CUSTOMER CHURN ANALYSIS, A PREDICTIVE MODEL

ADIGRA EMMANUEL CYRILLE AKA
BUSINESS AND DATA ANALYST
GRADUATE STUDENT AT SOUTHERN UTAH UNIVERSITY
Cedar City, UT, 84720, United States
cyrilleemmanuelaka@gmail.com

August 9, 2025

ABSTRACT

This document presents a comprehensive analysis of customer churn in the telecommunications industry. The project combines data science techniques using Python with business intelligence visualization in Power BI to identify key factors driving customer attrition. We developed predictive models that achieved accuracy 80% and created an interactive dashboard to help business stakeholders make data-driven retention decisions. The data set here that we have obtained from Kaggle is a sample of all customers registered by Telco, a telecommunication company. Telco wants to know better the customer and the rate at which they can experience churn.

1 INTRODUCTION–

Customer churn, the phenomenon in which customers discontinue their services, is a critical metric in the telecommunications industry. Reducing churn directly impacts revenue and profitability. This project analyzes a dataset of 7,043 customers from a telecommunication company to understand churn patterns and predict at-risk customers.

1.1 Objectives

- Identify key factors contributing to customer churn
- Develop predictive models to flag at-risk customers
- Create an interactive dashboard for business decision-making
- Propose data-driven retention strategies

2 Data Preparation

2.1 Dataset Overview

The dataset contains customer information including:

- Demographic data (gender, senior citizen status)
- Account information (tenure, contract type)
- Service details (internet service, add-ons)
- Charges (monthly, total)
- Churn status (target variable)

2.2 Data Cleaning

For this project, we needed analytical libraries such as pandas, numpy, matplotlib, seaborn, and Scikit-learn. Then we performed the following cleaning steps in Python:

```python
# Load the dataset
df = pd.read_csv('WA_Fn-UseC_-Telco-Customer-Churn.csv')

# Display first few rows to understand the data
print(df.head())

# Check basic info about the dataset
print("\nDataset Info:")
print(df.info())

# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Handle missing values (there's one in TotalCharges)
# Convert TotalCharges to numeric, coerce errors to NaN
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')

# Fill missing TotalCharges with 0 (likely new customers)
df['TotalCharges'].fillna(0, inplace=True)

# Check unique values in each column to understand categorical variables
print("\nUnique Values:")
for column in df.columns:
    if df[column].dtype == 'object':
        print(f"{column}: {df[column].unique()}")

```

Figure 1: Data Cleaning and Preparation Python Code

2.3 Feature Engineering

Key engineered features included:

Table 1: Engineered Features

Feature	Description
tenure_group	Customer tenure categorized in 12-month intervals
service_bundle	Combination of internet service and add-ons
charge_ratio	Monthly charges as percentage of total charges

