# OPTIMIZING BUSINESS PERFORMANCE THROUGH CUSTOMER CHURN ANALYSIS IN SUBSCRIPTION BASED SERVICES

**SUBMITTED BY :ARUNIMA K**

**SUBMISSION DATE : 25-09-2024**

# AIM

- The primary objective is to analyze customer churn in a subscription-based service by identifying patterns and trends from historical data.
- Here focuses on uncovering key factors and behaviors that contribute to customer churn and providing deeper insights into customer dynamics.
- Data preprocessing and visualization techniques are applied to ensure accurate analysis and clear interpretation of the data.
- By analyzing these patterns, actionable insights are generated to help the business understand customer needs and improve retention strategies.
- The overall aim is to support better decision-making for customer relationship management, reducing churn and enhancing customer satisfaction.

# BUSINESS PROBLEM

- The company faces a high customer churn rate, which significantly impacts revenue and growth potential.
- There is a lack of clear insight into why customers leave, making it difficult to develop effective customer retention strategies.
- The key drivers of customer churn remain unidentified, preventing the business from taking proactive steps to address customer dissatisfaction.
- Marketing and customer support resources may be inefficiently allocated with no clear focus on at-risk customer segments.
- This leads to missed opportunities to enhance the customer experience, improve product offerings and tailor services to better meet customer needs.

# DESCRIPTION

.

- This analysis focuses on customer churn in a subscription-based service, aiming to identify patterns and key factors that influence customer attrition.
- This encompasses a comprehensive approach to data understanding, preprocessing, and visualization, emphasizing actionable insights.
- The intention is to enhance customer retention strategies by uncovering churn trends and gaining a deeper understanding of customer behavior.
- The process involves data cleaning, exploratory data analysis (EDA), and visual representations utilizing technologies such as Pandas, Numpy, Matplotlib, and Seaborn.
- Important aspects include identifying high-risk customer segments, examining the drivers of churn, and generating insights to support informed decision-making and improve business strategies.

# FUNCTIONALITIES

## 1.DATA LOADING AND HANDLING

- Here involves reading the customer churn dataset from a specified file path to ensure accessibility for analysis.
- It inspects the data's shape and summary statistics to provide an overview of its structure and key characteristics.

## 2.DATA PREPROCESSING

- This process focuses on cleaning the dataset by handling missing values, converting data types, and identifying duplicates to ensure data quality and consistency.
- It also involves removing unwanted columns, thereby streamlining the dataset and preparing it for more effective analysis and visualization.

## 3.UNIVARIATE ANALYSIS

- This stage involves visualizing the distribution of the Churn feature using a pie chart to understand customer retention rates.
- Histograms are employed to analyze the frequency of Monthly Charges and Total Charges, while a count plot illustrates the distribution of Contract Types, helping to uncover trends in customer preferences

## 4.BIVARIATE ANALYSIS

- Here boxplots are employed to explore the relationship between Monthly Charges and Churn rates, as well as between Tenure and Monthly Charges, offering insights into how these factors influence customer retention.
- A scatter plot of Tenure against Monthly Charges reveals the correlation between these two variables, aiding in understanding customer behavior.

## 5.MULTIVARIATE ANALYSIS

- Here the pairplot is utilized to visualize the relationships between multiple features simultaneously, enabling the identification of patterns and correlations among variables.
- The heatmap is employed to illustrate the correlation matrix, highlighting the strength and direction of relationships between various factors impacting customer behavior.

# ERROR HANDLING AND EXCEPTION HANDLING

## 1.FileNotFoundError Handling during Data Loading

- A check is implemented while loading the dataset to handle cases where the file might not be found, ensuring that the program does not crash if the file is missing and providing a meaningful error message to guide the user.

```
try:
    self.data = pd.read_csv(self.file_path)
    return self.data
except FileNotFoundError:
    raise FileNotFoundError(f"File at path {self.file_path} was not found.")
except Exception as e:
    raise Exception(f"An error occurred while loading the data: {str(e)}")
```

## 2. ValueError Handling for Data Inspection

- When displaying the dataset and its shape, a ValueError might occur if there is an issue with the dataset structure, preventing unexpected crashes and providing clarity on the error.

3. **KeyError Handling for Data Conversion**

- Here it is handled during the conversion of the Total Charges column from object to numeric type to address cases where the column is missing or misnamed, preventing errors and ensuring smooth conversion and data processing.

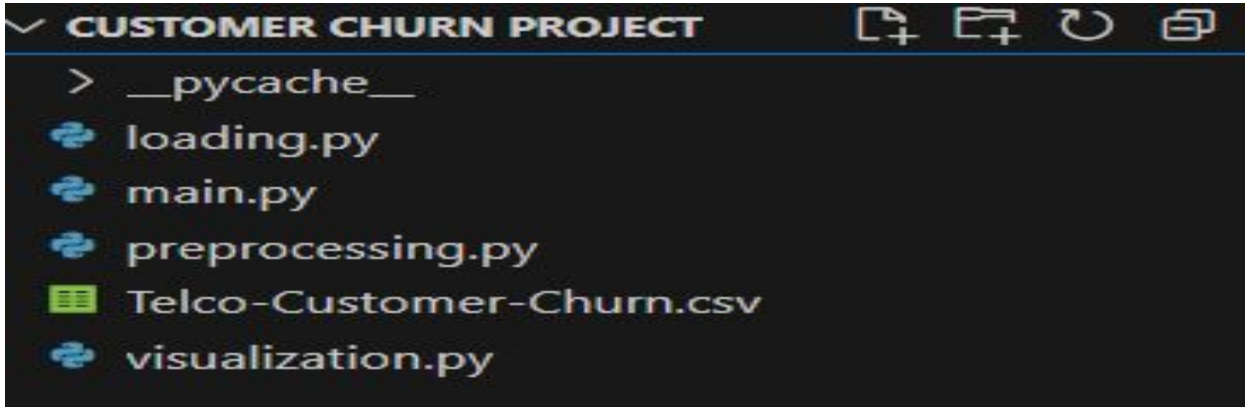
```
try:
    self.df['TotalCharges'] = pd.to_numeric(self.df['TotalCharges'], errors='coerce')
    self.df.loc[self.df['TotalCharges'].isnull(), 'TotalCharges'] = 0
    print("TotalCharges converted and missing values filled.")
except KeyError:
    print("Error: 'TotalCharges' column not found in the dataframe.")
except Exception as e:
    print(f"An error occurred while converting 'TotalCharges': {e}")
```

