Customer Churn Prediction Using Machine Learning

Project members:

Emrullah Ayaz

Selahattin Koray Yıldız

Problem Statement

Customer churn rate is the rate at which a user using an application or an individual who is receiving any service abandons the service or application they are using. This rate is a very important issue for companies. Customer churn, also known as customer attrition, refers to the phenomenon where customers stop doing business with a company. Predicting customer churn is crucial for businesses as it helps them identify at-risk customers and take proactive measures to retain them. Retaining existing customers is often more cost-effective than acquiring new ones, making churn prediction a critical task for improving customer satisfaction and profitability. Machine learning plays a significant role in solving this problem by analyzing historical customer data to identify patterns and predict which customers are likely to churn. By leveraging machine learning algorithms, businesses can develop targeted retention strategies, such as personalized offers or improved customer service, to reduce churn rates. This project aims to build a machine learning model to predict customer churn based on customer behavior, demographics, and transaction history.

Dataset Overview

- **Brief Description of the Dataset:** Telco Customer Churn dataset is a tabular dataset that generally contains detailed information about the customers of a telecommunications company. This dataset provides data for 7043 customers and includes many details such as whether the customers left or not, the reasons for leaving if so, the monthly and total amounts they paid, the locations they live in, and the services they received. With this data, we aimed to use machine learning techniques to identify existing customers at risk of leaving and to develop predictive models to prevent these customers from leaving. This dataset contains 19 categorical, 9 numerical, 4 geographical, and 1 textual feature.
- **Number of Instances (Rows):** 7043
- **Number of Features (Columns):** 33
- **Types of Features:**
  - Categorical: CustomerID, Gender, Senior Citizen, Partner, Dependents, Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tech Support, Streaming TV, Streaming Movies, Contract, Paperless Billing, Payment Method,Churn Label, Churn Reason

- **Numerical:** Count, Zip Code, Latitude, Longitude, Tenure Months, Monthly Charges, Total Charges, Churn Value, Churn Score, CLTV

- **Geographical:** Country, State, City, Lat Long
- **Textual:** Churn Reason

- Target Variable: Churn Value (Binary, 1 for churned and 0 for not churned), Churn Label (Categorical, yes/no)
- Duplicate Entries: No, there is no duplicate entry in the dataset.

[1]:
```python
import pandas as pd
df = pd.read_excel("Telco_customer_churn.xlsx")
duplicates = df.duplicated().sum()

if duplicates == 0:
    print("No duplicates found.")
else:
    print(f"Found {duplicates} duplicate rows.")
```

```
No duplicates found.
```

Summary Statistics

Our Dataset is a Tabular Dataset.

