Customer Churn Analysis Using Python

Python Data Analyst Project

1.Import Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

- Pandas and NumPy are essential for data manipulation and analysis.
- Matplotlib and Seaborn are used for data visualization, allowing us to create informative charts and graphs.
- Scikit-learn provides tools for model building, evaluation, and validation. #2. Load Data

```
df = pd.read_csv('/content/drive/MyDrive/Data Analysis/Python Project/Customer Churn Analysis/Customer Churn.csv') df.head()

{"type":"dataframe","variable_name":"df"}
```

- The dataset is read from a CSV file using pd.read_csv(), which allows us to work with the data in a structured format for analysis.

3.Data Exploration

Check Data Structure:

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
0   customerID        7043 non-null   object
1   gender            7043 non-null   object
2   SeniorCitizen     7043 non-null   int64
3   Partner           7043 non-null   object
4   Dependents        7043 non-null   object
5   tenure            7043 non-null   int64
6   PhoneService      7043 non-null   object
7   MultipleLines     7043 non-null   object
8   InternetService   7043 non-null   object
9   OnlineSecurity    7043 non-null   object
```

```
10  OnlineBackup      7043 non-null   object
11  DeviceProtection  7043 non-null   object
12  TechSupport       7043 non-null   object
13  StreamingTV       7043 non-null   object
14  StreamingMovies   7043 non-null   object
15  Contract          7043 non-null   object
16  PaperlessBilling  7043 non-null   object
17  PaymentMethod     7043 non-null   object
18  MonthlyCharges    7043 non-null   float64
19  TotalCharges      7043 non-null   object
    20  Churn             7043 non-null
    object dtypes: float64(1), int64(2),
    object(18) memory usage: 1.1+ MB
```

- df.head() displays the first few rows of the DataFrame for a quick look at the data.

Handle Missing Values and Data Types:

```
df["TotalCharges"] = df["TotalCharges"].replace(" ","0")
df["TotalCharges"] = df["TotalCharges"].astype("float")
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
0   customerID        7043 non-null   object
1   gender            7043 non-null   object
2   SeniorCitizen     7043 non-null   int64
3   Partner           7043 non-null   object
4   Dependents        7043 non-null   object
5   tenure            7043 non-null   int64
6   PhoneService      7043 non-null   object
7   MultipleLines     7043 non-null   object
8   InternetService   7043 non-null   object
9   OnlineSecurity    7043 non-null   object
10  OnlineBackup      7043 non-null   object
11  DeviceProtection  7043 non-null   object
12  TechSupport       7043 non-null   object
13  StreamingTV       7043 non-null   object
14  StreamingMovies   7043 non-null   object
15  Contract          7043 non-null   object
16  PaperlessBilling  7043 non-null   object
17  PaymentMethod     7043 non-null   object
18  MonthlyCharges    7043 non-null   float64
19  TotalCharges      7043 non-null   float64
    20  Churn             7043 non-null
```

```
    object dtypes: float64(2), int64(2),
    object(17)
memory usage: 1.1+ MB
```

- df.info() provides insights into the data types and non-null counts, helping identify any missing values.

Check for Null Values:

```
df.isnull().sum().sum()

0
```

Descriptive Statistics:

```
df.describe()
```

	SeniorCitizen	tenure	MonthlyCharges	TotalCharges
count	7043.000000	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692	2279.734304
std	0.368612	24.559481	30.090047	2266.794470
min	0.000000	0.000000	18.250000	0.000000
25%	0.000000	9.000000	35.500000	398.550000
50%	0.000000	29.000000	70.350000	1394.550000
75%	0.000000	55.000000	89.850000	3786.600000
max	1.000000	72.000000	118.750000	8684.800000

- df.describe() generates summary statistics for numerical columns, giving an overview of the data distribution.

4.Check for Duplicates

```
df["customerID"].duplicated().sum()

0
```

5. Convert Binary Variables:

```
def conv(value):
if value == 1:
return "yes"
else:          return
"no"