- Exception handling is implemented when dropping unwanted columns to ensure the program doesn't fail if a column is missing, allowing the project to continue running smoothly and making the code more robust.

```
try:

    self.df = self.df.drop(['PhoneService', 'MultipleLines', 'OnlineSecurity',
                            'OnlineBackup', 'DeviceProtection', 'TechSupport',
                            'StreamingTV', 'StreamingMovies'], axis=1)
    print("Unwanted columns dropped.")
except KeyError as e:
    print(f"Error: One or more columns to drop were not found: {e}")
```

# CODE

# INDEX


CUSTOMER CHURN PROJECT
__pycache__
loading.py
main.py
preprocessing.py
Telco-Customer-Churn.csv
visualization.py

# DATA LOADING

loading.py > ...

```python
import pandas as pd

class DataLoader:
    def __init__(self, file_path):
        self.file_path = file_path
        self.data = None

    def load_data(self):

        try:
            self.data = pd.read_csv(self.file_path)
            return self.data
        except FileNotFoundError:
            raise FileNotFoundError(f"File at path {self.file_path} was not found.")
        except Exception as e:
            raise Exception(f"An error occurred while loading the data: {str(e)}")

    def display_head(self, num_rows=5):
        if self.data is not None:
            return self.data.head(num_rows)
        else:
            raise ValueError("Data not loaded. Please call load_data() first.")

    def display_shape(self):
        if self.data is not None:
            return self.data.shape
        else:
            raise ValueError("Data not loaded. Please call load_data() first.")

    def display_summary(self):
        if self.data is not None:
            return self.data.describe(include='all')
        else:
            raise ValueError("Data not loaded. Please call load_data() first.")


```

```python

if __name__ == "__main__":
    try:
        loader = DataLoader(r'C:\Users\aruni\Desktop\customer churn project\Telco-Customer-Churn.csv')

        data = loader.load_data()

        print("First few rows of the dataset:")
        print(loader.display_head())

        print("Dataset shape:", loader.display_shape())

        print("Summary statistics:")
        print(loader.display_summary())


    except ValueError as ve:
        print(f"ValueError: {ve}")
    except FileNotFoundError as fnf_error:
        print(f"FileNotFoundError: {fnf_error}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
```

# DATA PREPROCESSING

```python
import pandas as pd
import numpy as np
from loading import DataLoader

class DataPreprocessor:
    def __init__(self, df):
        self.df = df

    def check_info(self):
        print(self.df.info())

    def check_missing_values(self):
        missing_values = self.df.isnull().sum()
        print("Missing values in each column:")
        print(missing_values)

    def convert_total_charges(self):

        try:
            self.df['TotalCharges'] = pd.to_numeric(self.df['TotalCharges'], errors='coerce')
            self.df.loc[self.df['TotalCharges'].isnull(), 'TotalCharges'] = 0
            print("TotalCharges converted and missing values filled.")
        except KeyError:
            print("Error: 'TotalCharges' column not found in the dataframe.")
        except Exception as e:
            print(f"An error occurred while converting 'TotalCharges': {e}")

    def find_duplicates(self):
        duplicate_count = self.df.duplicated().sum()
        print(f"Number of duplicate rows: {duplicate_count}")

    def drop_unwanted_columns(self):

        try:

            self.df = self.df.drop(['PhoneService', 'MultipleLines', 'OnlineSecurity',
                                    'OnlineBackup', 'DeviceProtection', 'TechSupport',
```

```
                                        'OnlineBackup', 'DeviceProtection', 'TechSupport',
                                        'StreamingTV', 'StreamingMovies'], axis=1)
            print("Unwanted columns dropped.")
        except KeyError as e:
            print(f"Error: One or more columns to drop were not found: {e}")

    def final_check(self):
        if self.df.isnull().sum().sum() == 0:
            print("No missing values remain in the dataset.")
        else:
            print("There are still missing values in the dataset.")

    def display_shape(self):
        print(f"Dataset shape: {self.df.shape}")


if __name__ == "__main__":

    data_loader = DataLoader(r'C:\Users\aruni\Desktop\customer churn project\Telco-Customer-Churn.csv')
    df = data_loader.load_data()

    preprocessor = DataPreprocessor(df)

    preprocessor.check_info()
    preprocessor.check_missing_values()
    preprocessor.convert_total_charges()
    preprocessor.find_duplicates()
    preprocessor.drop_unwanted_columns()
    preprocessor.display_shape()
    preprocessor.final_check()
```