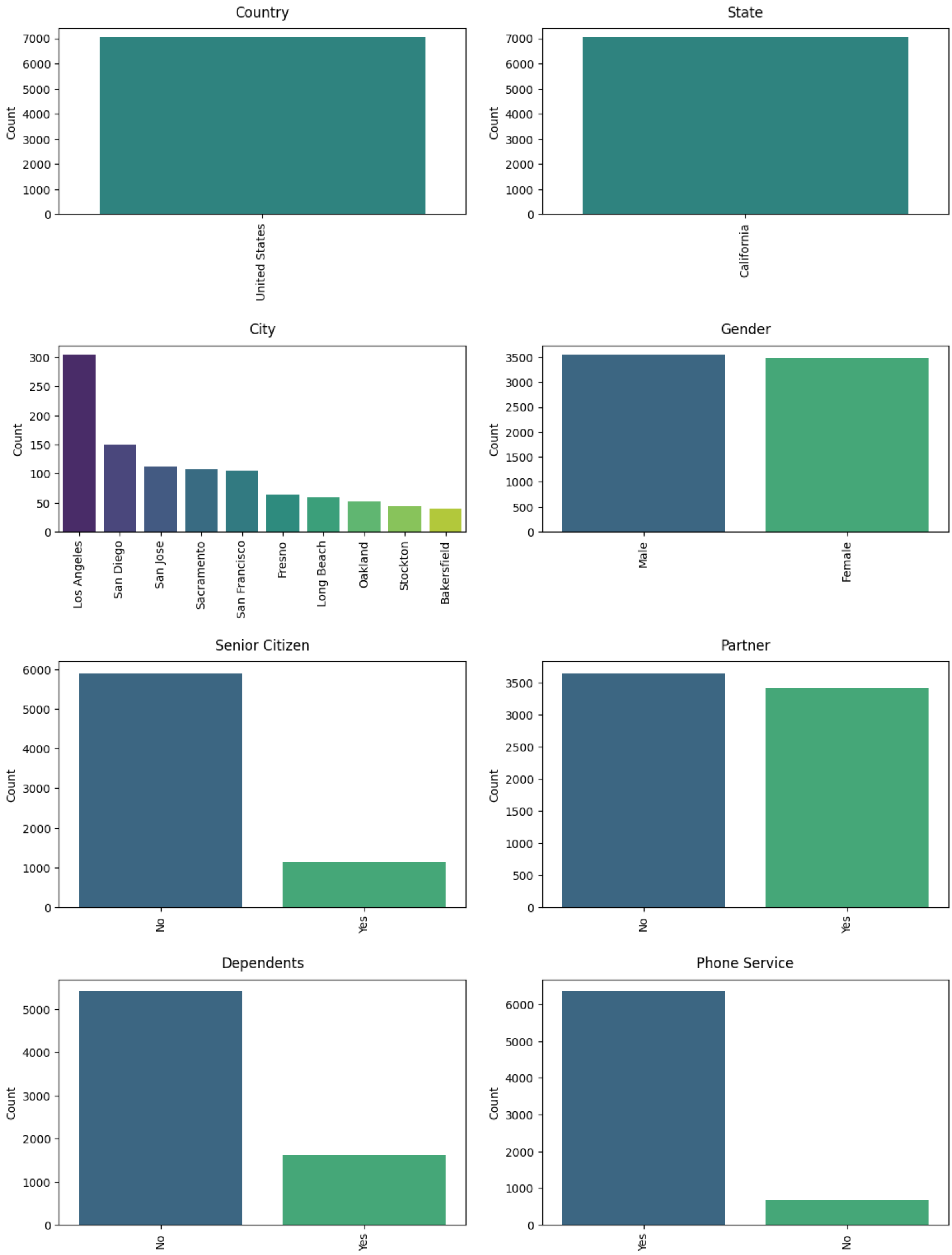
- **Numerical Features:**

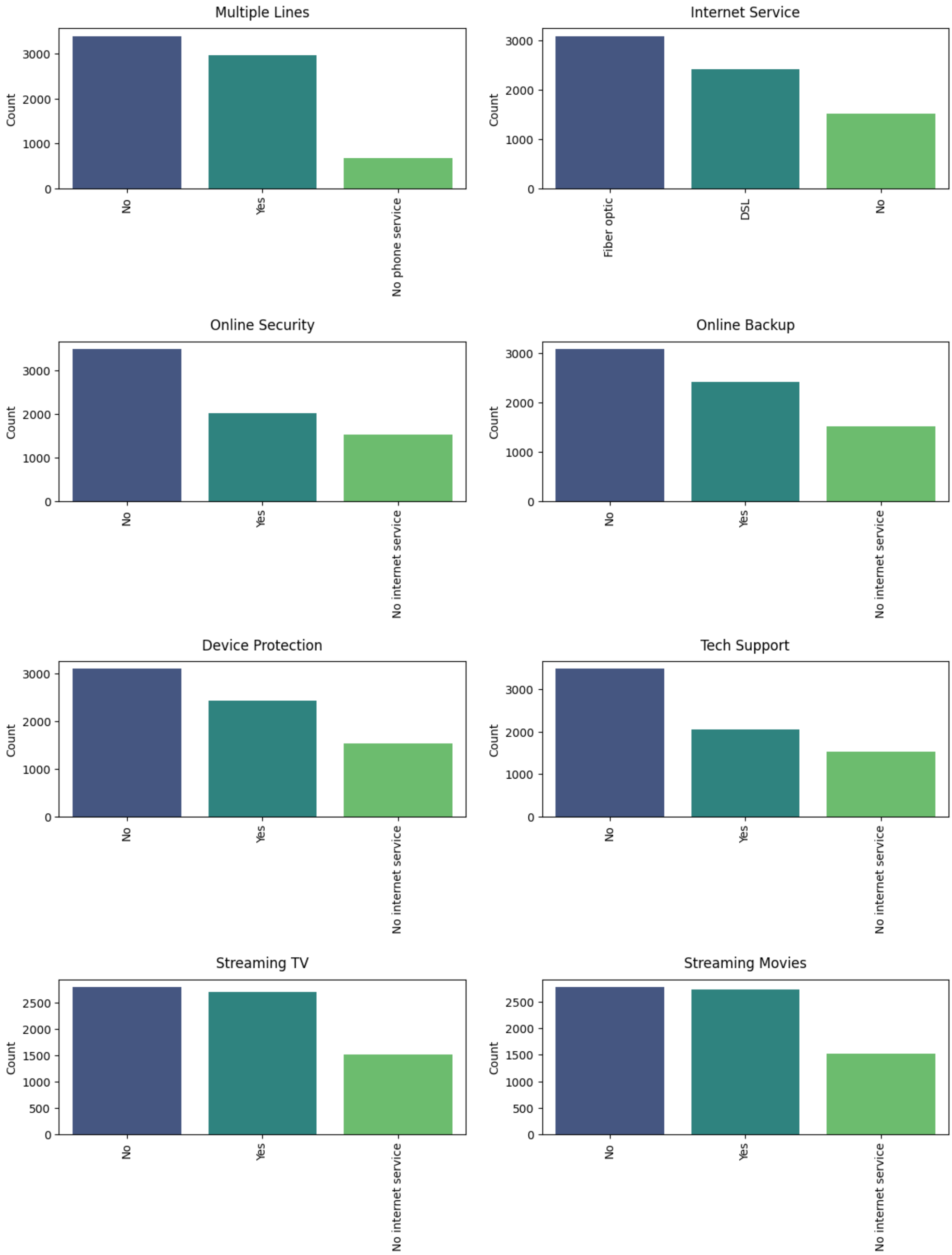
```python
# 'Total Charges' column datatype is seems an object, but it is actually numerical feature.
# To show correct summary statistics, we change the total charges datatype from object to flaot64
df['Total Charges'] = pd.to_numeric(df['Total Charges'], errors='coerce')
df.describe()
```

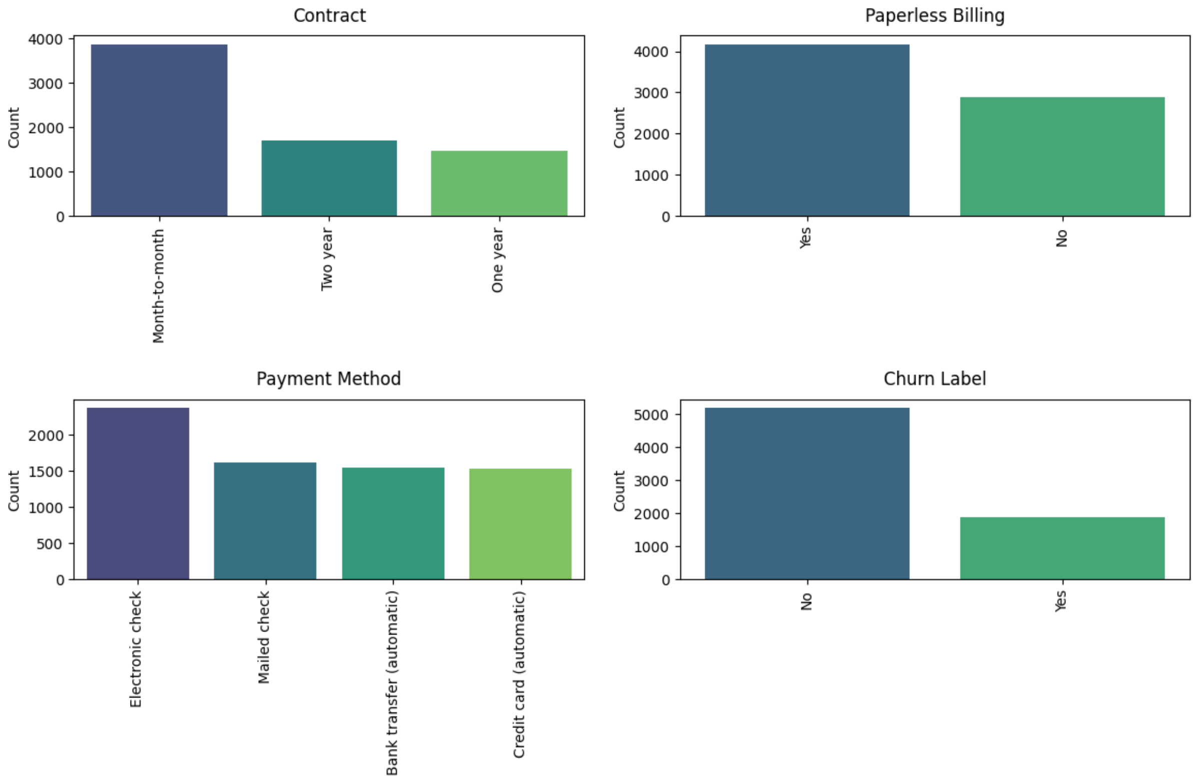
	Count	Zip Code	Latitude	Longitude	Tenure Months	Monthly Charges	Total Charges	Churn Value	Churn Score	CLTV
count	7043.0	7043.000000	7043.000000	7043.000000	7043.000000	7043.000000	7032.000000	7043.000000	7043.000000	7043.000000
mean	1.0	93521.964646	36.282441	-119.798880	32.371149	64.761692	2283.300441	0.265370	58.699418	4400.295755
std	0.0	1865.794555	2.455723	2.157889	24.559481	30.090047	2266.771362	0.441561	21.525131	1183.057152
min	1.0	90001.000000	32.555828	-124.301372	0.000000	18.250000	18.800000	0.000000	5.000000	2003.000000
25%	1.0	92102.000000	34.030915	-121.815412	9.000000	35.500000	401.450000	0.000000	40.000000	3469.000000
50%	1.0	93552.000000	36.391777	-119.730885	29.000000	70.350000	1397.475000	0.000000	61.000000	4527.000000
75%	1.0	95351.000000	38.224869	-118.043237	55.000000	89.850000	3794.737500	1.000000	75.000000	5380.500000
max	1.0	96161.000000	41.962127	-114.192901	72.000000	118.750000	8684.800000	1.000000	100.000000	6500.000000