df['SeniorCitizen'] = df["SeniorCitizen"].apply(conv)
df.head()
```

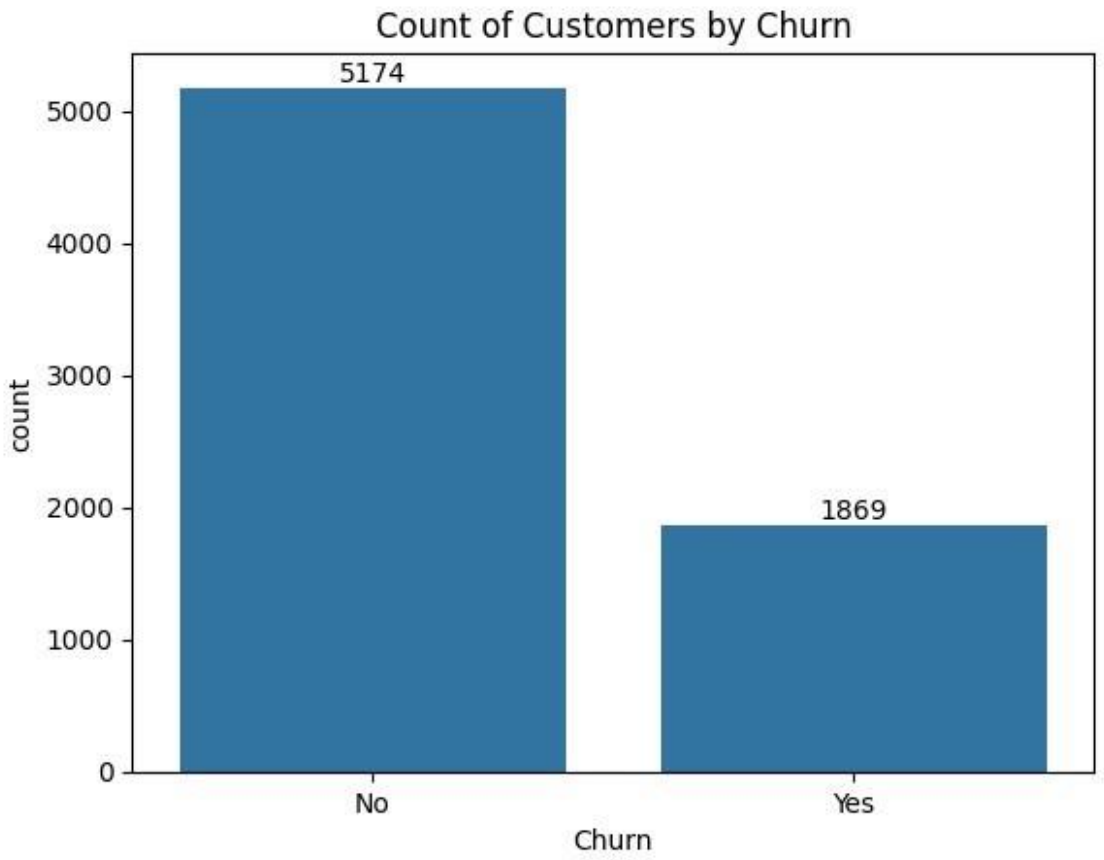
{"type":"dataframe","variable_name":"df"}

The code defines a function that converts numerical values in the "SeniorCitizen" column from 1 and 0 to the strings "yes" and "no" respectively. It then applies this function to the DataFrame, improving the readability of the data regarding senior citizen status.

6. Visualization

Churn Count Plot:

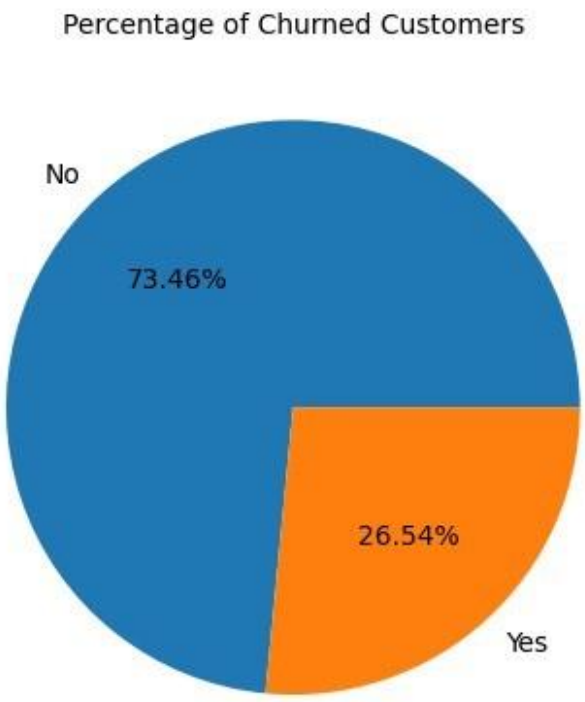
```
ax = sns.countplot(x='Churn', data=df)
ax.bar_label(ax.containers[0])
plt.title("Count of Customers by Churn")
plt.show()
```



Visualizes the count of customers who have churned versus those who have not.

Churn Percentage Pie Chart:

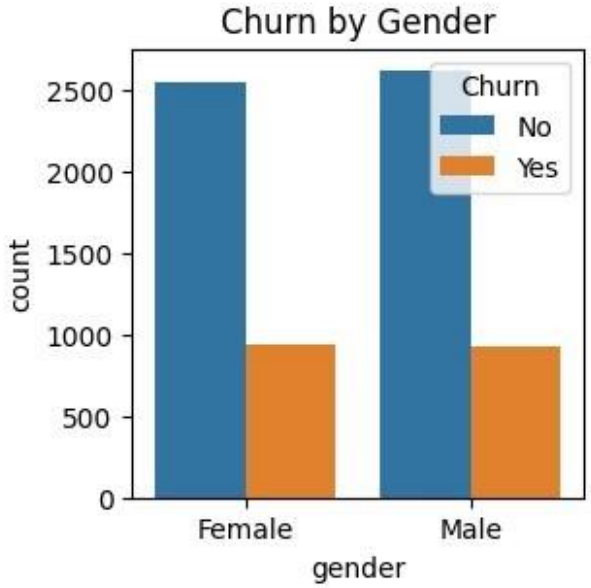
```
gb = df.groupby("Churn").agg({'Churn': "count"})
plt.pie(gb['Churn'], labels=gb.index, autopct="%1.2f%%")
plt.title("Percentage of Churned Customers", fontsize=10)