3 Exploratory Data Analysis

3.1 Churn Distribution

First, we performed the following Python code to make everything clear:

```python
# Set style for plots
sns.set(style="whitegrid")

# 1. Churn distribution
plt.figure(figsize=(6, 4))
sns.countplot(x='Churn', data=df)
plt.title('Customer Churn Distribution')
plt.show()

# Percentage of churn
churn_percentage = df['Churn'].value_counts(normalize=True) * 100
print("\nChurn Percentage:")
print(churn_percentage)

# 2. Numeric features analysis
numeric_features = ['tenure', 'MonthlyCharges', 'TotalCharges']
plt.figure(figsize=(12, 4))
for i, feature in enumerate(numeric_features, 1):
    plt.subplot(1, 3, i)
    sns.boxplot(x='Churn', y=feature, data=df)
    plt.title(f'{feature} vs Churn')
plt.tight_layout()
plt.show()

# 3. Categorical features analysis
categorical_features = ['gender', 'SeniorCitizen', 'Partner', 'Dependents',
                        'PhoneService', 'MultipleLines', 'InternetService',
                        'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
                        'TechSupport', 'StreamingTV', 'StreamingMovies',
                        'Contract', 'PaperlessBilling', 'PaymentMethod']

plt.figure(figsize=(20, 30))
for i, feature in enumerate(categorical_features, 1):
    plt.subplot(6, 3, i)
    sns.countplot(x=feature, hue='Churn', data=df)
    plt.title(f'{feature} vs Churn')
    plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# 4. Correlation matrix for numeric features
plt.figure(figsize=(8, 6))
corr_matrix = df[numeric_features].corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

Figure 2: Exploratory Data Analysis Python Code

Then we saw that the churn rate was 26.5%, with significant variation between the customer segments:

3.2 Key Findings

- Month-to-month contracts had 43% churn vs 12% for 2-year contracts
- Fiber optic internet users churned at 41% vs 19% for DSL
- Customers without online security had 2.5× higher churn rate

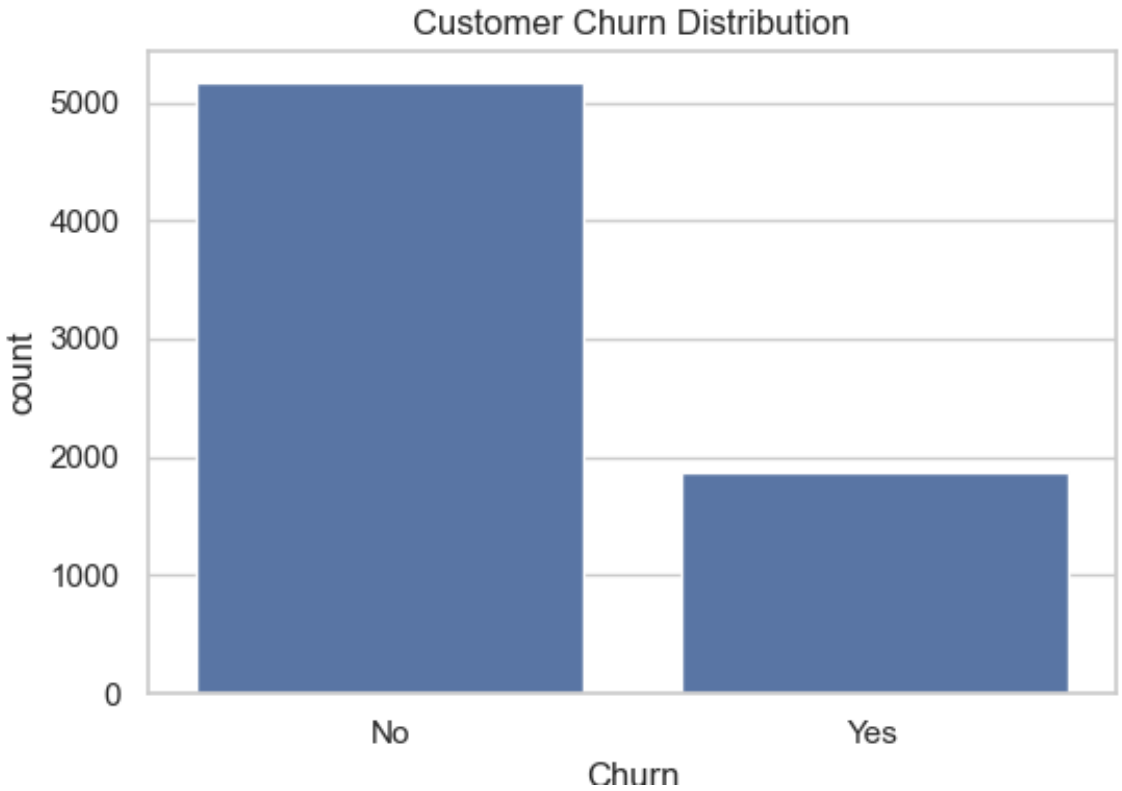

Figure 3: Distribution of Churned vs Retained Customers

4 Predictive Modeling

Before modeling, we need to prepare the data by encoding categorical variables and scaling numeric. Here is the Python code:

```python
# Drop customerID as it's not useful for prediction
df.drop('customerID', axis=1, inplace=True)

