Customer Churn Prediction using Trained Model

We will use a telecommunications dataset for predicting customer churn. This is a historical customer dataset where each row represents one customer. The data is relatively easy to understand, and you may uncover insights you can use immediately. Typically it is less expensive to keep customers than acquire new ones, so the focus of this analysis is to predict the customers who will stay with the company.

This data set provides information to help you predict what behavior will help you to retain customers. You can analyze all relevant customer data and develop focused customer retention programs.

The dataset includes information about:

Customers who left within the last month – the column is called Churn

Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies

Customer account information – how long they had been a customer, contract, payment method, paperless billing, monthly charges, and total charges

Demographic info about customers – gender, age range, and if they have partners and dependents

Objectives:

Cleaning the Data.

Feature Analysis.

Label Encoding.

Feature Selection.

Hypothesis Generation and Testing.

Analyzing the Selected Features.

Comparing Classification Models.

Model Evaluation.

Making Predictions.

Table of Contents

Loading and Initial Data Preprocessing

Feature Analysis

TotalCharges and Tenure Analysis

Label Encoding for Categorical Features

Feature Selection

Feature-Specific Accuracy Calculation using Logistic Regression

Hypothesis Testing using Chi-Square Test

Proportion of Churned Customers by Categorical Variable

Visualizing Churn Analysis

Comparing Classification Models

Logistic Regression Model Evaluation and Confusion Matrix

Making Predictions with a Trained Model

Importing Libraries and Modules

Importing necessary libraries and modules required for data analysis, visualization, preprocessing, and machine learning..

Loading and Initial Data Preprocessing

Loading the dataset and performing initial preprocessing steps such as renaming columns, setting display options, dropping irrelevant columns, and handling missing data and categorical values.

```
df = pd.read_csv('Telco_Customer_Churn.csv')
```

```
df.columns = df.columns.str[0].str.upper() + df.columns.str[1:]

pd.set_option('display.max_columns', 21)

df.drop(columns='CustomerID', inplace=True)

df['MultipleLines'] = df['MultipleLines'].replace('No phone service', 'No')

df[['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']] = df[['OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV', 'StreamingMovies']].replace('No internet service', 'No')

df.head()
```

Gender SeniorCitizen Partner Dependents Tenure PhoneService MultipleLines InternetService OnlineSecurity OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies Contract PaperlessBilling PaymentMethod MonthlyCharges TotalCharges Churn

