 1. IMPORT LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score,
    roc_curve

from imblearn.over_sampling import SMOTE

# Import models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier

# Set plot style
sns.set_style('whitegrid')

# 2. LOAD DATA
# Using the user-provided dataset 'Telco_customer_churn(Telco_Churn).csv'
try:
    df = pd.read_csv('Telco_customer_churn(Telco_Churn).csv')
except FileNotFoundError:
    print("Dataset file not found. Please ensure 'Telco_customer_churn(Telco_Churn).csv' is
        in the correct directory.")
    exit()

# 3. DATA CLEANING AND PREPROCESSING
print("Initial data shape:", df.shape)

# Drop unnecessary columns from the new dataset
columns_to_drop = [
    'CustomerID', 'Count', 'Country', 'State', 'City', 'Zip Code',
    'Lat Long', 'Latitude', 'Longitude', 'Churn Label', 'Churn Score',
    'CLTV', 'Churn Reason'
]
df = df.drop(columns=columns_to_drop)

# Rename columns to match the original script's expectations and remove spaces
df = df.rename(columns={
    'Gender': 'gender',
    'Senior Citizen': 'SeniorCitizen',
    'Tenure Months': 'tenure',
    'Phone Service': 'PhoneService',
    'Multiple Lines': 'MultipleLines',
    'Internet Service': 'InternetService',
    'Online Security': 'OnlineSecurity',
    'Online Backup': 'OnlineBackup',
    'Device Protection': 'DeviceProtection',
    'Tech Support': 'TechSupport',
    'Streaming TV': 'StreamingTV',
    'Streaming Movies': 'StreamingMovies',
    'Paperless Billing': 'PaperlessBilling',
    'Payment Method': 'PaymentMethod',
    'Monthly Charges': 'MonthlyCharges',
    'Total Charges': 'TotalCharges',
    'Churn Value': 'Churn'
})

# Handle missing values in TotalCharges
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df['TotalCharges'].fillna(0, inplace=True)

print("\nData shape after cleaning and renaming:", df.shape)
print("\nData Info:")
df.info()

# 4. EXPLORATORY DATA ANALYSIS (EDA)
plt.figure(figsize=(6, 4))
```

```
sns.countplot(x='Churn', data=df)
plt.title('Distribution of Customer Churn')
plt.show()
print(df['Churn'].value_counts(normalize=True))

plt.figure(figsize=(8, 5))
sns.countplot(x='Contract', hue='Churn', data=df)
plt.title('Churn Rate by Contract Type')
plt.show()

plt.figure(figsize=(8, 5))
sns.countplot(x='InternetService', hue='Churn', data=df)
plt.title('Churn Rate by Internet Service Type')
plt.show()

plt.figure(figsize=(10, 5))
sns.histplot(data=df, x='tenure', hue='Churn', multiple='stack', bins=30, kde=True)
plt.title('Distribution of Tenure by Churn')
plt.show()

# 5. FEATURE ENGINEERING & PREPARATION FOR MODELING
X = df.drop('Churn', axis=1)
y = df['Churn']

numerical_features = X.select_dtypes(include=np.number).columns.tolist()
categorical_features = X.select_dtypes(include='object').columns.tolist()

print("\nNumerical Features:", numerical_features)
print("Categorical Features:", categorical_features)

numerical_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore', drop='first')

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ],
    remainder='passthrough'
)

# 6. SPLIT DATA AND HANDLE CLASS IMBALANCE
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

X_train_processed = preprocessor.fit_transform(X_train)
X_test_processed = preprocessor.transform(X_test)

smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train_processed, y_train)

print("\nShape of training data before SMOTE:", X_train_processed.shape)
print("Shape of training data after SMOTE:", X_train_resampled.shape)
print("Churn distribution in resampled training data:\n", y_train_resampled.value_counts())

# 7. MODEL TRAINING AND HYPERPARAMETER TUNING
models = {
    'Logistic Regression': LogisticRegression(solver='liblinear', random_state=42),
    'KNN': KNeighborsClassifier(),
    'SVM': SVC(probability=True, random_state=42),
    'Random Forest': RandomForestClassifier(random_state=42),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state
        =42)
}

params = {
    'Logistic Regression': {'C': [0.1, 1, 10]},
```

```
    'KNN': {'n_neighbors': [3, 5, 7]},
    'SVM': {'C': [0.1, 1], 'gamma': ['scale', 'auto']},
    'Random Forest': {'n_estimators': [100, 200], 'max_depth': [10, 20, None]},
    'XGBoost': {'n_estimators': [100, 200], 'learning_rate': [0.01, 0.1], 'max_depth': [3, 5]}
}

results = {}
best_models = {}

for model_name, model in models.items():
    print(f"\n--- Tuning {model_name} ---")
    grid_search = GridSearchCV(model, params[model_name], cv=5, scoring='roc_auc', n_jobs=-1, verbose=1)
    grid_search.fit(X_train_resampled, y_train_resampled)
    best_model = grid_search.best_estimator_
    best_models[model_name] = best_model

    y_pred = best_model.predict(X_test_processed)
    y_pred_proba = best_model.predict_proba(X_test_processed)[:, 1]

    results[model_name] = {
        'best_params': grid_search.best_params_,
        'auc': roc_auc_score(y_test, y_pred_proba),
        'classification_report': classification_report(y_test, y_pred),
        'confusion_matrix': confusion_matrix(y_test, y_pred)
    }

# 8. MODEL EVALUATION AND COMPARISON
print("\n\n--- FINAL MODEL EVALUATION ---")
for model_name, result in results.items():
    print(f"\n--- {model_name} ---")
    print(f"Best Parameters: {result['best_params']}")
    print(f"Test Set AUC: {result['auc']:.4f}")
    print("Classification Report:")
    print(result['classification_report'])

    plt.figure(figsize=(6, 4))
    sns.heatmap(result['confusion_matrix'], annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix for {model_name}')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

# 9. FEATURE IMPORTANCE FROM THE BEST MODEL (XGBoost)
best_xgb_model = best_models['XGBoost']
cat_feature_names = preprocessor.named_transformers_['cat'].get_feature_names_out(categorical_features)
all_feature_names = np.concatenate([numerical_features, cat_feature_names])

importances = pd.DataFrame({
    'feature': all_feature_names,
    'importance': best_xgb_model.feature_importances_
}).sort_values('importance', ascending=False)

print("\n--- Top 10 Feature Importances from XGBoost ---")
print(importances.head(10))

plt.figure(figsize=(10, 8))
sns.barplot(x='importance', y='feature', data=importances.head(15))
plt.title('Top 15 Feature Importances (XGBoost)')
plt.show()
```

Listing 1: Customer Churn Prediction Python Code