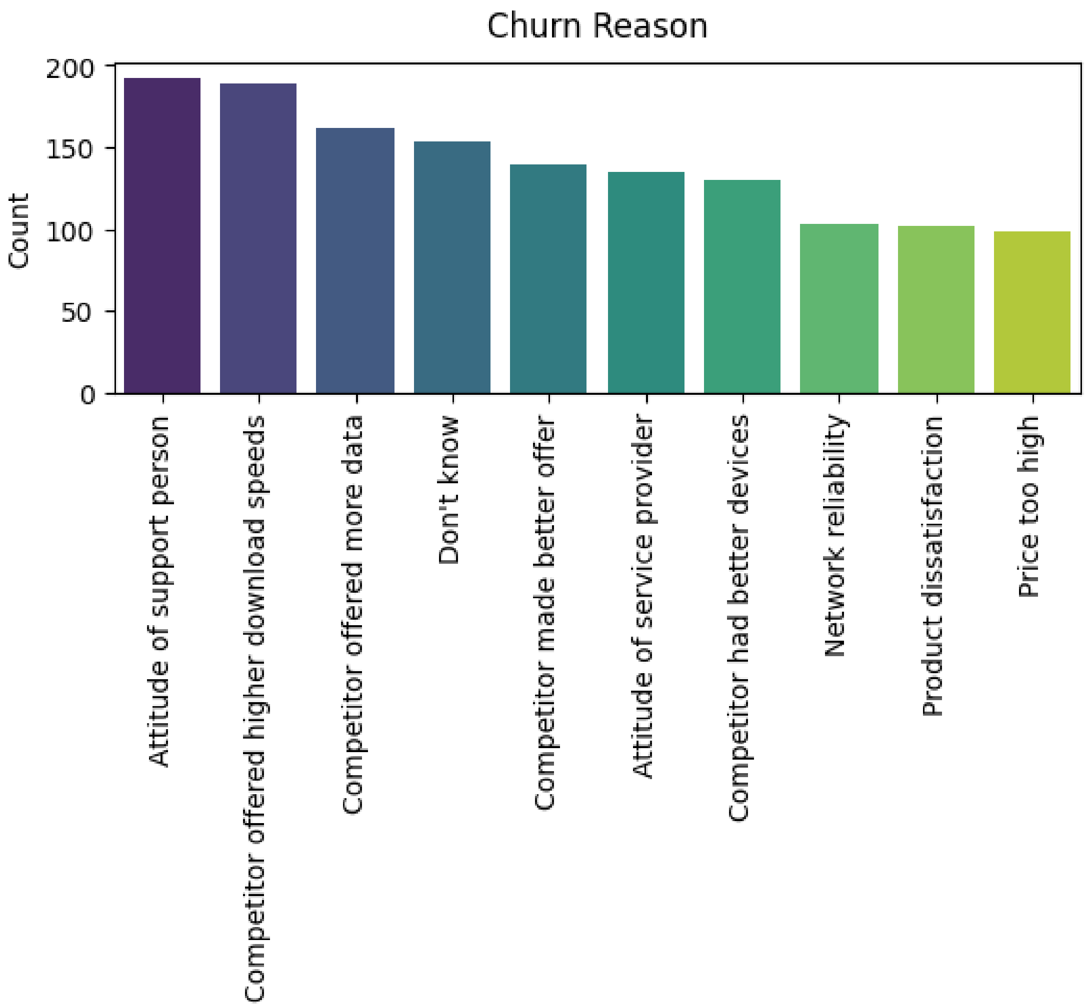
- ## **Categorical Features:**

```
# All categorical columns to visualize (excluding IDs and coordinates)
categorical_columns = ['Country', 'State', 'City', 'Gender','Senior Citizen', 'Partner', 'Dependents', 'Phone Service', 'Multiple Lines', 'Internet Service', 'Online Secur
    'Streaming TV', 'Streaming Movies', 'Contract', 'Paperless Billing','Payment Method', 'Churn Label', 'Churn Reason']
PLOTS_PER_IMAGE = 4
PLOT_WIDTH = 6
PLOT_HEIGHT = 4
num_images = math.ceil(len(categorical_columns) / PLOTS_PER_IMAGE)

for img_num in range(num_images):
    start_idx = img_num * PLOTS_PER_IMAGE
    end_idx = start_idx + PLOTS_PER_IMAGE
    current_features = categorical_columns[start_idx:end_idx]
    fig, axes = plt.subplots(2, 2, figsize=(PLOT_WIDTH * 2, PLOT_HEIGHT * 2))  # 2x2 grid
    axes = axes.flatten()  # Flatten to make iterating easier

    for i, col in enumerate(current_features):
        # Get the frequency count of each category
        value_counts = df[col].value_counts()
        # If there's no data or not enough categories, skip this column
        if len(value_counts) == 0:
            continue
        # Limit to the top 10 most frequent categories for readability
        value_counts = value_counts.nlargest(10)
        # Create plot - Bars representing frequency of each category
        sns.barplot(x=value_counts.index, y=value_counts.values, palette="viridis", ax=axes[i]) axes[i].set_title(f'{col}', pad=10)
        axes[i].set_ylabel('Count')
        axes[i].tick_params(axis='x', rotation=90)
        axes[i].set_xlabel('')  # Removes x-axis label
        plt.tight_layout(pad=2.0)

    for j in range(len(current_features), len(axes)):
        fig.delaxes(axes[j])

    # Save the figure
    output_file = f"categorical_features_{img_num+1}.png"
    plt.savefig(output_file, bbox_inches='tight')
    plt.close()
```


Country
Count
United States
State
California
City
Los Angeles
San Diego
San Jose
Sacramento
San Francisco
Fresno
Long Beach
Oakland
Stockton
Bakersfield
Gender
Male
Female
Senior Citizen
No
Yes
Partner
Dependents
Phone Service

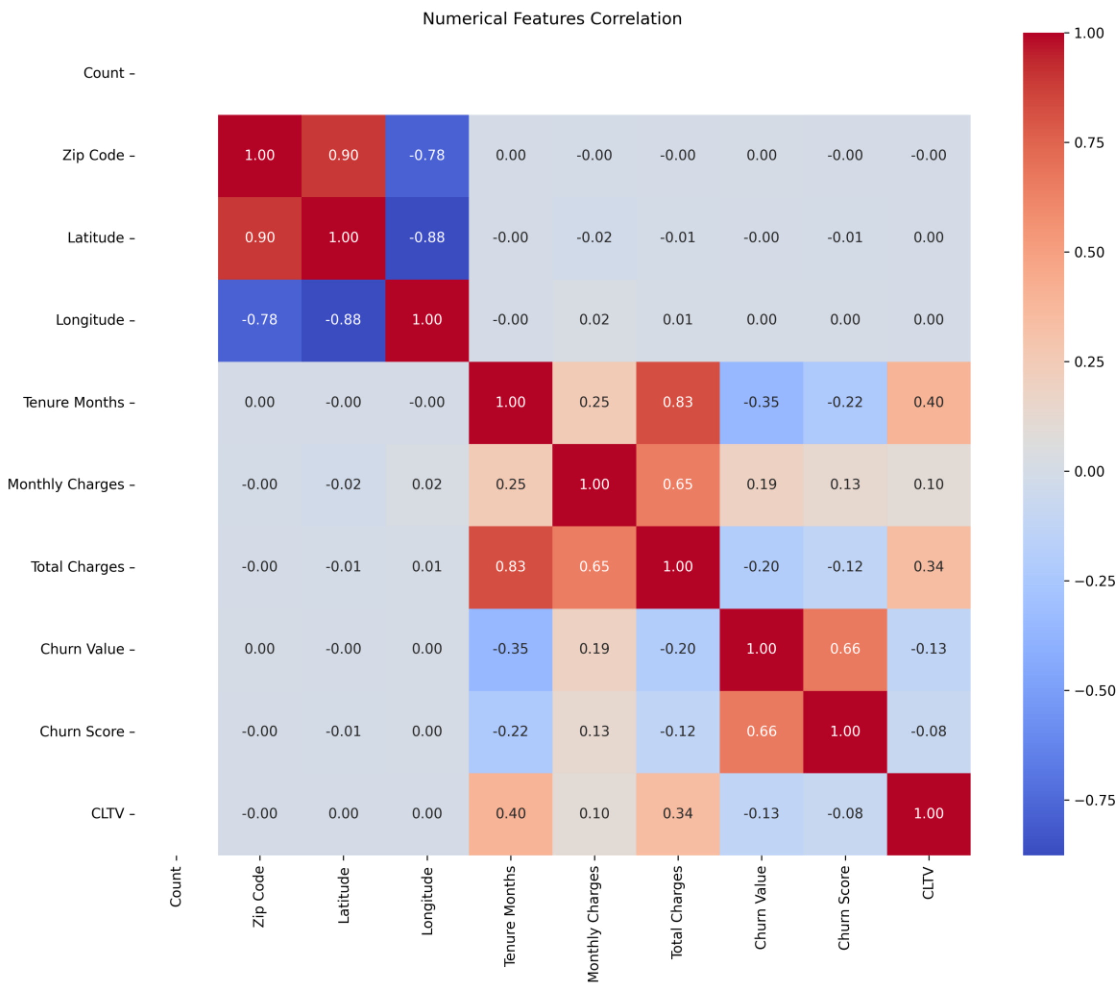

Multiple Lines
Count
0
1000
2000
3000
No
Yes
No phone service
Internet Service
Count
0
1000
2000
3000
Fiber optic
DSL
No
Online Security
Count
0
1000
2000
3000
No
Yes
No internet service
Online Backup
Count
0
1000
2000
3000
No
Yes
No internet service
Device Protection
Count
0
1000
2000
3000
No
Yes
No internet service
Tech Support
Count
0
1000
2000
3000
No
Yes
No internet service
Streaming TV
Count
0
500
1000
1500
2000
2500
No
Yes
No internet service
Streaming Movies
Count
0
500
1000
1500
2000
2500
No
Yes
No internet service