0 Female 0 Yes No 1 No No DSL No Yes No No No No Month-to-month Yes Electronic check 29.85 29.85 No

1 Male 0 No No 34 Yes No DSL Yes No Yes No No No One year No Mailed check 56.95 1889.5 No

2 Male 0 No No 2 Yes No DSL Yes Yes
No No No No Month-to-month Yes Mailed
check 53.85 108.15 Yes

3 Male 0 No No 45 No No DSL Yes No
Yes Yes No No One year No Bank transfer
(automatic) 42.30 1840.75 No

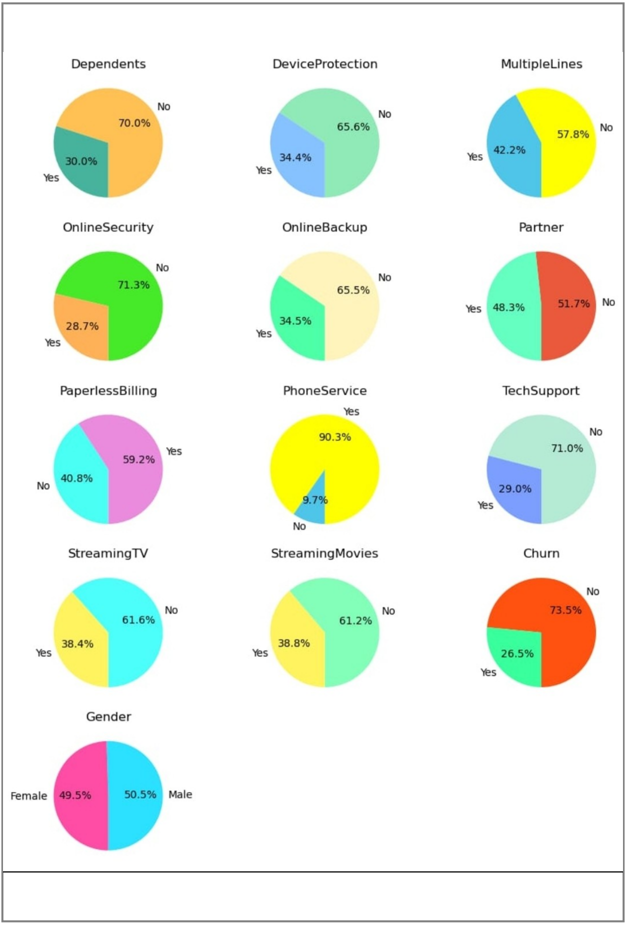
4 Female 0 No No 2 Yes No Fiber optic
No No No No No No Month-to-month Yes
Electronic check 70.70 151.65 Yes

df.shape

(7043, 20)

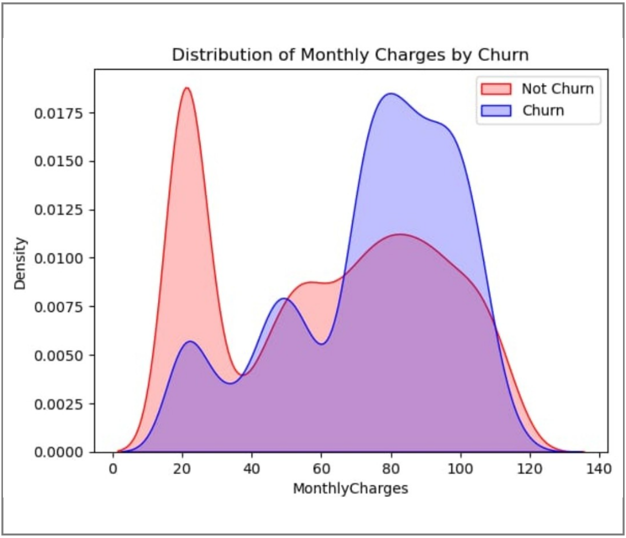
df.describe()

	SeniorCitizen	Tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000



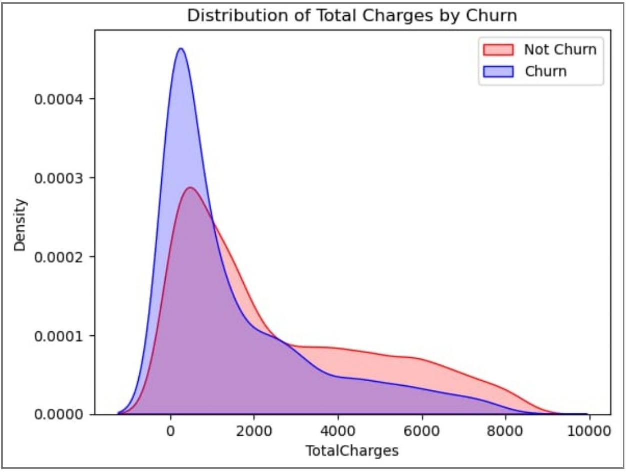
def plot_churn_distribution(data, column_name, title):

```
    ax = sns.kdeplot(data[column_name]
[(data["Churn"] == 'No')], color="Red", shade=True)

    ax = sns.kdeplot(data[column_name]
[(data["Churn"] == 'Yes')], ax=ax, color="Blue",
shade=True)

    ax.legend(["Not Churn", "Churn"], loc='upper right')

    ax.set_ylabel('Density')

    ax.set_xlabel(column_name)

    ax.set_title(title)

    plt.show()


# Example 1: Monthly Charges

plot_churn_distribution(df, '
```



Example 2: Total Charges

plot_churn_distribution(df, '



Label Encoding for Categorical Features

Performing label encoding for categorical features to convert them into numerical format for machine learning algorithms.