plt.show()
```



Shows the percentage of customers who have churned.

Churn by Gender:

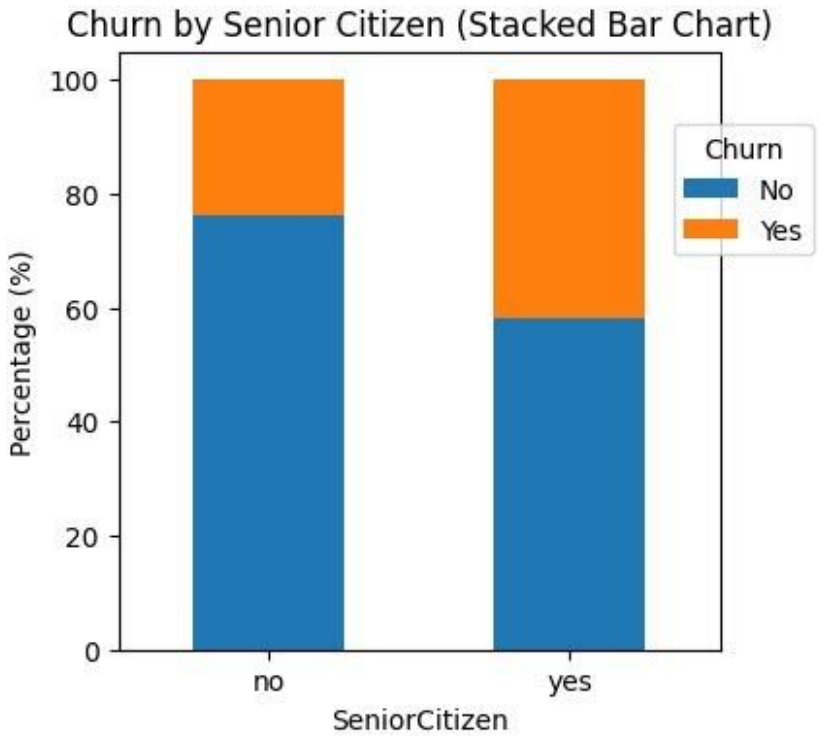
```
plt.figure(figsize=(3,3))
sns.countplot(x="gender", data=df, hue="Churn")
plt.title("Churn by Gender") plt.show()
```



Analyzes churn rates based on gender.

Churn by Senior Citizen Status

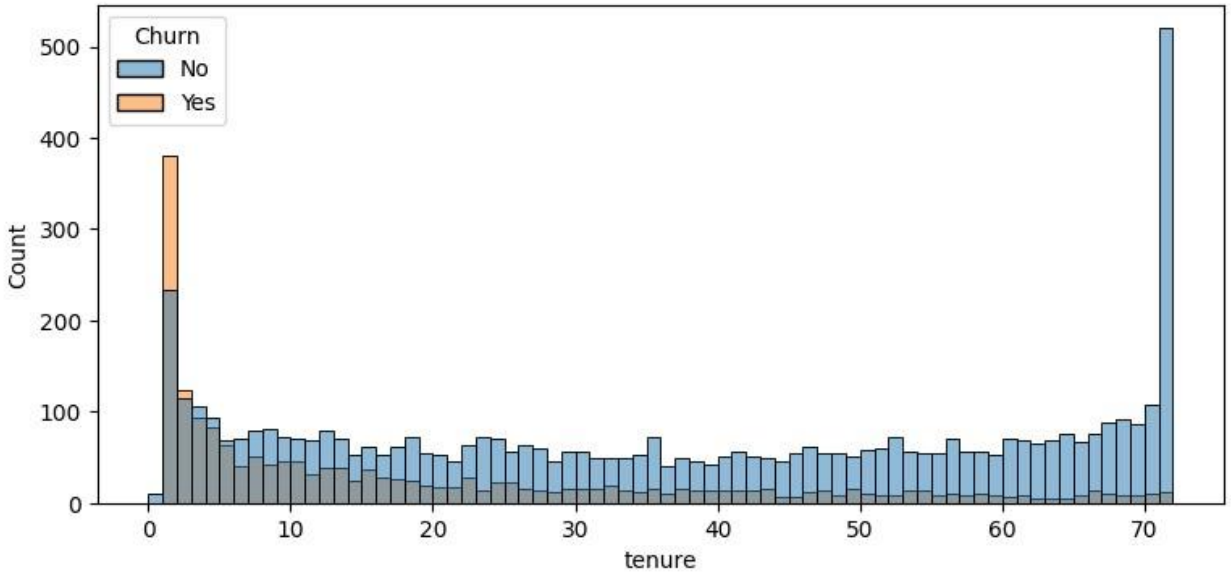
```
total_counts = df.groupby('SeniorCitizen')
['Churn'].value_counts(normalize=True).unstack() * 100
fig, ax = plt.subplots(figsize=(4, 4))
total_counts.plot(kind='bar', stacked=True, ax=ax, color=['#1f77b4',
'#ff7f0e'])
plt.title('Churn by Senior Citizen (Stacked Bar Chart)')
plt.xlabel('SeniorCitizen') plt.ylabel('Percentage (%)')
plt.xticks(rotation=0)
plt.legend(title='Churn', bbox_to_anchor=(0.9,0.9))
plt.show()
```



Displays a stacked bar chart showing the churn rate for senior citizens versus non-senior citizens.

Churn by Tenure