Contract
Count
4000
3000
2000
1000
0
Month-to-month
Two year
One year
Paperless Billing
Count
4000
3000
2000
1000
0
Yes
No
Payment Method
Count
2000
1500
1000
500
0
Electronic check
Mailed check
Bank transfer (automatic)
Credit card (automatic)
Churn Label
Count
5000
4000
3000
2000
1000
0
No
Yes


Churn Reason
Count
200
150
100
50
0
Attitude of support person
Competitor offered higher download speeds
Competitor offered more data
Don't know
Competitor made better offer
Attitude of service provider
Competitor had better devices
Network reliability
Product dissatisfaction
Price too high

- **Correlation Analysis:**

```
numerical_features = df.select_dtypes(include=['int64', 'float64'])

corr_matrix = numerical_features.corr(method='pearson')

# Heatmap
import seaborn as sns
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Numerical Features Correlation")
plt.tight_layout()
plt.tight_layout()

plt.savefig("correlation_matrix.png", dpi=300)
plt.show()
```

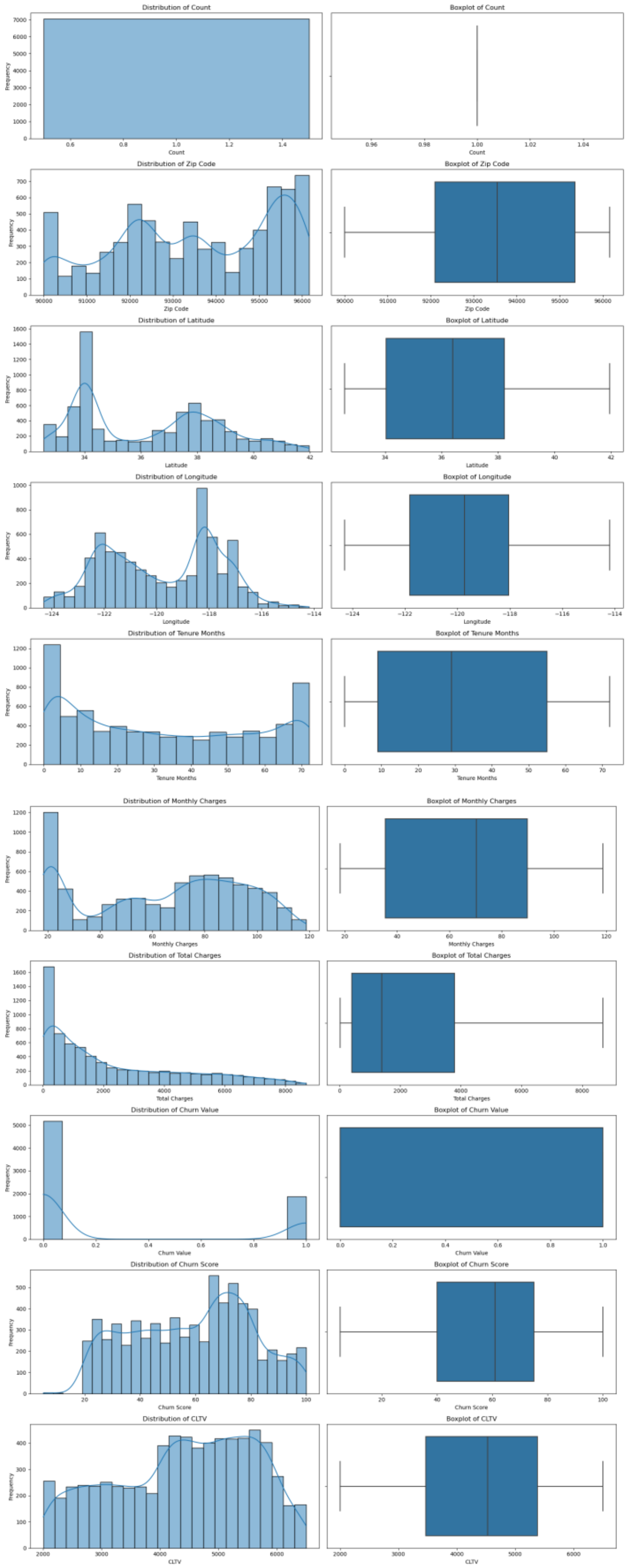


Exploratory Data Analysis (EDA)

- **Feature Distribution:**

Histograms & Boxplots for categorical features

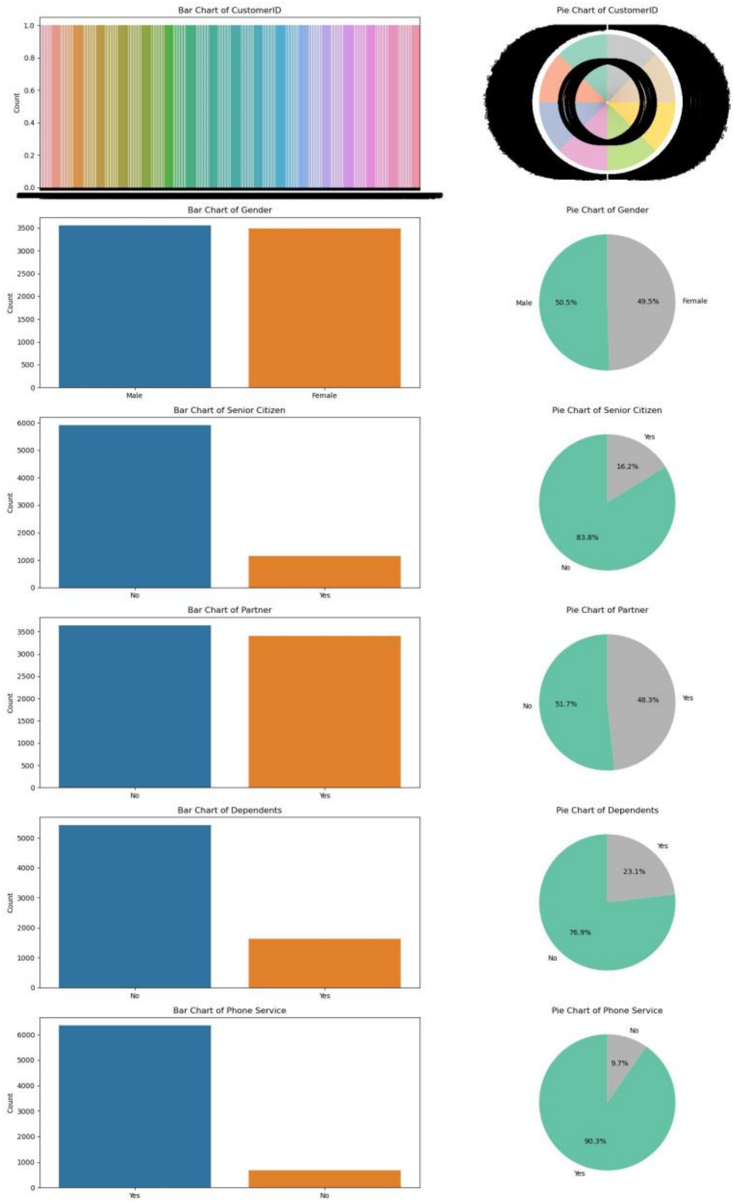
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import math

numerical_features = ['Count', 'Zip Code', 'Latitude', 'Longitude', 'Tenure Months',
                      'Monthly Charges', 'Total Charges', 'Churn Value', 'Churn Score', 'CLTV']

for feature in numerical_features:
    df[feature] = pd.to_numeric(df[feature], errors='coerce')

n = len(numerical_features)
cols = 2
rows = math.ceil(n / 2)


fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(16, 4 * rows))
for i, feature in enumerate(numerical_features[:5]):
    # Histogram
    sns.histplot(df[feature].dropna(), kde=True, ax=axes[i, 0])
    axes[i, 0].set_title(f'Distribution of {feature}')
    axes[i, 0].set_xlabel(feature)
    axes[i, 0].set_ylabel('Frequency')