# VISUALIZATION

```python
import matplotlib.pyplot as plt
import seaborn as sns

class DataVisualizer:
    def __init__(self, df):
        self.df = df

    def plot_churn_pie_chart(self):
        ax = self.df['Churn'].value_counts().plot(kind='pie', autopct='%1.1f%%', title='Pie Chart of Churn', figsize=(6, 6))
        ax.legend(['No', 'Yes'])
        plt.show()

    def plot_histograms(self):
        plt.figure(figsize=(12, 5))

        plt.subplot(1, 2, 1)
        plt.hist(self.df['MonthlyCharges'], bins=6, edgecolor='k')
        plt.title('Histogram of Monthly Charges')
        plt.xlabel('Monthly Charges')
        plt.ylabel('Frequency')

        plt.subplot(1, 2, 2)
        plt.hist(self.df['TotalCharges'], bins=6, edgecolor='k')
        plt.title('Histogram of Total Charges')
        plt.xlabel('Total Charges')
        plt.ylabel('Frequency')

        plt.tight_layout()
        plt.show()

    def plot_contract_churn(self):
        sns.countplot(data=self.df, x='Contract', hue='Churn')
        plt.title('Distribution of Contract Type by Churn')
        plt.show()

    def plot_boxplot_churn_vs_tenure(self):
        sns.boxplot(x='Churn', y='tenure', data=self.df)
```

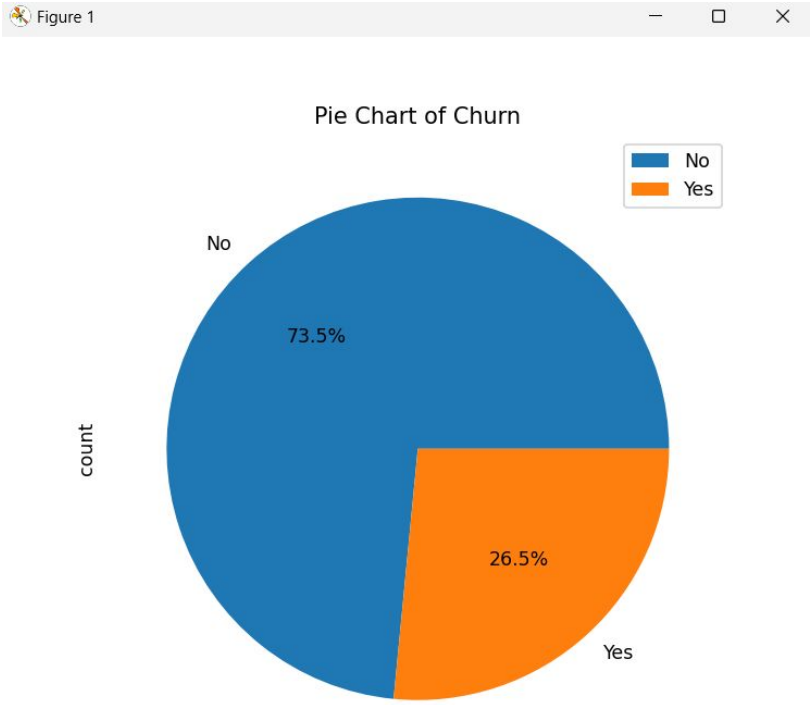
```
        plt.title('Churn vs. Tenure')
        plt.show()

    def plot_boxplot_churn_vs_monthly_charges(self):
        sns.boxplot(x='Churn', y='MonthlyCharges', data=self.df)
        plt.title('Churn vs. Monthly Charges')
        plt.show()

    def plot_scatter_tenure_vs_monthly_charges(self):
        sns.scatterplot(x='tenure', y='MonthlyCharges', hue='Churn', data=self.df)
        plt.title('Tenure vs. Monthly Charges (Colored by Churn)')
        plt.show()

    def plot_pairplot(self):
        sns.pairplot(self.df[['tenure', 'MonthlyCharges', 'TotalCharges', 'Churn']], hue='Churn')
        plt.suptitle('Pair Plot of Tenure, Monthly Charges, and Total Charges by Churn', y=1.02)
        plt.show()

    def plot_correlation_heatmap(self):
        numerical_features = self.df[['tenure', 'MonthlyCharges', 'TotalCharges']]
        corr = numerical_features.corr()

        plt.figure(figsize=(8, 6))
        sns.heatmap(corr, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
        plt.title('Heatmap: Correlation Matrix of Tenure, Monthly Charges, and Total Charges')
        plt.show()


if __name__ == "__main__":
    from loading import DataLoader
    from preprocessing import DataPreprocessor


    loader = DataLoader(r'C:\Users\aruni\Desktop\customer churn project\Telco-Customer-Churn.csv')
    data = loader.load_data()
```

```
    loader = DataLoader(r'C:\Users\aruni\Desktop\customer churn project\Telco-Customer-Churn.csv')
    data = loader.load_data()

    preprocessor = DataPreprocessor(data)
    preprocessor.convert_total_charges()
    preprocessor.drop_unwanted_columns()

    visualizer = DataVisualizer(data)

    visualizer.plot_churn_pie_chart()
    visualizer.plot_histograms()
    visualizer.plot_contract_churn()
    visualizer.plot_boxplot_churn_vs_tenure()
    visualizer.plot_boxplot_churn_vs_monthly_charges()
    visualizer.plot_scatter_tenure_vs_monthly_charges()
    visualizer.plot_pairplot()
    visualizer.plot_correlation_heatmap()
```