```
plt.figure(figsize=(9,4))
sns.histplot(x="tenure", data=df, bins=72, hue="Churn")
plt.show()
```



Visualizes the distribution of customer tenure with respect to churn. **Churn by**

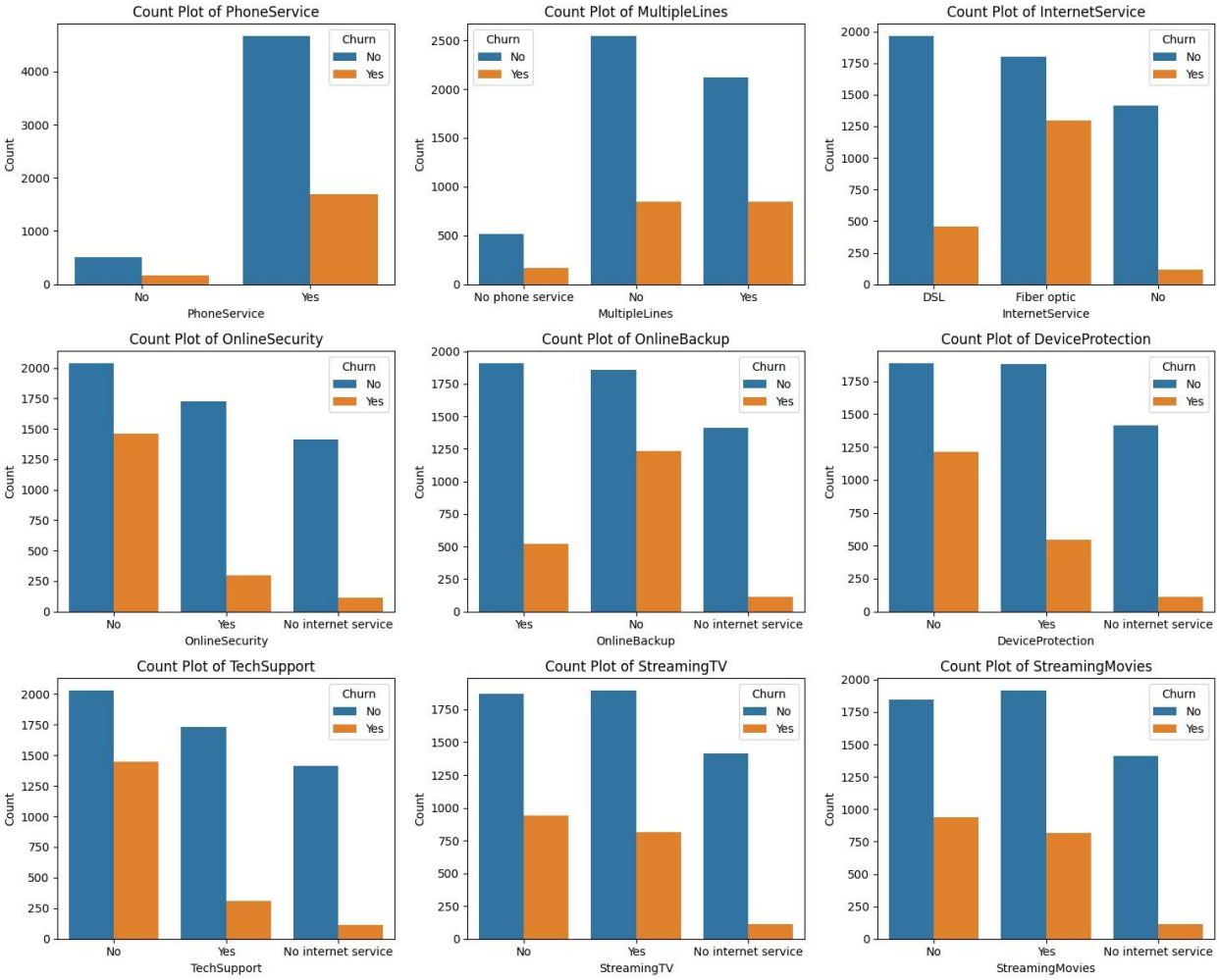
Contract Type