# Convert Churn to binary (1 for Yes, 0 for No)
df['Churn'] = df['Churn'].apply(lambda x: 1 if x == 'Yes' else 0)

# Encode categorical variables
categorical_cols = [col for col in df.columns if df[col].dtype == 'object']
df_encoded = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

# Split data into features (X) and target (y)
X = df_encoded.drop('Churn', axis=1)
y = df_encoded['Churn']

# Split into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Scale numeric features
scaler = StandardScaler()
numeric_cols = ['tenure', 'MonthlyCharges', 'TotalCharges']
X_train[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
X_test[numeric_cols] = scaler.transform(X_test[numeric_cols])

print("\nData shapes:")
print(f"Training set: {X_train.shape}, {y_train.shape}")
print(f"Test set: {X_test.shape}, {y_test.shape}")
```

Figure 4: Data Preprocessing for Modeling

4.1 Model Selection

We evaluated two classification algorithms:

Table 2: Model Performance Comparison

Model	Accuracy	Precision	Recall	F1-Score
Logistic Regression	0.80	0.68	0.53	0.60
Random Forest	0.79	0.67	0.55	0.60

The logistic regression model can be expressed as:

$$\log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_n X_n \tag{1}$$

```python
# 1. Logistic Regression
logreg = LogisticRegression(max_iter=1000)
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)

print("\nLogistic Regression Results:")
print(classification_report(y_test, y_pred_logreg))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_logreg))
print(f"Accuracy: {accuracy_score(y_test, y_pred_logreg):.2f}")

