# **Telco Churn Prediction Project Report**

## Introduction

The goal of this project is to build a predictive model to identify customers likely to churn for a telecommunications company. The dataset used contains various customer attributes and their churn status.

The dataset contains various types of information, including:

* **Churn Information:** Indicates whether a customer has left within the last month.
* **Service Subscriptions:** Details on services each customer has signed up for, such as phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies.
* **Account Information:** Includes data on the customer's tenure, contract type, payment method, paperless billing status, monthly charges, and total charges.
* **Demographics:** Provides demographic details like gender, age range, and whether the customer has partners and dependents.

## Data Preprocessing and EDA

**Data Loading and Initial Inspection:**

import pandas as pd

import numpy as np

data = pd.read\_csv('WA\_Fn-UseC\_-Telco-Customer-Churn.csv')

data.head()

data.info()

The dataset is loaded, and initial inspection shows a mix of numerical and categorical features.

**Handling Missing Values:**

data = data.replace(' ', np.nan)

data['TotalCharges'] = pd.to\_numeric(data['TotalCharges'], errors='coerce')

data = data.dropna()

Missing values in the TotalCharges column are replaced with NaN and then dropped.

**Dropping Irrelevant Columns:**

data = data.drop(columns=['customerID'])

The customerID column is dropped as it is not relevant for prediction.

## Feature Engineering

**Label Encoding and Scaling:**

from sklearn.preprocessing import LabelEncoder, StandardScaler

le = LabelEncoder()

for column in data.select\_dtypes(include=['object']).columns:

data[column] = le.fit\_transform(data[column])

scaler = StandardScaler()

numerical\_columns = data.select\_dtypes(include=['int64','float64']).columns.drop('SeniorCitizen')

data[numerical\_columns] = scaler.fit\_transform(data[numerical\_columns])

Categorical features are label-encoded and numerical features are standardized.

Standard Scaler = (x – mean) / standard deviation

**Creating New Features:**

data['TotalCharges\_per\_tenure'] = data['TotalCharges'] / data['tenure']

A new feature TotalCharges\_per\_tenure is created to capture the average monthly charge.

## Exploratory Data Analysis (EDA)

**Correlation Matrix:**

import matplotlib.pyplot as plt

import seaborn as sns

corr\_matrix = data.corr()

plt.figure(figsize=(12,8))

sns.heatmap(corr\_matrix, annot=True, fmt='.2f')

plt.title('Correlation Matrix')

plt.show()

The correlation matrix helps understand the relationships between different features.

**Distribution of Churn:**

plt.figure(figsize=(6,4))

sns.countplot(data['Churn'])

plt.title('Distribution of Churn')

plt.show()

The churn distribution is visualized, showing the imbalance in the target variable.

**Distribution of Tenure:**

plt.figure(figsize=(6,4))

sns.histplot(data['tenure'], kde=True)

plt.title('Distribution of Tenure')

plt.show()

The tenure distribution provides insights into customer longevity.

**Monthly Charges by Churn:**

plt.figure(figsize=(8,6))

sns.boxplot(x='Churn', y='MonthlyCharges', data=data)

plt.title('Monthly Charges by Churn')

plt.show()

This visualization shows how monthly charges vary with churn status.

## Model Building and Evaluation

**Data Splitting:**

from sklearn.model\_selection import train\_test\_split

x = data.drop(columns=['Churn'])

y = data['Churn']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

The data is split into training and test sets.

**Model Training:**

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

log\_reg = LogisticRegression()

rf\_clf = RandomForestClassifier()

log\_reg.fit(x\_train, y\_train)

rf\_clf.fit(x\_train, y\_train)

Logistic Regression and Random Forest models are trained.

**Model Evaluation:**

from sklearn.metrics import accuracy\_score, classification\_report

y\_pred\_log\_reg = log\_reg.predict(x\_test)

y\_pred\_rf = rf\_clf.predict(x\_test)

print("Logistic Regression Accuracy:", accuracy\_score(y\_test, y\_pred\_log\_reg))

print("Random Forest Accuracy:", accuracy\_score(y\_test, y\_pred\_rf))

print("Logistic Regression Report:\n", classification\_report(y\_test, y\_pred\_log\_reg))

print("Random Forest Report:\n", classification\_report(y\_test, y\_pred\_rf))

Both models are evaluated using accuracy and classification report metrics.

A screenshot of a computer

Description automatically generated

## Challenges

**Data Imbalance:** The churn data is imbalanced, affecting model performance.

**Feature Selection:** Selecting the most relevant features was challenging due to the high dimensionality.

## Conclusion

This project involved building a churn prediction model using logistic regression and random forest classifiers. The EDA provided insights into feature relationships and customer behavior. Although this is an initial models showed reasonable performance and addressing data imbalance.

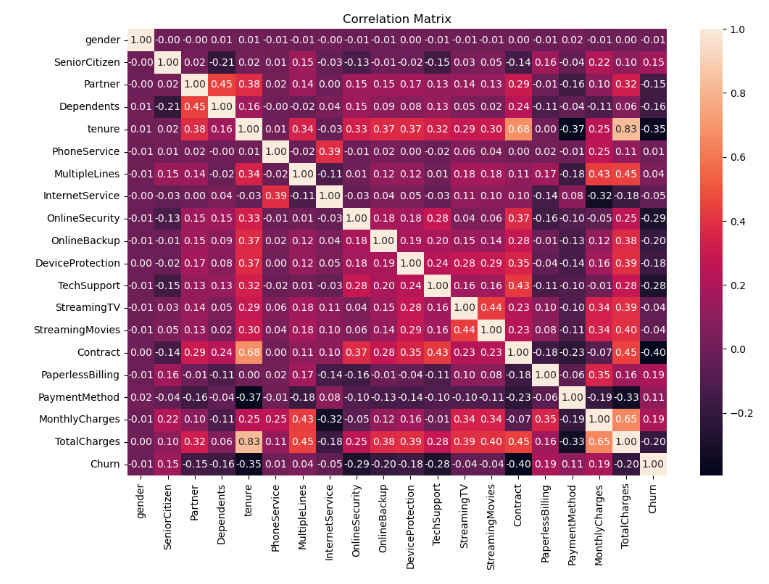
## Appendices

**Code:** All code used in this project is documented within the provided Jupyter notebook.

**Data:** The dataset is available in the accompanying file WA\_Fn-UseC\_-Telco-Customer-Churn.csv.

## Visualization Examples

**Correlation Matrix**



We can clearly see that TotalCharges and tenure are highly positive correlated, it means that if tenure is increased than TotalCharges is also increasing.

**Churn Distribution**

**A blue and orange rectangular bar graph

Description automatically generated**

**Tenure Distribution**

**A graph of a number of columns

Description automatically generated with medium confidence**

**Monthly Charges by Churn**

**A blue and orange rectangular boxes

Description automatically generated**