```
plt.figure(figsize=(4,4))
ax = sns.countplot(x="Contract", data=df, hue="Churn")
ax.bar_label(ax.containers[0]) plt.title("Count of
Customers by Contract") plt.show()
```

Analyzes the relationship between contract type and churn.

Churn by Service Usage

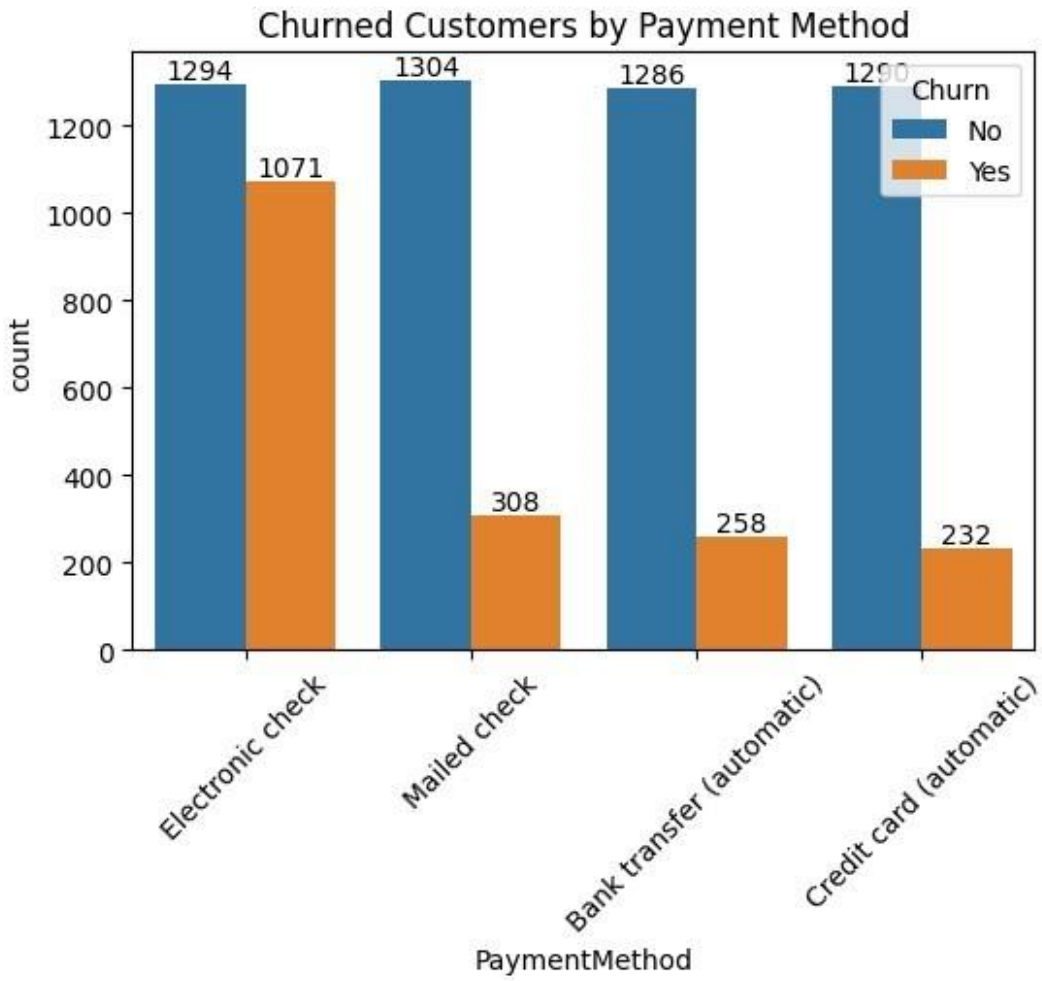
```
columns = ['PhoneService', 'MultipleLines', 'InternetService',
'OnlineSecurity',
           'OnlineBackup', 'DeviceProtection', 'TechSupport',
'StreamingTV', 'StreamingMovies']