# 2. Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)

print("\nRandom Forest Results:")
print(classification_report(y_test, y_pred_rf))
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_rf))
print(f"Accuracy: {accuracy_score(y_test, y_pred_rf):.2f}")
```

Figure 5: Logistic Regression and Random Forest Python Code

4.2 Feature Importance

The Random Forest model identified these top predictive features:

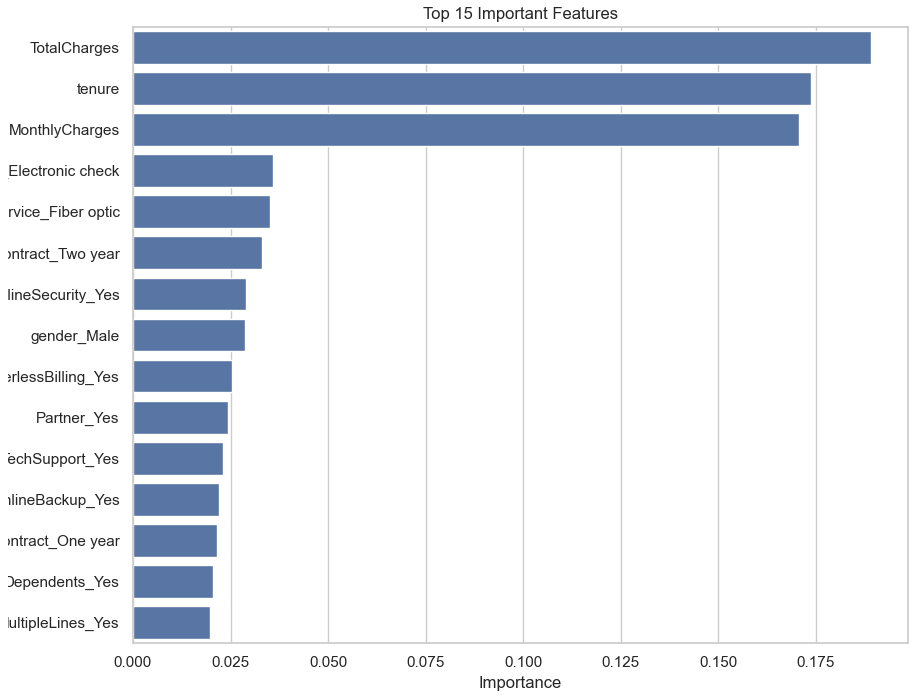

Figure 6: Top 15 Features by Importance

5 Power BI Dashboard

5.1 Dashboard Architecture

The interactive dashboard includes:

- Overview metrics (churn rate, customer counts)
- Demographic analysis (age, gender, tenure)
- Service usage patterns
- Contract and payment analysis
- Predictive insights

5.2 Advanced Features

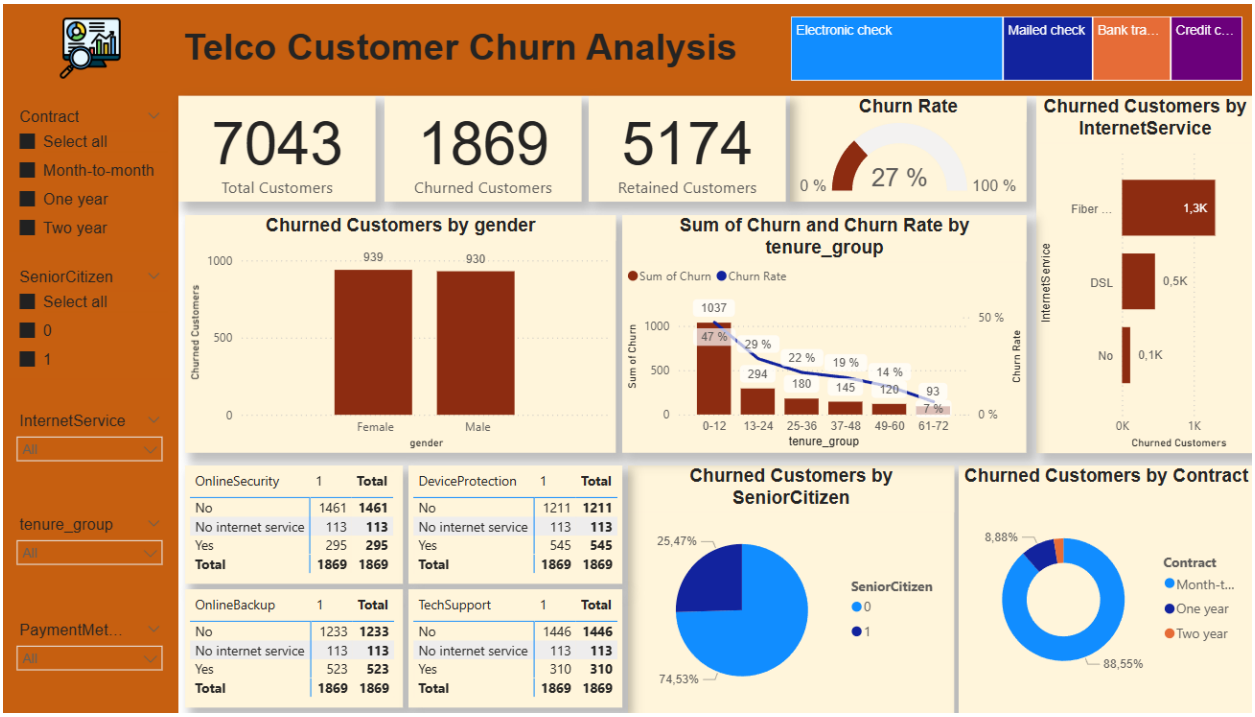

Figure 7: Power BI Dashboard Overview

6 Business Recommendations

Based on our analysis, we recommend:

6.1 Retention Strategies

- Target month-to-month contract customers with incentives to switch to longer contracts
- Bundle online security with internet service
- Develop personalized offers for high-tenure customers

6.2 Potential Impact

Table 3: Estimated Impact of Recommendations

Strategy	Potential Churn Reduction