    # Boxplot
    sns.boxplot(x=df[feature].dropna(), ax=axes[i, 1])
    axes[i, 1].set_title(f'Boxplot of {feature}')
    axes[i, 1].set_xlabel(feature)

plt.tight_layout()
plt.savefig("numerical_distributions_part1.png")



fig, axes = plt.subplots(nrows=rows, ncols=cols, figsize=(16, 4 * rows))
for i, feature in enumerate(numerical_features[5:]):
    # Histogram
    sns.histplot(df[feature].dropna(), kde=True, ax=axes[i, 0])
    axes[i, 0].set_title(f'Distribution of {feature}')
    axes[i, 0].set_xlabel(feature)
    axes[i, 0].set_ylabel('Frequency')

    # Boxplot
    sns.boxplot(x=df[feature].dropna(), ax=axes[i, 1])
    axes[i, 1].set_title(f'Boxplot of {feature}')
    axes[i, 1].set_xlabel(feature)

plt.tight_layout()
plt.savefig("numerical_distributions_part2.png")
```


Distribution of Count
Frequency
Count
Boxplot of Count
Count
Distribution of Zip Code
Frequency
Zip Code
Boxplot of Zip Code
Zip Code
Distribution of Latitude
Frequency
Latitude
Boxplot of Latitude
Latitude
Distribution of Longitude
Frequency
Longitude
Boxplot of Longitude
Longitude
Distribution of Tenure Months
Frequency
Tenure Months
Boxplot of Tenure Months
Tenure Months
Distribution of Monthly Charges
Frequency
Monthly Charges
Boxplot of Monthly Charges
Monthly Charges
Distribution of Total Charges
Frequency
Total Charges
Boxplot of Total Charges
Total Charges
Distribution of Churn Value
Frequency
Churn Value
Boxplot of Churn Value
Churn Value
Distribution of Churn Score
Frequency
Churn Score
Boxplot of Churn Score
Churn Score
Distribution of CLTV
Frequency
CLTV
Boxplot of CLTV
CLTV

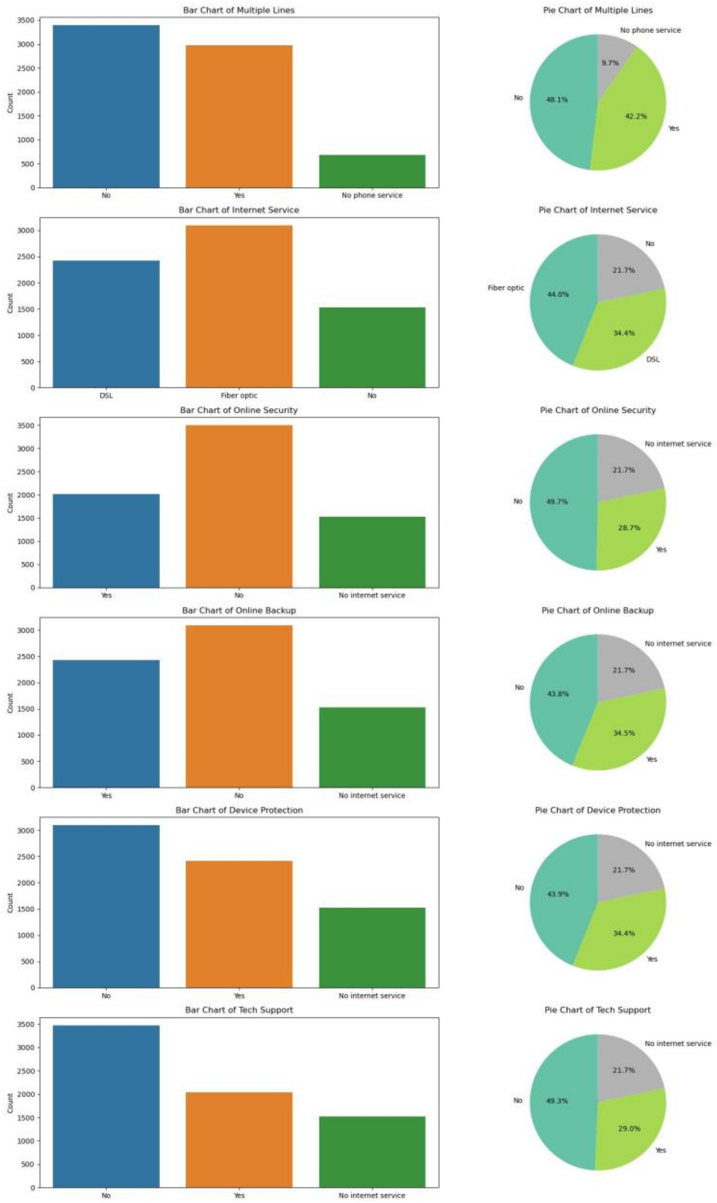
Bar charts &Pie charts for categorical features

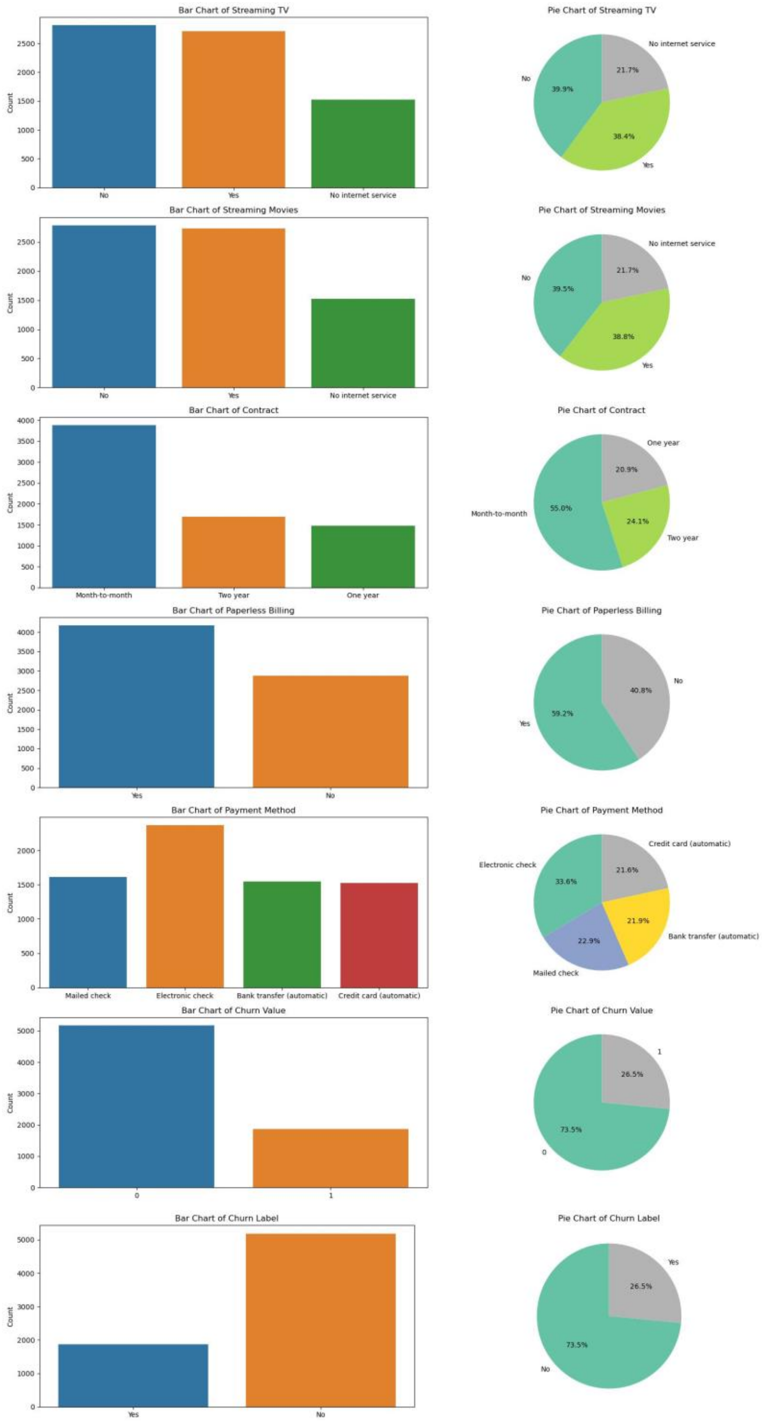
•[6]:

```
features_per_part = 6  # 6 features = 12 plots per part
num_parts = (len(categorical_features) + features_per_part - 1) // features_per_part

for part in range(num_parts):
    start_idx = part * features_per_part
    end_idx = min(start_idx + features_per_part, len(categorical_features))

    # 6 rows (1 for each feature) and 2 columns (bar and pie)
    fig, axes = plt.subplots(nrows=6, ncols=2, figsize=(16, 6*4))  # 6*4=24 inches height
    axes = axes.flatten()

    for i in range(start_idx, end_idx):
        feature = categorical_features[i]
        plot_idx = (i - start_idx) * 2  # Each feature takes 2 plots

        # Bar plot (left column)
        sns.countplot(x=df[feature], ax=axes[plot_idx])
        axes[plot_idx].set_title(f'Bar Chart of {feature}')
        axes[plot_idx].set_xlabel('')
        axes[plot_idx].set_ylabel('Count')

        # Pie plot (right column)
        df[feature].value_counts().plot(kind='pie', autopct='%1.1f%%',
                                        startangle=90, cmap='Set2', ax=axes[plot_idx+1])
        axes[plot_idx+1].set_title(f'Pie Chart of {feature}')
        axes[plot_idx+1].set_ylabel('')

    # Hide unused axes if the last part has less than 6 features
    for j in range((end_idx - start_idx) * 2, 12):
        axes[j].axis('off')