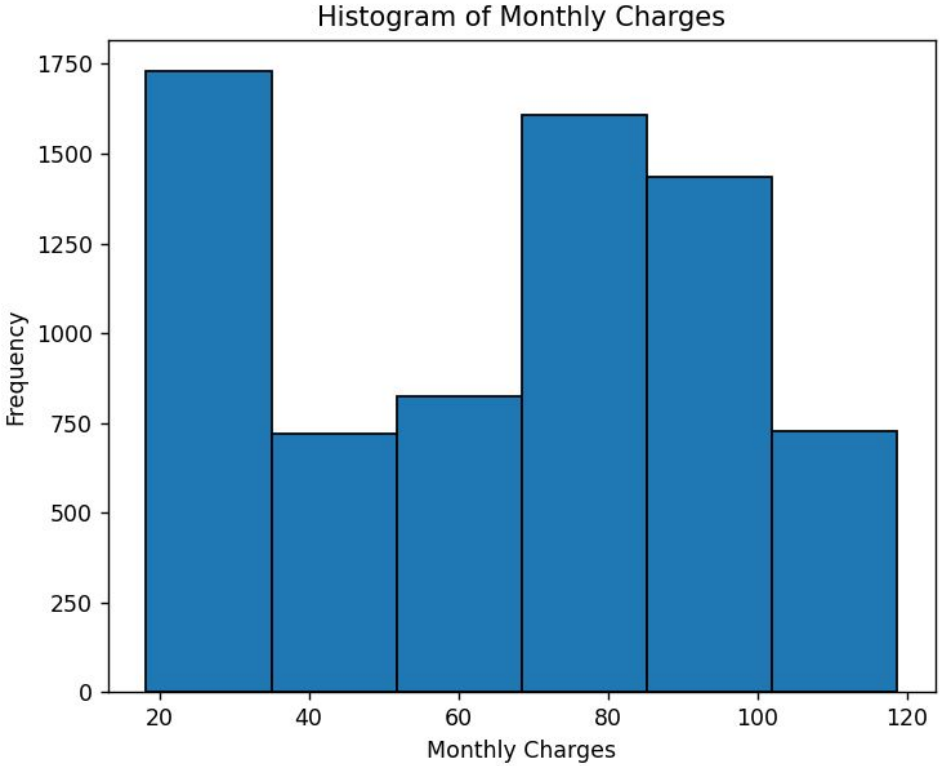
```python
from loading import DataLoader
from preprocessing import DataPreprocessor
from visualization import DataVisualizer

def main():
    try:
        loader = DataLoader(r'C:\Users\aruni\Desktop\customer churn project\Telco-Customer-Churn.csv')
        data = loader.load_data()

        print("First few rows of the dataset:\n", loader.display_head())
        print("Dataset shape:", loader.display_shape())
        print("Summary statistics:\n", loader.display_summary())

        preprocessor = DataPreprocessor(data)
        preprocessor.check_info()
        preprocessor.convert_total_charges()
        preprocessor.drop_unwanted_columns()
        preprocessor.display_shape()
        preprocessor.final_check()