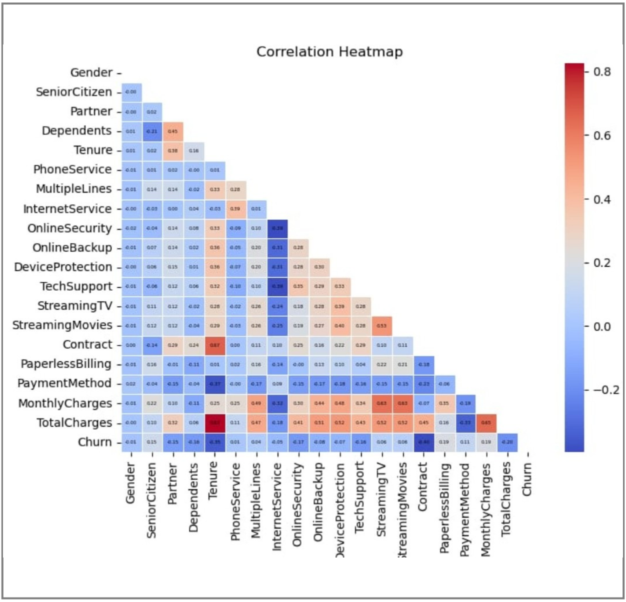
```
def label_encode_columns(df, columns_to_encode, label_mapping=None):
    if label_mapping is None:
        label_mapping = {}

    for column in columns_to_encode:
        le = preprocessing.LabelEncoder()
```

```
        unique_values = label_mapping.get(column, df[column].unique())
        df[column] = le.fit_transform(df[column])

    return df

# Columns to be encoded
columns_to_encode = ['Gender', 'Partner', 'Dependents', 'PhoneService', 'MultipleLines',
                     'InternetService', 'OnlineSecurity', 'OnlineBackup', 'TechSupport',
                     'DeviceProtection', 'StreamingTV', 'StreamingMovies', 'Contract',
                     'PaperlessBilling', 'PaymentMethod', 'Churn']

# Create a dictionary to map unique values to labels
label_mapping = {
    'Gender': ['Female', 'Male'],
    'Partner': ['No', 'Yes'],
    'Dependents': ['No', 'Yes'],
    'PhoneService': ['No', 'Yes'],
    'MultipleLines' : ['No', 'Yes'],
```

```
    'InternetService' : ['DSL', 'Fiber optic', 'No'],
    'OnlineSecurity' : ['No', 'Yes'],
    'OnlineBackup' : ['No', 'Yes'],
    'TechSupport' : ['No', 'Yes'],
    'DeviceProtection' : ['No', 'Yes',],
    'StreamingTV' : ['No', 'Yes'],
    'StreamingMovies' : ['No', 'Yes'],
    'Contract' : ['Month-to-month','Two year','One year'],
    'PaperlessBilling' :['No','Yes'],
    'PaymentMethod' : ['Electronic check','Mailed check','Bank transfer (automatic)','Credit card (automatic)'],
    'Churn': ['No', 'Yes']
}


df = label_encode_columns(df.copy(), columns_to_encode, label_mapping)
# 0-> No, 1 -> Yes
# 0-> Female, 1 -> Male
# 0->Month-to-month, 1->One year, 2->Two year
# 0->Bank transfer (automatic), 1->Credit card (automatic), 2->Electronic check, 3->Mailed check
```

```python
# DSL ->0, Fiber->1, No->2

# Assuming you already have the correlation matrix

correlation_matrix = df.corr()


# Create a mask for the upper triangle (including diagonal)

mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))


# Set the upper triangle values to NaN

correlation_matrix = correlation_matrix.mask(mask)


# Define annotation style

annot_font_size = 4

annot_style = {

    'fontsize': annot_font_size,

    'color': 'black', # You can customize the color if needed

    'verticalalignment': 'center',

    'horizontalalignment': 'center',

}
```

```
# Create a heatmap with custom annotation style

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

sns.heatmap(correlation_matrix, annot=True,
cmap="coolwarm", linewidths=.5, fmt=".2f",
annot_kws=annot_style)

plt.title("Correlation Heatmap")

plt.show()
```




```
df.describe()
```