    fig.tight_layout()
    fig.savefig(f"categorical_distributions_part{part+1}.jpg", dpi=100, bbox_inches='tight')
    plt.close(fig)
```



Bar Chart of CustomerID
Count
1.0
0.8
0.6
0.4
0.2
0.0
Pie Chart of CustomerID
Bar Chart of Gender
Count
3500
3000
2500
2000
1500
1000
500
0
Male
Female
Pie Chart of Gender
Male
50.5%
49.5%
Female
Bar Chart of Senior Citizen
Count
6000
5000
4000
3000
2000
1000
0
No
Yes
Pie Chart of Senior Citizen
Yes
16.2%
83.8%
No
Bar Chart of Partner
Count
3500
3000
2500
2000
1500
1000
500
0
No
Yes
Pie Chart of Partner
No
51.7%
48.3%
Yes
Bar Chart of Dependents
Count
5000
4000
3000
2000
1000
0
No
Yes
Pie Chart of Dependents
Yes
23.1%
76.9%
No
Bar Chart of Phone Service
Count
6000
5000
4000
3000
2000
1000
0
Yes
No
Pie Chart of Phone Service
No
9.7%
90.3%
Yes

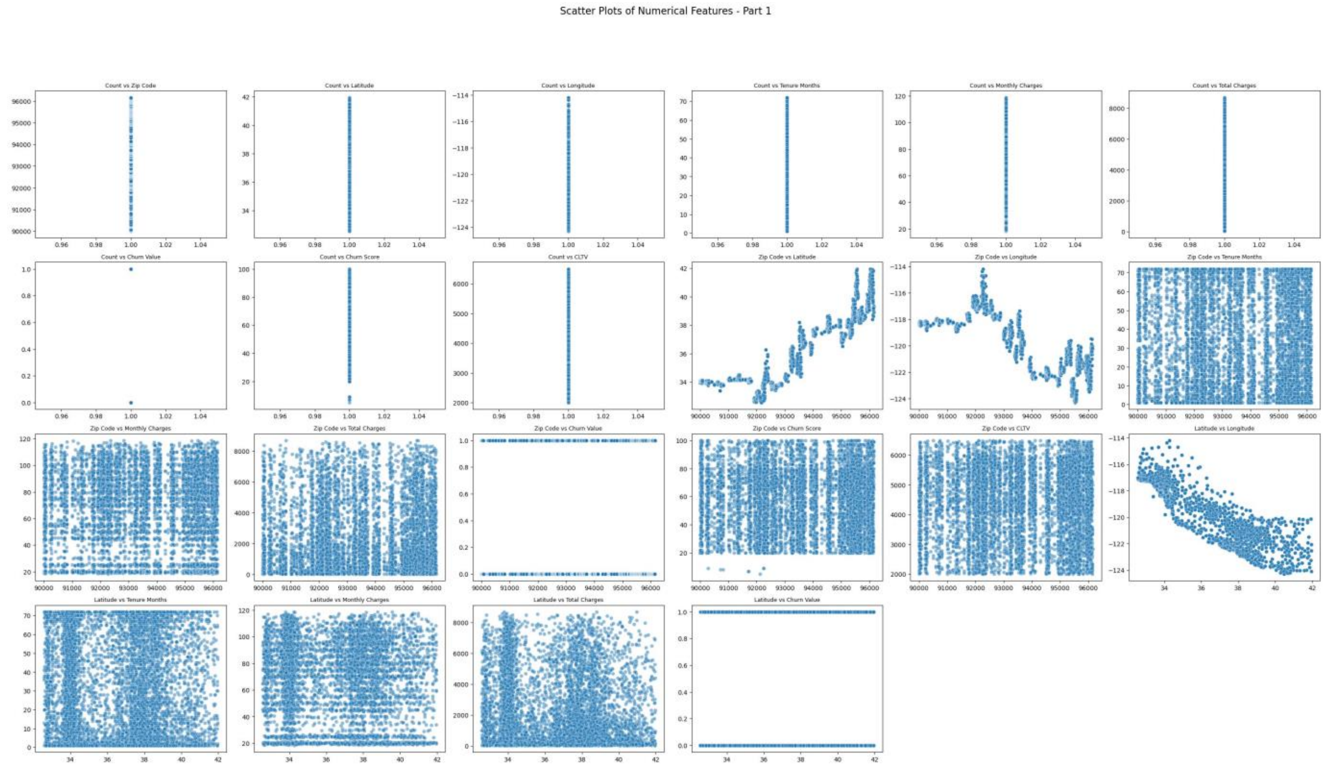

Bar Chart of Multiple Lines
Count
No
Yes
No phone service
Pie Chart of Multiple Lines
No phone service
9.7%
No
48.1%
42.2%
Yes
Bar Chart of Internet Service
Count
DSL
Fiber optic
No
Pie Chart of Internet Service
No
21.7%
Fiber optic
44.0%
34.4%
DSL
Bar Chart of Online Security
Count
Yes
No
No internet service
Pie Chart of Online Security
No internet service
21.7%
No
49.7%
28.7%
Yes
Bar Chart of Online Backup
Count
Yes
No
No internet service
Pie Chart of Online Backup
No internet service
21.7%
No
43.8%
34.5%
Yes
Bar Chart of Device Protection
Count
No
Yes
No internet service
Pie Chart of Device Protection
No internet service
21.7%
No
43.9%
34.4%
Yes
Bar Chart of Tech Support
Count
No
Yes
No internet service
Pie Chart of Tech Support
No internet service
21.7%
No
49.3%
29.0%
Yes

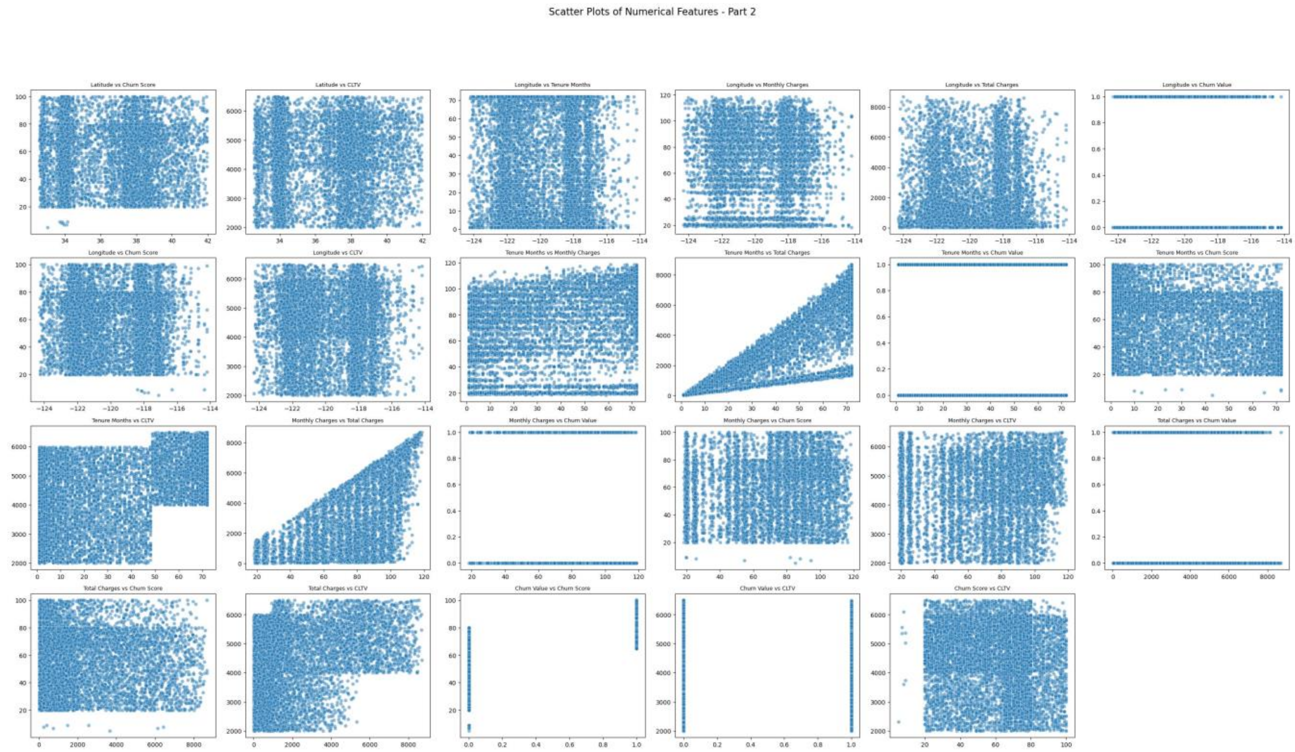


- **Pairwise Relationships:**

Scatter plots

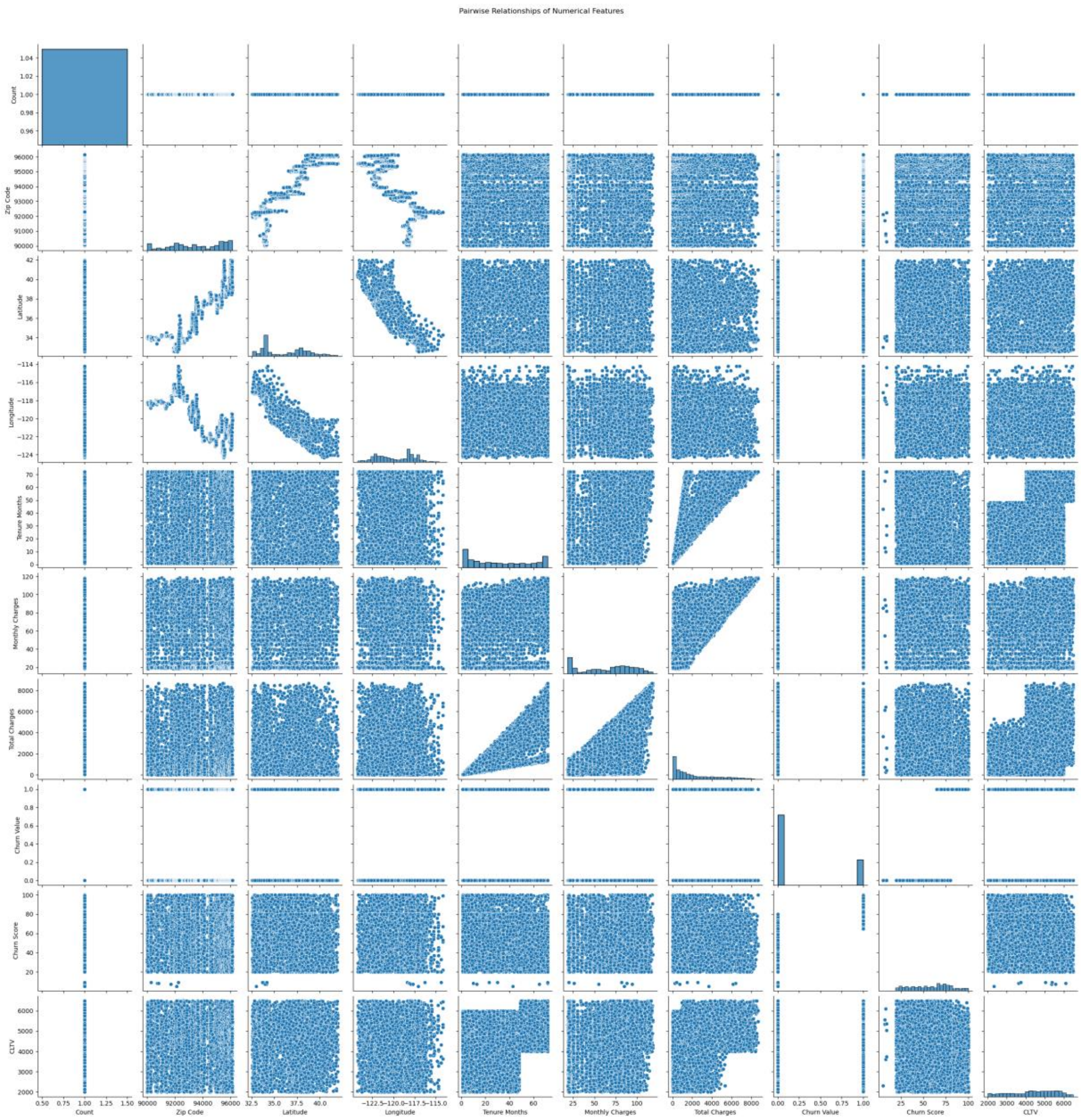
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from itertools import combinations
import math

numerical_features = ['Count', 'Zip Code', 'Latitude', 'Longitude', 'Tenure Months',
                      'Monthly Charges', 'Total Charges', 'Churn Value', 'Churn Score', 'CLTV']

for feature in numerical_features:
    df[feature] = pd.to_numeric(df[feature], errors='coerce')
df_num = df[numerical_features].dropna()
os.makedirs("scatterplots", exist_ok=True)
pairs = list(combinations(numerical_features, 2))
n = len(pairs)

cols = 6
rows = math.ceil(n / cols)
fig, axes = plt.subplots(rows, cols, figsize=(cols * 5, rows * 4))
axes = axes.flatten()

for i, (x, y) in enumerate(pairs[:n // 2]):
    ax = axes[i]
    sns.scatterplot(data=df_num, x=x, y=y, alpha=0.5, ax=ax)
    ax.set_title(f'{x} vs {y}', fontsize=9)
    ax.set_xlabel('')
    ax.set_ylabel('')

for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.suptitle('Scatter Plots of Numerical Features - Part 1', fontsize=16, y=1.02)
plt.subplots_adjust(top=0.96)
fig.savefig("scatterplots/scatter_plots_part1.jpeg", bbox_inches='tight')
plt.close()


fig, axes = plt.subplots(rows, cols, figsize=(cols * 5, rows * 4))
axes = axes.flatten()

for i, (x, y) in enumerate(pairs[n // 2:]):
    ax = axes[i]
    sns.scatterplot(data=df_num, x=x, y=y, alpha=0.5, ax=ax)
    ax.set_title(f'{x} vs {y}', fontsize=9)
    ax.set_xlabel('')
    ax.set_ylabel('')

for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])
plt.tight_layout()
plt.suptitle('Scatter Plots of Numerical Features - Part 2', fontsize=16, y=1.02)
plt.subplots_adjust(top=0.96)
fig.savefig("scatterplots/scatter_plots_part2.jpeg", bbox_inches='tight')
plt.close()
```