        visualizer = DataVisualizer(data)
        visualizer.plot_churn_pie_chart()
        visualizer.plot_histograms()
        visualizer.plot_contract_churn()
        visualizer.plot_boxplot_churn_vs_tenure()
        visualizer.plot_boxplot_churn_vs_monthly_charges()
        visualizer.plot_scatter_tenure_vs_monthly_charges()
        visualizer.plot_pairplot()
        visualizer.plot_correlation_heatmap()

    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```

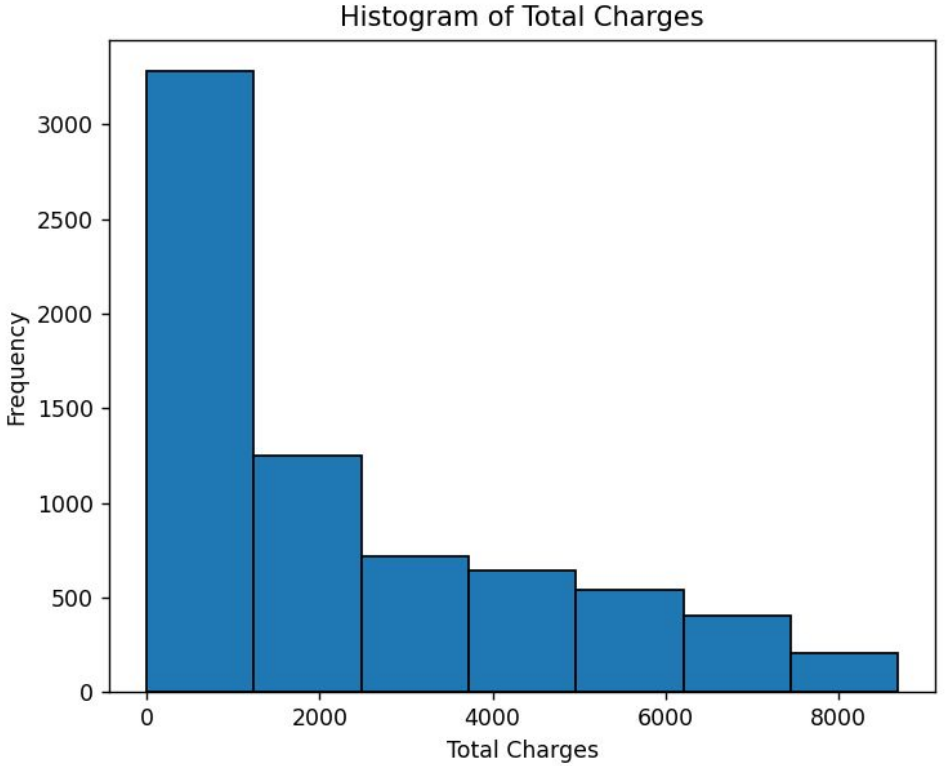
# RESULTS AND OUTCOMES

# Pie Chart

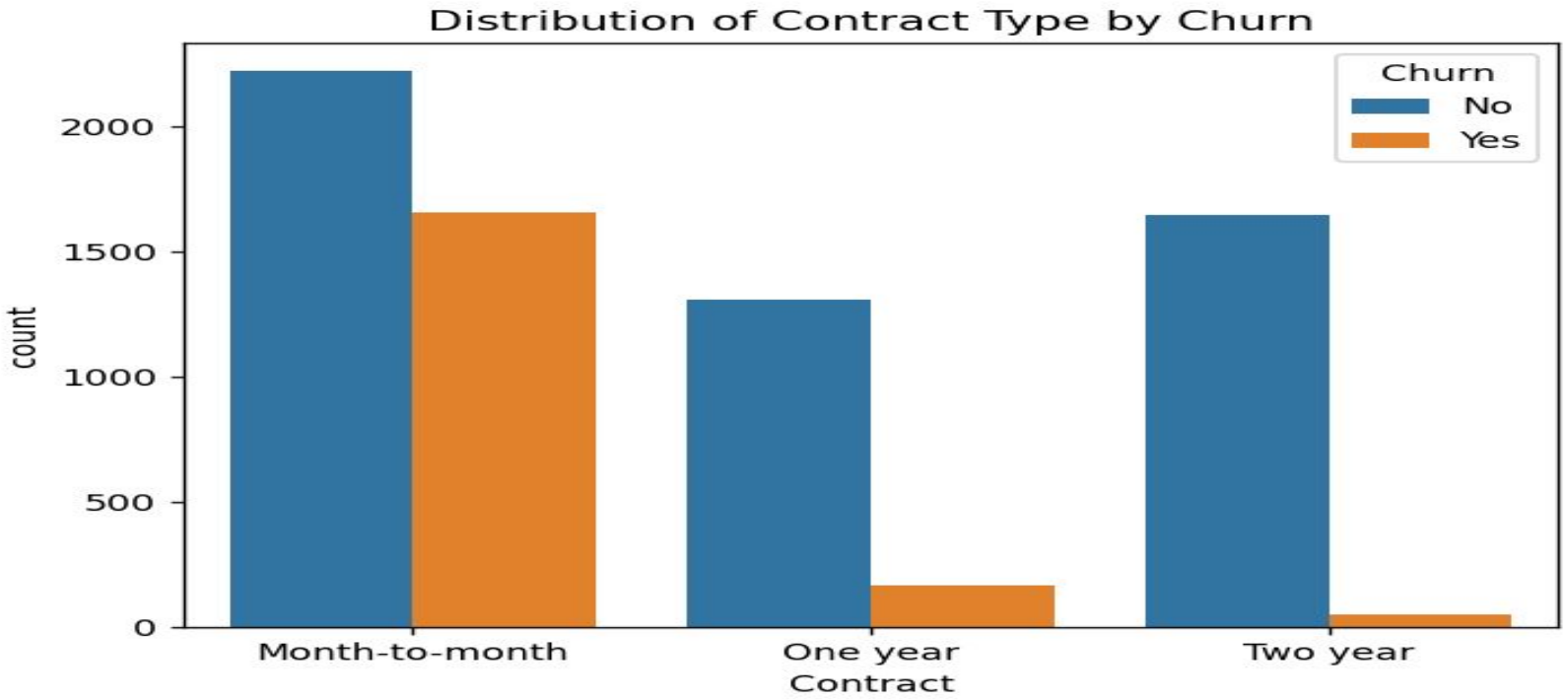


# Histogram Plot

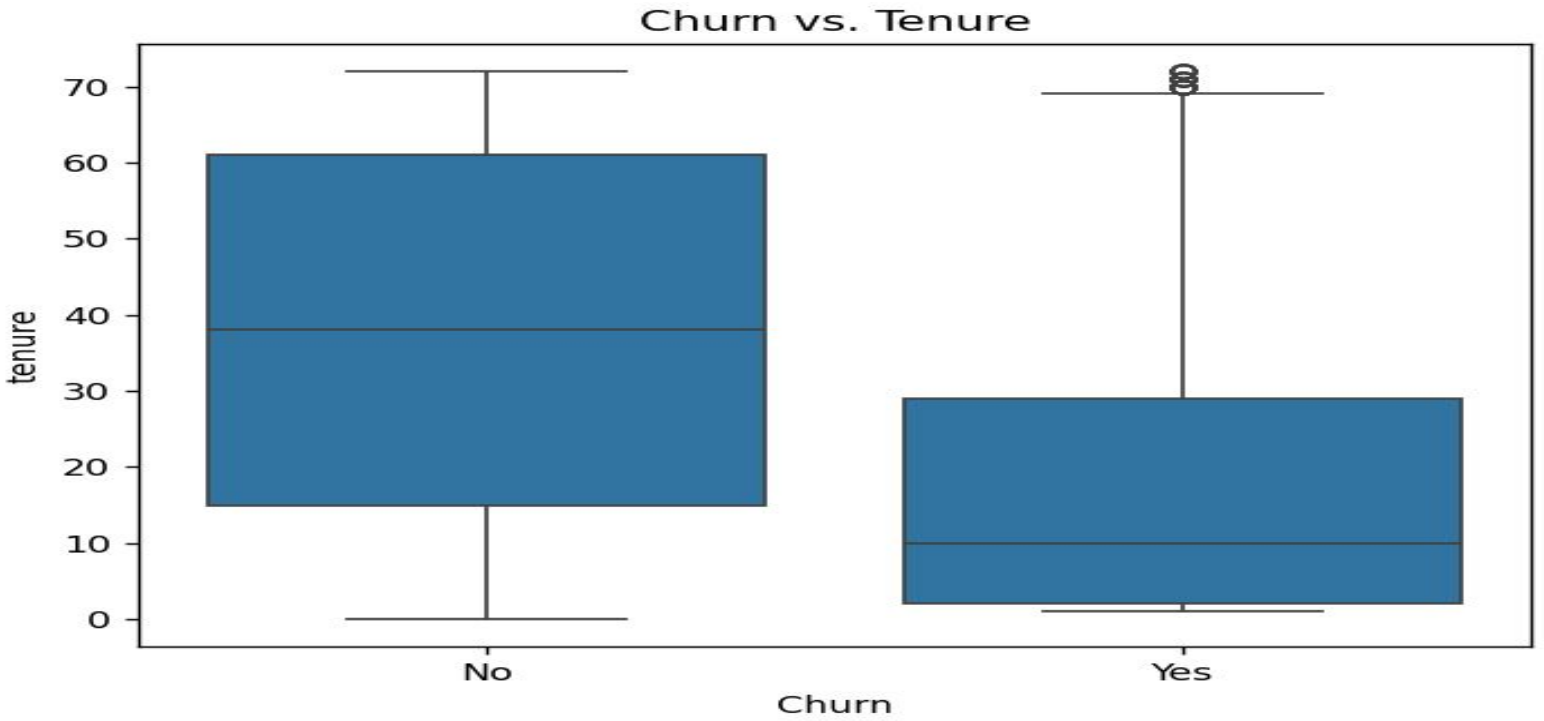


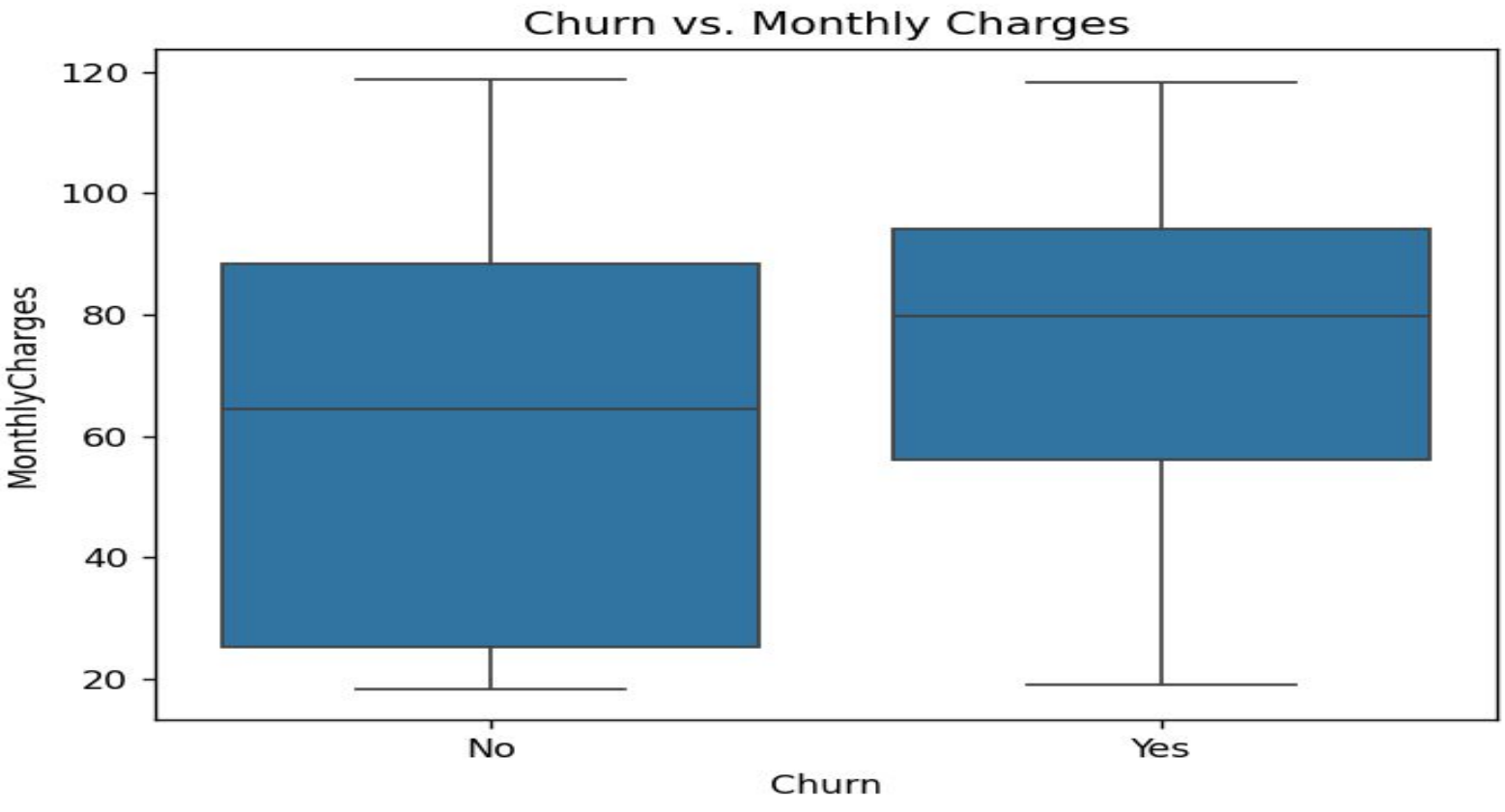



# Count Plot

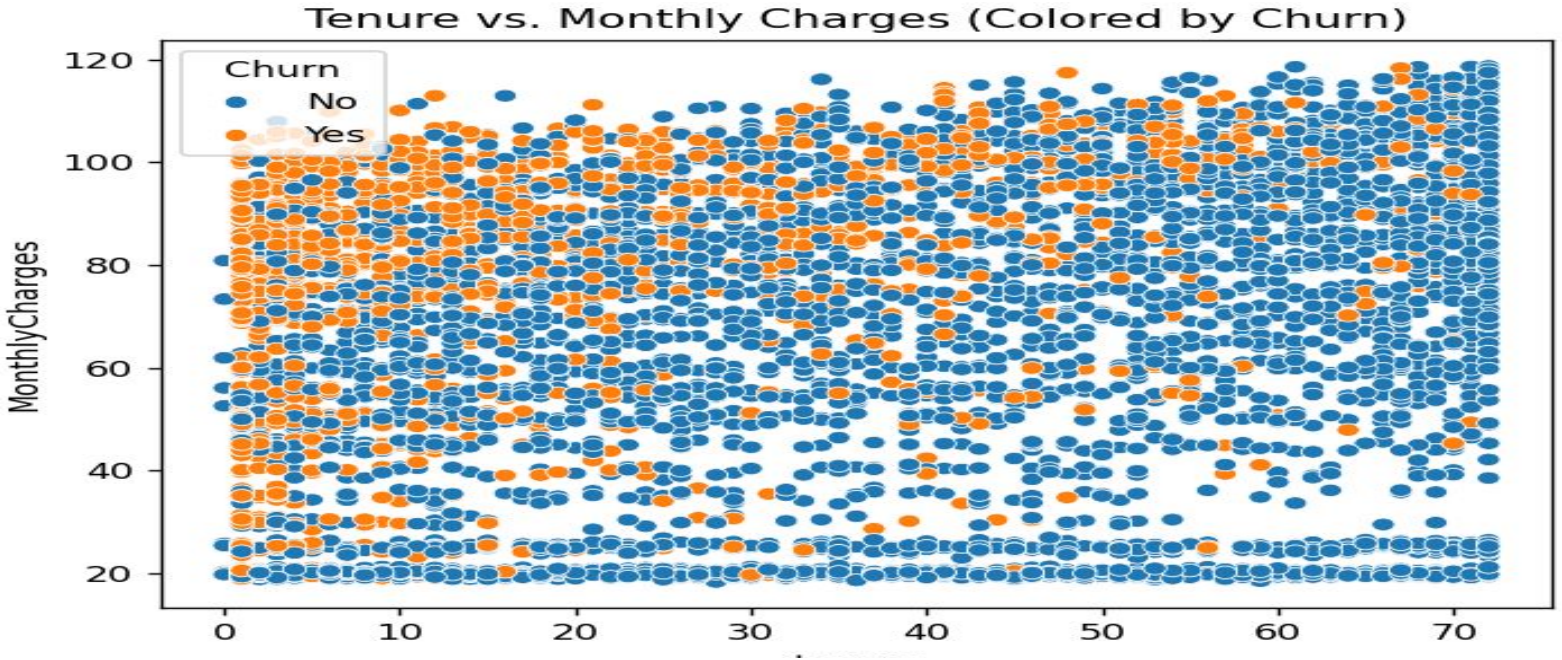