Gender SeniorCitizen Partner Dependents Tenure PhoneService MultipleLines InternetService

OnlineSecurity OnlineBackup DeviceProtection TechSupport StreamingTV StreamingMovies Contract PaperlessBilling PaymentMethod MonthlyCharges TotalCharges Churn

count 7043.000000

mean 0.504756 0.162147 0.483033 0.299588 32.371149 0.903166 0.421837 0.872923 0.286668 0.344881 0.343888 0.290217 0.384353 0.387903 0.690473 0.592219 1.574329 64.761692 2281.916928 0.265370

std 0.500013 0.368612 0.499748 0.458110 24.559481 0.295752 0.493888 0.737796 0.452237 0.475363 0.475038 0.453895 0.486477 0.487307 0.833755 0.491457 1.068104 30.090047 2265.270398 0.441561

min 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 18.250000 18.800000 0.000000

25% 0.000000 0.000000 0.000000 0.000000 9.000000 1.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 1.000000 35.500000 402.225000 0.000000

50% 1.000000 0.000000 0.000000 0.000000 29.000000 1.000000 0.000000 1.000000 0.000000 0.000000

0.000000 0.000000 0.000000 0.000000 0.000000
1.000000 2.000000 70.350000 1397.475000 0.000000

75% 1.000000 0.000000 1.000000 1.000000 55.000000
1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 1.000000 1.000000 1.000000 1.000000
1.000000 2.000000 89.850000 3786.600000 1.000000