Contract incentives	15-20%
Service bundling	10-15%
Personalized offers	5-10%

7 FIGURES–

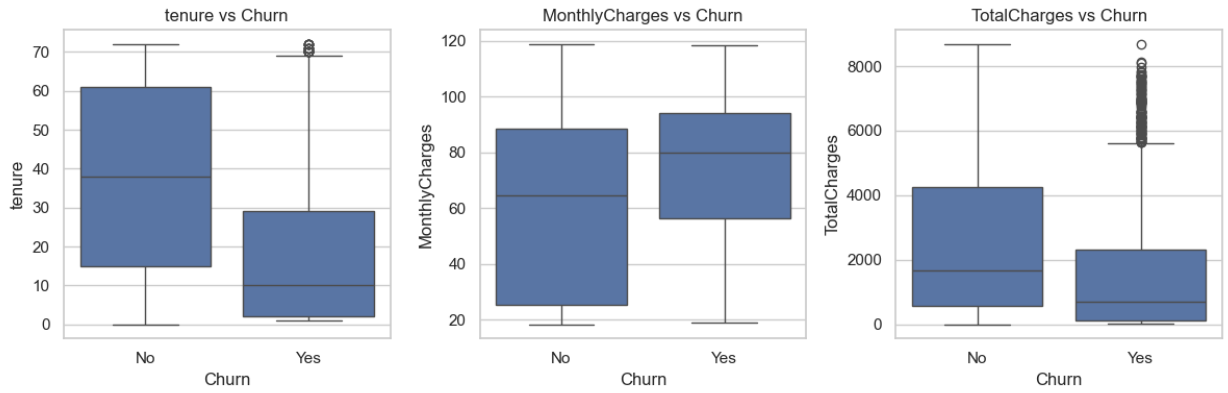

Figure 8: Box and Whisker

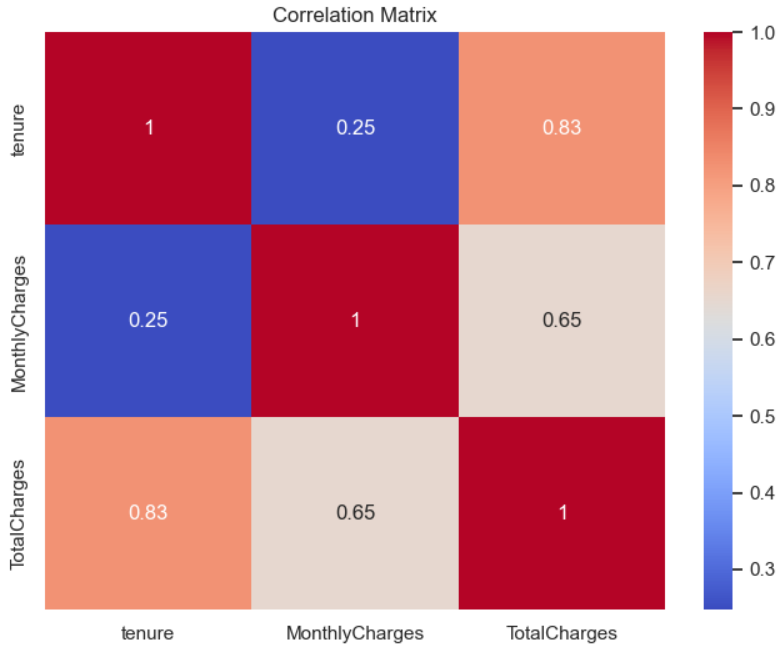

Figure 9: Correlation Matrix

8 CONCLUSIONS–

This project demonstrated how data science and business intelligence can combine to address customer churn. The predictive models achieved 80% accuracy in identifying at-risk customers, while the Power BI dashboard enabled actionable insights for business stakeholders. Future work could incorporate real-time data pipelines and A/B testing of retention strategies.

9 ACKNOWLEDGEMENT–

I would like to acknowledge the open-source community for providing the tools and libraries that made this project possible, including Python's data science ecosystem and Microsoft Power BI.

References

[1] George Mount: Advancing into Analytics, from Excel to Python and R

[2] Sharmila K. Wagh - Aishwarya A. Andhale b - Kishor S. Wagh c - Jayshree R. Pansare a - Sarita P. Ambadekar d - S.H. Gawande e: Customer churn prediction in telecom sector using machine learning techniques

[3] https://www.kaggle.com/datasets/blastchar/telco-customer-churn

[4] https://www.lightico.com/blog/complete-guide-to-reduce-churn-in-telecom/