Distribution of Contract Type by Churn
Churn
No
Yes
count
2000
1500
1000
500
0
Month-to-month
One year
Two year
Contract

# Box Plot

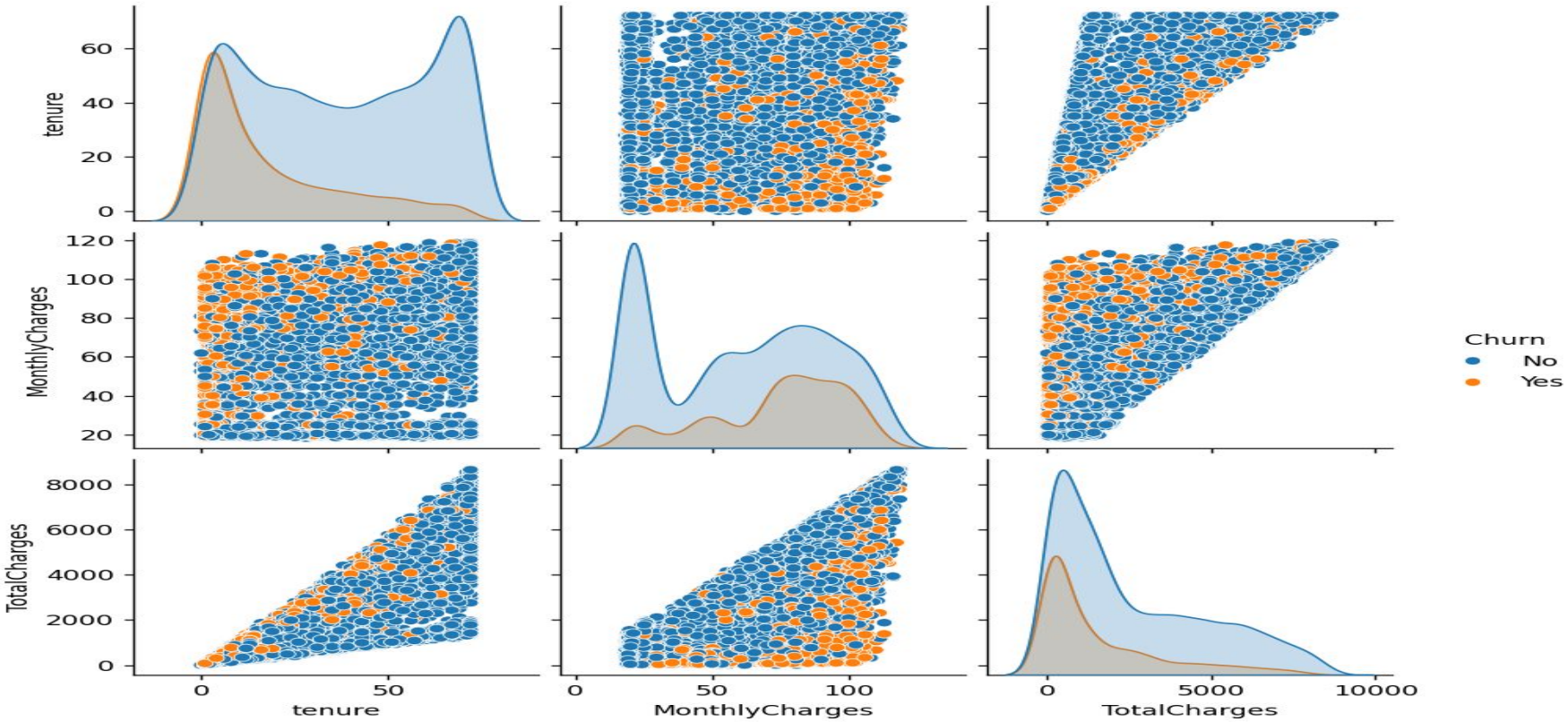



Churn vs. Monthly Charges
120
100
80
60
40
20
MonthlyCharges
No
Yes
Churn

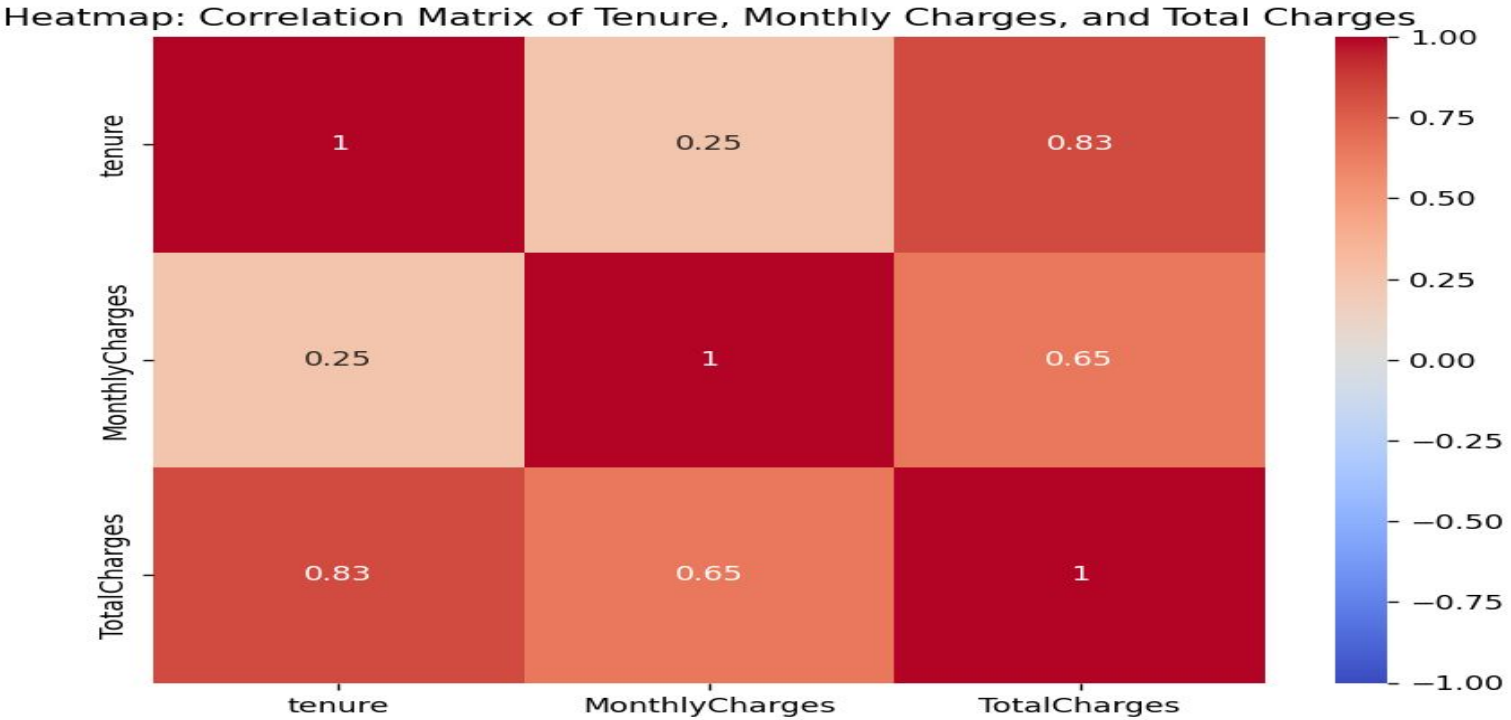
# Scatter plot



# Pair Plot


tenure
MonthlyCharges
TotalCharges
tenure
MonthlyCharges
TotalCharges
Churn
No
Yes

# Heatmap


Heatmap: Correlation Matrix of Tenure, Monthly Charges, and Total Charges
tenure
MonthlyCharges
TotalCharges
1
0.25
0.83
0.25
1
0.65
0.83
0.65
1
tenure
MonthlyCharges
TotalCharges
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

- The churn rate of  26.5% poses a significant challenge to revenue growth, particularly with short-term contract customers being the most likely to leave the service.
- Customers with higher monthly charges are more prone to churn, suggesting that the company is losing some of its most valuable, high-paying users.
- Analysis shows that shorter-tenure customers  are at greater risk of churning, highlighting the importance of retention strategies for new customers.
- Longer-term customers contribute more to total charges, emphasizing the need to incentivize extended contracts for sustained revenue.
- Targeted retention strategies focused on high-paying, low-tenure customers will be critical for reducing churn and enhancing long-term revenue growth.

# CONCLUSION

- The analysis highlights that customer retention is crucial for sustained revenue growth, as longer-tenured customers contribute significantly more to total charges.
- Short-term contracts and higher monthly charges are strong indicators of churn, making it essential for the company to implement retention strategies targeting these at-risk customers.
- The findings suggest that upselling premium services and encouraging longer-term contracts can effectively increase both monthly and total charges, boosting overall revenue.
- Future improvements could focus on refining customer segmentation to develop personalized retention strategies for high-value, low-tenure customers.
- By leveraging insights from data analysis, the company can continue optimizing retention efforts and improving pricing models to enhance long-term growth and profitability.

# THANK YOU…