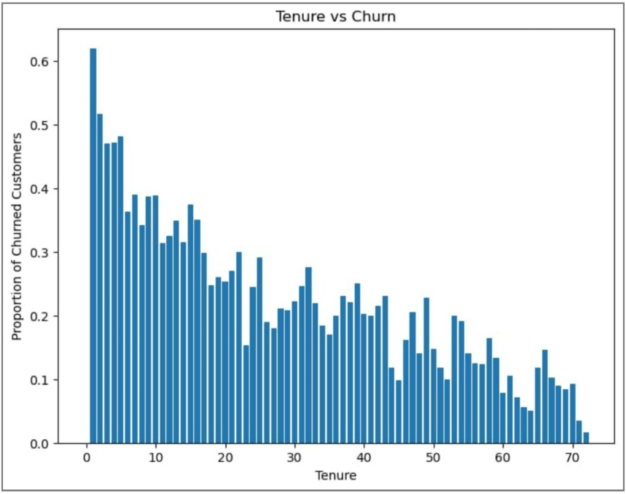
max 1.000000 1.000000

Proportion of Churned Customers

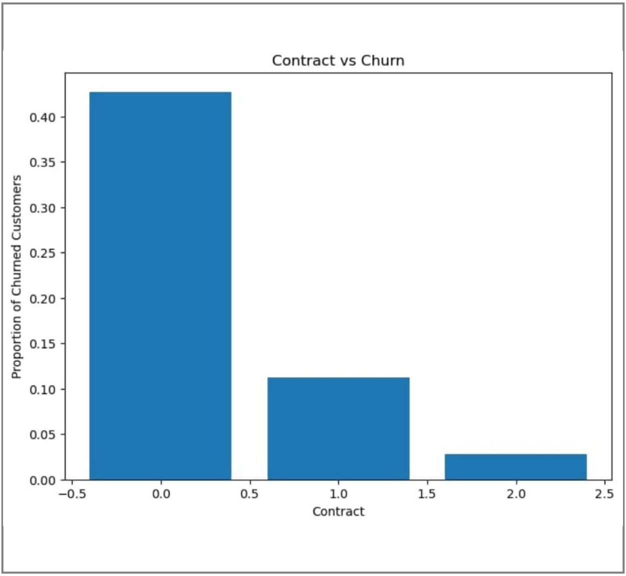
Visualizing the proportion of churned customers for each category of selected categorical variables using bar plots.

```
def plot_categorical_vs_churn(df, categorical_var, churn):
    # Create a contingency table
    contingency_table = pd.crosstab(df[categorical_var], df[churn])

    # Perform the Chi-Square test
    chi2, p_value, dof, expected = chi2_contingency(contingency_table)
```

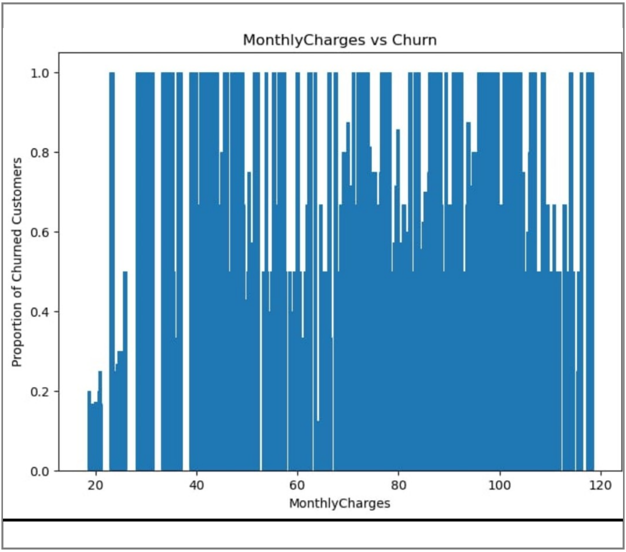
```
    # Calculate the proportion of churned customers for each category
    churn_proportion = contingency_table[1] / contingency_table.sum(axis=1)

    # Plot the bar plot
    plt.figure(figsize=(8, 6))
    plt.bar(churn_proportion.index, churn_proportion.values)
    plt.title(f"{categorical_var} vs Churn")
    plt.xlabel(categorical_var)
    plt.ylabel("Proportion of Churned Customers")
    plt.show()

# Select categorical variables and churn
categorical_vars = ['Tenure', 'Contract', 'MonthlyCharges', 'TotalCharges', 'OnlineSecurity']
churn = 'Churn'

# Plot each categorical variable against churn
for categorical_var in categorical_vars:
    plot_categorical_vs_churn
```