n_cols = 3
n_rows = (len(columns) + n_cols - 1) // n_cols

fig, axes = plt.subplots(n_rows, n_cols, figsize=(15, n_rows * 4))
axes = axes.flatten()

for i, col in enumerate(columns):     sns.countplot(x=col,
data=df, ax=axes[i], hue=df["Churn"])
    axes[i].set_title(f'Count Plot of {col}')
axes[i].set_xlabel(col)
axes[i].set_ylabel('Count')

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

plt.tight_layout() plt.show()
```



Visualizes how various services used by customers relate to churn.

** Churn by Payment Method**

```
plt.figure(figsize=(6,4))
ax = sns.countplot(x="PaymentMethod", data=df, hue="Churn")
ax.bar_label(ax.containers[0])
ax.bar_label(ax.containers[1])
plt.title("Churned Customers by Payment Method")
plt.xticks(rotation=45) plt.show()
```

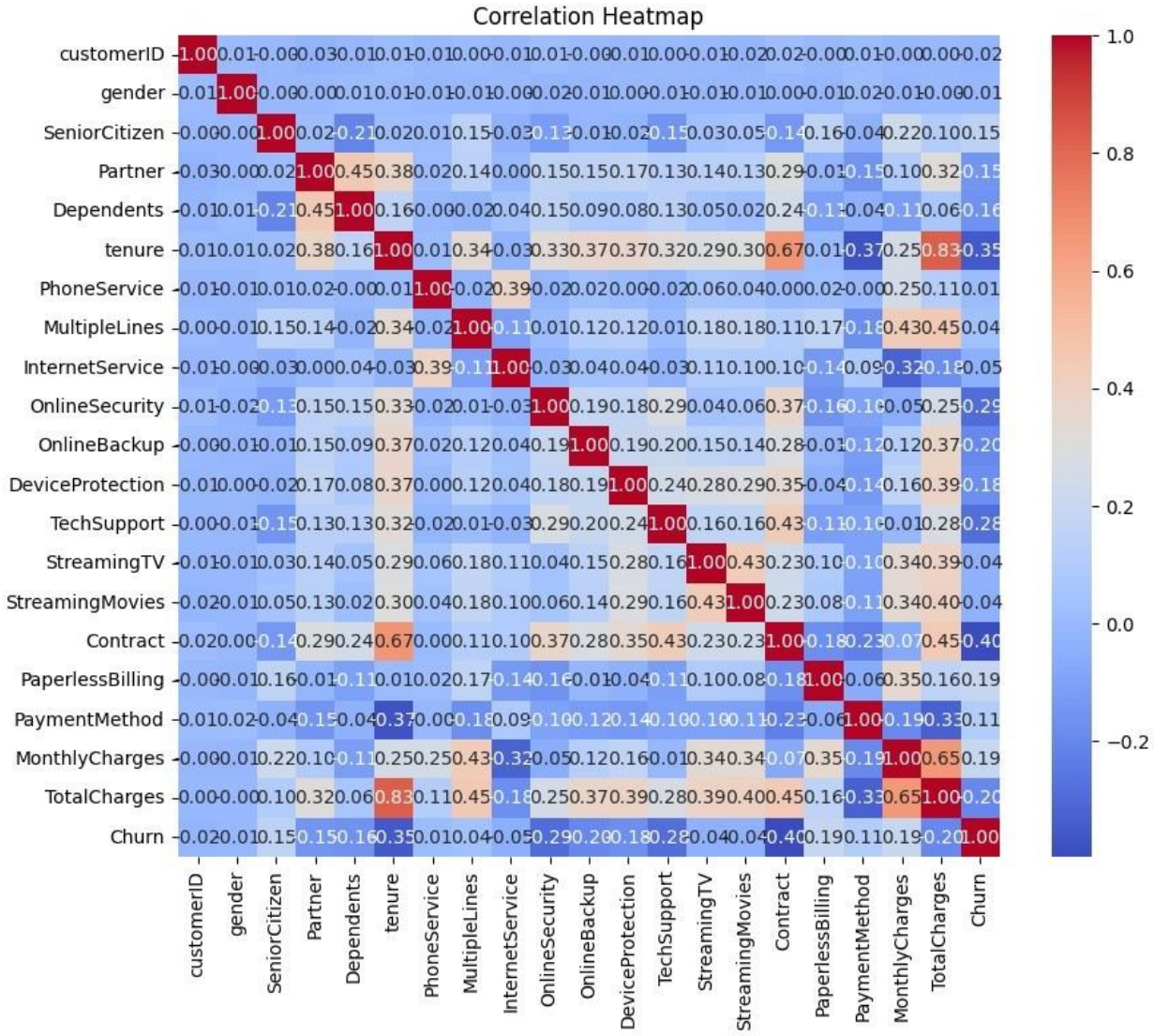


Analyzes the churn rate based on payment methods.

Correlation Analysis

```
df = pd.get_dummies(df, drop_first=True)  # drop_first to avoid
multicollinearity

import pandas as pd import
seaborn as sns import
matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder

# Sample DataFrame creation for demonstration (use your actual
DataFrame)
# df = pd.read_csv('your_data.csv')  # Load your data

# Encode categorical variables using Label Encoding or One-Hot
```

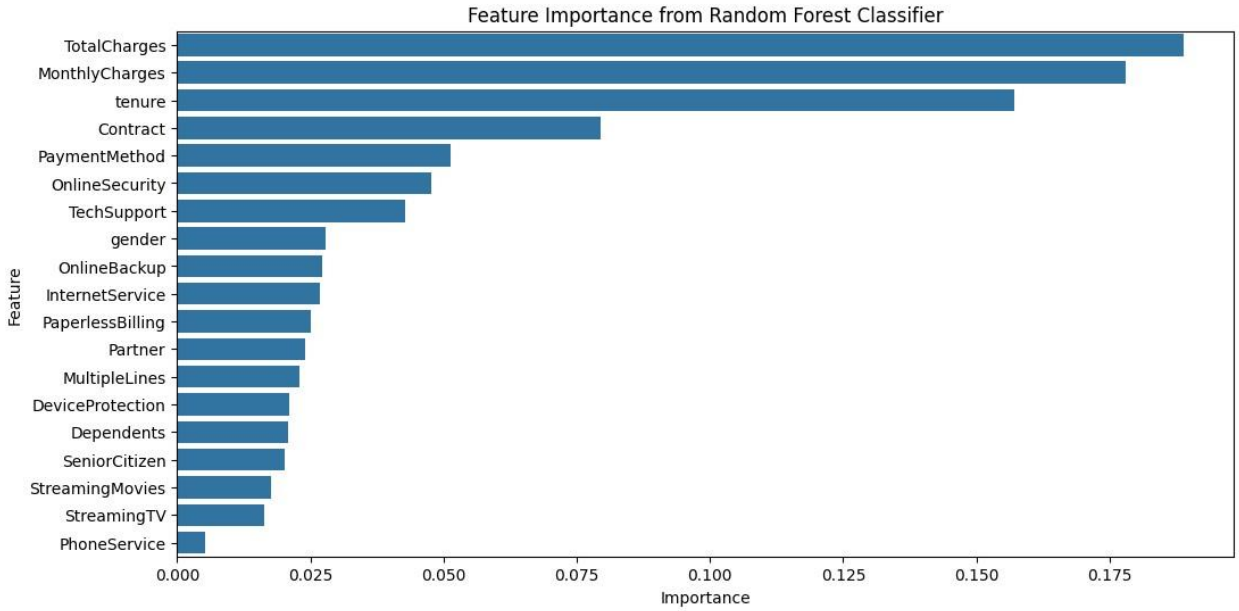
```
Encoding
label_encoders = {} for column in
df.select_dtypes(include=['object']).columns:     le =
LabelEncoder()
    df[column] = le.fit_transform(df[column])
label_encoders[column] = le