Scatter Plots of Numerical Features - Part 1


Scatter Plots of Numerical Features - Part 2

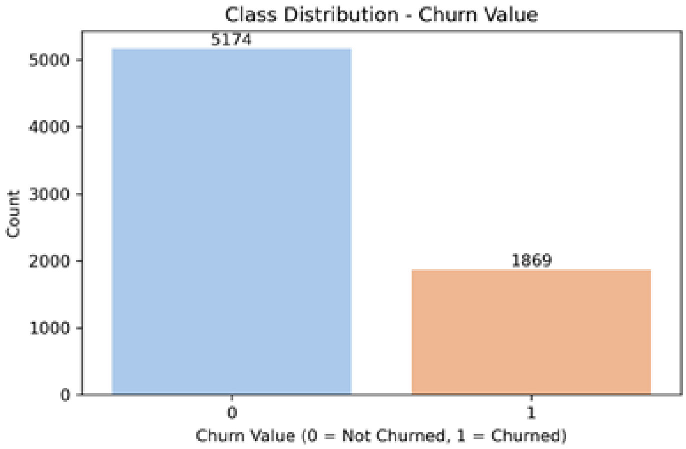
Pair plots

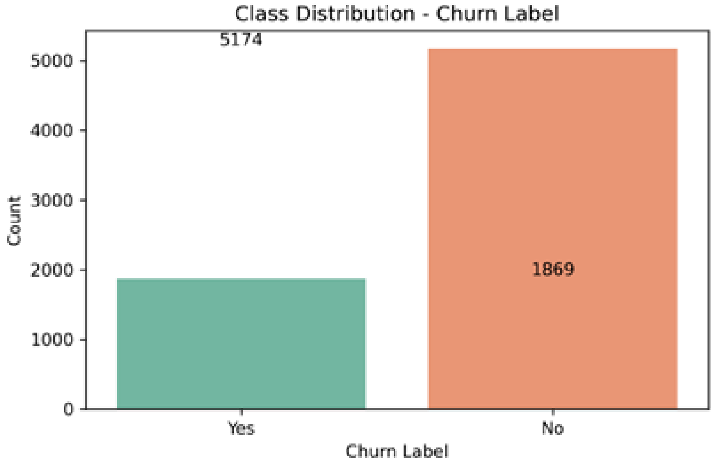
```
[ ]: import pandas as pd
     import seaborn as sns
     import matplotlib.pyplot as plt
     import os
     from itertools import combinations

     numerical_features = ['Count', 'Zip Code', 'Latitude', 'Longitude', 'Tenure Months',
                           'Monthly Charges', 'Total Charges', 'Churn Value', 'Churn Score', 'CLTV']


     for feature in numerical_features:
         df[feature] = pd.to_numeric(df[feature], errors='coerce')


     df_num = df[numerical_features].dropna()


     os.makedirs("pairplots", exist_ok=True)



     pairplot_fig = sns.pairplot(df_num)
     pairplot_fig.fig.suptitle("Pairwise Relationships of Numerical Features", y=1.02)
     pairplot_fig.savefig("pairplots/pairplot_all_numerical.png", bbox_inches='tight')
     plt.close()
```



- **Class Imbalance Check :**

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os


os.makedirs("class_imbalance", exist_ok=True)


plt.figure(figsize=(6, 4))
sns.countplot(x='Churn Value', data=df, palette='pastel')
plt.title('Class Distribution - Churn Value')
plt.xlabel('Churn Value (0 = Not Churned, 1 = Churned)')
plt.ylabel('Count')


for i, count in enumerate(df['Churn Value'].value_counts().sort_index()):
    plt.text(i, count + 50, str(count), ha='center')

plt.tight_layout()
plt.savefig("class_imbalance/churn_value_distribution.png", dpi=300)
plt.show()


plt.figure(figsize=(6, 4))
sns.countplot(x='Churn Label', data=df, palette='Set2')
plt.title('Class Distribution - Churn Label')
plt.xlabel('Churn Label')
plt.ylabel('Count')


for i, count in enumerate(df['Churn Label'].value_counts()):
    plt.text(i, count + 50, str(count), ha='center')

plt.tight_layout()
plt.savefig("class_imbalance/churn_label_distribution.png", dpi=300)
plt.show()
```





Summary and Insights

- **Key Findings from EDA:**

There is an 83 percent correlation between tenure months and total charges. We observe that these 2 features are highly correlated with each other and affect each other.

The 66 percent correlation between churn score and churn value shows that there is a connection between these 2 features, although not very tight.

There is a 90 percent correlation between zip code and latitude, but we comment that these 2 features cannot give us any information.

The results we reached from the correlation heatmap are as follows:

We are faced with a correlation table of 60 percent and above between churn score-churn value, total charges-tenure months, total charges-monthly charges, total charges-cltv.

Here are some of our important conclusions:
1- The high correlation between total charges-monthly charges can tell us that customers are churned after the first month, or we can have this data since it is the customer's first month. When we analyze in detail, if we remove the customers who are in their first month from this dataset and this correlation still continues, we can conclude that the customers left after the first month.

2- We see that the correlation between churn-score and churn-value is high. Churn score is a value between 0-100 and churn value is a binary value of 0-1. The correlation between them can give us information about the threshold after which customers tend to leave.

When we look at the Monthly Charges graph, we see that approximately 15 percent of customers pay 20 dollars. In general, other customers pay a monthly fee between 50-100 dollars. However, we could not see a visible difference between churn value and monthly charges in the scatter plot. From here, we reach the following conclusion: The reason for customers leaving is not the amount they pay monthly, but those who pay 20 dollars, 50 dollars and 100 dollars leave. This shows that the reason for leaving is not the monthly fee, but another reason.

In the scotter plot between churn value and churn score, we see that if the customer's churn value is above 80, they leave at a high rate, and if it is below 80, they do not leave. We also see this correlation clearly in the pairwise table. From here, we can conclude that we can make our prediction models based on churn value.

When we look at the Churn Reason table, we see that the rate of those leaving due to the behavior of our customer representative is higher than those leaving for other reasons. The conclusion we reach from here may be to warn our customer representatives. In addition, a high rate of those leaving does not know why they left, which shows that we need to do research on this issue.

- **Interesting Patterns and Trends:**

We found it strange that there was no solid correlation between churn score and monthly charges. The important conclusion we reached from this is that people pay but leave for another reason.

We also found the sharp distinction between churn score and churn value interesting. The vast majority of our data is that if the churn score is over 80, the churn value is definitely 1. From this we came to the conclusion that if we wanted to set a categorical threshold, our churn score would definitely be 80.

- **Challenges Faced During Analysis:**

It is challenging for us to eliminate features that will not be useful to us. When we think about what some features that we see have high correlations really mean, it is challenging to ask the question "Could it be possible?" and give the answer as a comment. For example, the correlation rate between zip code and latitude is 90 percent, but will this information be useful to us or not, will it provide us with real information while we are analyzing the zip code, will we do geographical analysis or numerical analysis, and so on. It is challenging to find answers to questions like these.

References

Jain, H., Khunteta, A., & Srivastava, S. (2020). Churn prediction in telecommunication using logistic regression and logit boost. *Procedia Computer Science*, *167*, 101-112. https://www.sciencedirect.com/science/article/pii/S1877050920306529

Kayaalp, F. (2017). Review of customer churn analysis studies in telecommunications industry. *Karaelmas Fen ve Mühendislik Dergisi*, *7*(2), 696-705.https://dergipark.org.tr/tr/download/article-file/1329508

Provost, F., & Fawcett, T. (2013). *Data science for business*. O'Reilly Media

Wagh, S. K., Andhale, A. A., Wagh, K. S., Pansare, J. R., Ambadekar, S. P., & Gawande, S. H. (2024). Customer churn prediction in telecom sector using machine learning techniques. Results in Control and Optimization, 14, 100342. https://doi.org/10.1016/j.rico.2023.100342

Yean, Z. C. (n.d.). Telco customer churn (IBM) dataset. Kaggle. Retrieved April 4, 2025, from https://www.kaggle.com/datasets/yeanzc/telco-customer-churn-ibm-dataset