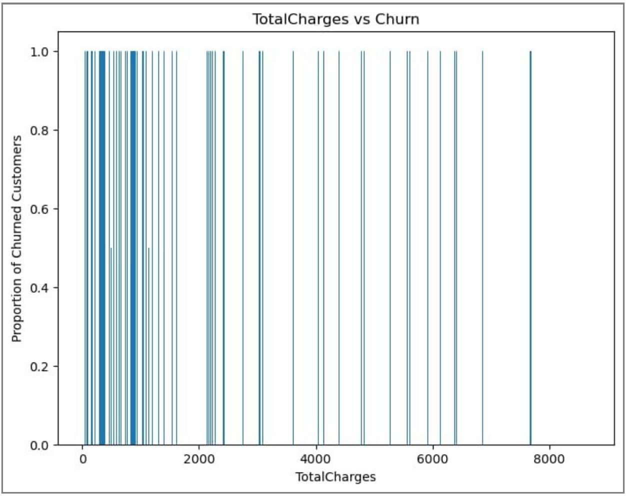


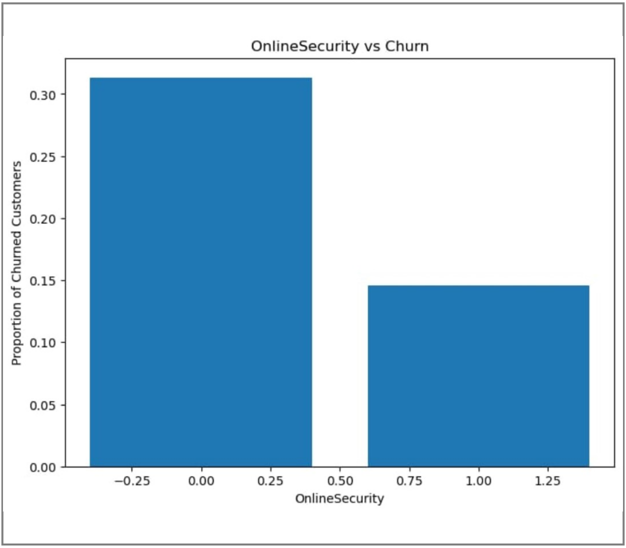
Tenure vs Churn
Proportion of Churned Customers
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
10
20
30
40
50
60
70
Tenure



Contract vs Churn
Proportion of Churned Customers
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
Contract


MonthlyCharges vs Churn
Proportion of Churned Customers
MonthlyCharges
0.0
0.2
0.4
0.6
0.8
1.0
20
40
60
80
100
120


TotalCharges vs Churn
Proportion of Churned Customers
TotalCharges
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000



Visualizing Churn Analysis

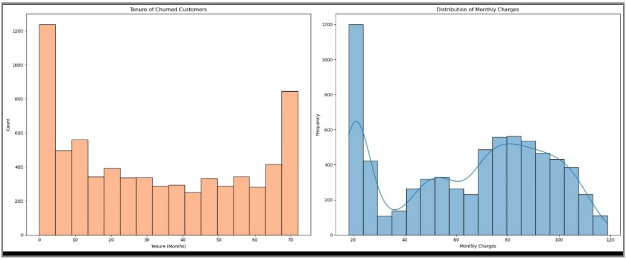
Creating various visualizations to analyze churn in the dataset, including histograms of tenure and monthly charges, as well as a count plot comparing contract types and churn.

Create a figure with subplots

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(20, 8))

Plot 1: Tenure of Churned Customers

```
sns.histplot(df['Churn'], x=df['Tenure'], bins='auto',
color='#ffa26e', ax=axes[0])
axes[0].set_xlabel("Tenure (Months)")
axes[0].set_ylabel("Count")
axes[0].set_title("Tenure of Churned Customers")

# Plot 2: Distribution of Monthly Charges
sns.histplot(df['MonthlyCharges'], bins='auto',
kde=True, ax=axes[1])
axes[1].set_xlabel("Monthly Charges")
axes[1].set_ylabel("Frequency")
axes[1].set_title("Distribution of Monthly Charges")

# Adjust spacing between subplots
plt.tight_layout()

# Display the combined plot
plt.show()
```




```
plt.figure(figsize=(20, 8))

ax = sns.countplot(x='Contract', hue='Churn', data=df)

ax.set_xticklabels(('Month-to-month', 'One-year', 'Two-years'))

ax.tick_params(axis='x', labelsize=10)


# Adding labels and title

plt.xlabel("Contract", size=15)

plt.ylabel("Count")

plt.title("Contract vs. Churn")


# Calculating the count for each category

total = len(df)

for p in ax.patches:

    count = p.get_height()

    x = p.get_x() + p.get_width() / 2
```

```
    y = p.get_height()
    ax.text(x, y, f'{count}', ha='center', va='bottom')

# Displaying the count plot
plt.legend(title="Churn", loc='upper right', labels=['No', 'Yes'])
plt.show()
```