# Now calculate the correlation matrix
correlation_matrix = df.corr()

# Plot the heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f",
cmap='coolwarm', square=True) plt.title("Correlation
Heatmap") plt.show()
```



We will calculate the correlation matrix and visualize it using a heatmap to see how numerical features relate to each other and to the churn status.

Feature Importance Using Random Forest

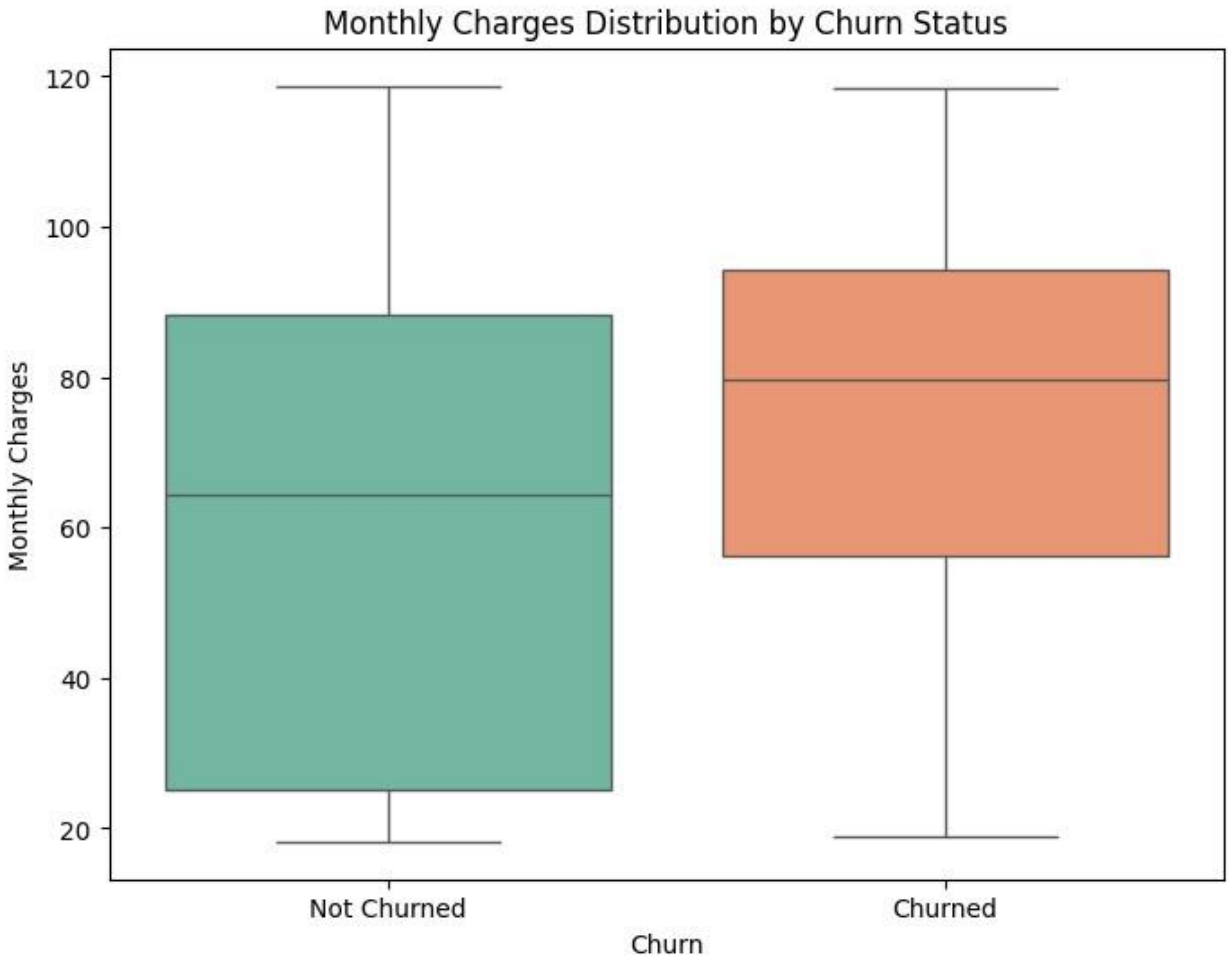
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder

# Encode categorical variables label_encoders = {} for
column in df.select_dtypes(include=['object']).columns:
le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
label_encoders[column] = le

# Split data into features and target variable
X = df.drop(columns=['customerID', 'Churn'])  # Dropping customerID
and Churn
y = df['Churn']  # Target variable

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Fit Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Get feature importance
importances = rf_model.feature_importances_ feature_names = X.columns
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':
importances}).sort_values(by='Importance', ascending=False)

# Plot feature importance
plt.figure(figsize=(12, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title("Feature Importance from Random Forest Classifier")
plt.show()
```



We will use a Random Forest classifier to determine which features are the most significant predictors of customer churn.

Monthly Charges vs. Churn Analysis

```
plt.figure(figsize=(8, 6))
sns.boxplot(x='Churn', y='MonthlyCharges', data=df, palette='Set2')
plt.title("Monthly Charges Distribution by Churn Status")
plt.xlabel("Churn") plt.ylabel("Monthly Charges")
plt.xticks([0, 1], ['Not Churned', 'Churned'])
plt.show() <ipython-input-23-bac2d49c0abd>:2:
FutureWarning:
Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.boxplot(x='Churn', y='MonthlyCharges', data=df, palette='Set2')
```



This analysis will include a box plot to visualize how monthly charges differ between customers who churned and those who did not.

Project Summary: Customer Churn Analysis

Objective The objective of the Customer Churn Analysis project is to identify and analyze the factors contributing to customer churn. By understanding these factors, businesses can develop strategies to retain customers, improve service offerings, and ultimately enhance customer satisfaction and profitability. Steps Involved

1. Data Collection Dataset Overview: The analysis begins with acquiring a dataset containing customer information, which typically includes: Customer demographics (age, gender, income) Account information (tenure, account type) Service usage details (monthly charges, total charges) Churn status (whether the customer has churned or not)
2. Data Preprocessing Data Cleaning: Handle missing values by either removing or imputing them. Remove duplicates if any exist. Data Transformation: Convert categorical variables into numerical formats (e.g., using one-hot encoding). Normalize or standardize numerical features to ensure they contribute equally to the analysis.
   Exploratory Data Analysis (EDA): Analyze the dataset to understand its structure, distributions, and relationships between variables.
3. Exploratory Data Analysis (EDA) Univariate Analysis: Examine the distribution of individual features (e.g., age, tenure) using histograms or box plots. Bivariate Analysis: Investigate the relationship between churn status and other variables through visualizations like bar charts and violin plots. Correlation Analysis: Calculate and visualize the correlation matrix to identify which features have the strongest relationships with churn.
4. Feature Engineering Creating New Features: Based on insights from EDA, create new features that may help in predicting churn (e.g., customer engagement metrics). Feature Selection: Select the most relevant features using techniques like correlation analysis or model-based feature importance.
5. Model Building Choosing a Model: Depending on the nature of the data, select appropriate models for classification (e.g., Logistic Regression, Decision Trees, Random Forest). Splitting Data: Divide the dataset into training and testing sets to evaluate model performance. Model Training: Train the chosen model using the training dataset.
6. Model Evaluation Performance Metrics: Assess model performance using metrics like accuracy, precision, recall, F1-score, and AUC-ROC. Confusion Matrix: Visualize the performance of the model to identify true positives, false positives, true negatives, and false negatives.
7. Insights Generation Identifying Key Factors: Analyze model outputs and feature importance scores to determine which factors most significantly influence customer churn. Customer Segmentation: Segment customers based on churn likelihood to tailor retention strategies.
8. Data Visualization Creating Visuals: Use visualizations to present insights effectively. Key visualizations may include: Correlation heatmaps Bar charts showing churn rates by demographic segments Pie charts illustrating the distribution of churn vs. non-churn customers ROC curve to evaluate model performance. Dashboards: Consider creating interactive dashboards using tools like Power BI or Tableau for stakeholders to explore insights dynamically.
9. Recommendations Based on the analysis, provide actionable recommendations to reduce churn, such as: Improving customer engagement through targeted marketing campaigns. Enhancing customer support services for high-risk segments. Offering personalized promotions based on usage patterns. Key Insights The analysis reveals that younger customers with lower tenure are more likely to churn. Customers with higher monthly charges and low service usage have a

higher churn rate. Specific demographic groups may be more susceptible to churn, indicating areas for focused retention efforts.

Conclusion

The Customer Churn Analysis provides businesses with valuable insights into the factors leading to customer attrition. By understanding these factors, companies can implement targeted strategies to improve customer retention, enhance satisfaction, and increase overall profitability. Author Information

Shaun Mia | LinkedIn