Exploratory Data Analysis

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

 table = pd.read_csv("Dataset.csv");

print(table)
```

```
     State  Account length  Area code International plan Voice mail
plan  \
0       KS             128        415                 No
Yes
1       OH             107        415                 No
Yes
2       NJ             137        415                 No
No
3       OH              84        408                Yes
No
4       OK              75        415                Yes
No
...    ...             ...        ...                ...                .
..
2661    SC              79        415                 No
No
2662    AZ             192        415                 No
Yes
2663    WV              68        415                 No
No
2664    RI              28        510                 No
No
2665    TN              74        415                 No
Yes

      Number vmail messages  Total day minutes  Total day calls  \
0                        25              265.1              110
1                        26              161.6              123
2                         0              243.4              114
3                         0              299.4               71
4                         0              166.7              113
...                     ...                ...              ...
2661                      0              134.7               98
2662                     36              156.2               77
2663                      0              231.1               57
2664                      0              180.8              109
2665                     25              234.4              113
```

	Total day charge	Total eve minutes	Total eve calls	Total eve charge \
0	45.07	197.4	99	16.78
1	27.47	195.5	103	16.62
2	41.38	121.2	110	10.30
3	50.90	61.9	88	5.26
4	28.34	148.3	122	12.61
...	...	...	...	...
2661	22.90	189.7	68	16.12
2662	26.55	215.5	126	18.32
2663	39.29	153.4	55	13.04
2664	30.74	288.8	58	24.55
2665	39.85	265.9	82	22.60

	Total night minutes	Total night calls	Total night charge \
0	244.7	91	11.01
1	254.4	103	11.45
2	162.6	104	7.32
3	196.9	89	8.86
4	186.9	121	8.41
...	...	...	...
2661	221.4	128	9.96
2662	279.1	83	12.56
2663	191.3	123	8.61
2664	191.9	91	8.64
2665	241.4	77	10.86

	Total intl minutes	Total intl calls	Total intl charge \
0	10.0	3	2.70
1	13.7	3	3.70
2	12.2	5	3.29
3	6.6	7	1.78
4	10.1	3	2.73
...	...	...	...
2661	11.8	5	3.19
2662	9.9	6	2.67
2663	9.6	4	2.59
2664	14.1	6	3.81

```
2665                13.7                 4              3.70

      Customer service calls  Churn
0                          1  False
1                          1  False
2                          0  False
3                          2  False
4                          3  False
...                      ...    ...
2661                       2  False
2662                       2  False
2663                       3  False
2664                       2  False
2665                       0  False

[2666 rows x 20 columns]
```

```
print(table.head())
```

```
  State  Account length  Area code International plan Voice mail plan
\
0    KS             128        415                 No             Yes

1    OH             107        415                 No             Yes

2    NJ             137        415                 No              No

3    OH              84        408                Yes              No

4    OK              75        415                Yes              No


   Number vmail messages  Total day minutes  Total day calls  \
0                     25              265.1              110
1                     26              161.6              123
2                      0              243.4              114
3                      0              299.4               71
4                      0              166.7              113

   Total day charge  Total eve minutes  Total eve calls  Total eve
charge  \
0             45.07              197.4               99
16.78
1             27.47              195.5              103
16.62
2             41.38              121.2              110
10.30
3             50.90               61.9               88
5.26
4             28.34              148.3              122
12.61
```

```
   Total night minutes  Total night calls  Total night charge  \
0                244.7                 91               11.01   
1                254.4                103               11.45   
2                162.6                104                7.32   
3                196.9                 89                8.86   
4                186.9                121                8.41   

   Total intl minutes  Total intl calls  Total intl charge  \
0                10.0                 3               2.70   
1                13.7                 3               3.70   
2                12.2                 5               3.29   
3                 6.6                 7               1.78   
4                10.1                 3               2.73   

   Customer service calls  Churn  
0                       1  False  
1                       1  False  
2                       0  False  
3                       2  False  
4                       3  False  
```

```
# Overview of the dataset
print("\nData types and null values:")
print(table.info())

print("\nSummary statistics:")
print(table.describe())
```

```
Data types and null values:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2666 entries, 0 to 2665
Data columns (total 20 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   State                   2666 non-null   object 
 1   Account length          2666 non-null   int64  
 2   Area code               2666 non-null   int64  
 3   International plan      2666 non-null   object 
 4   Voice mail plan         2666 non-null   object 
 5   Number vmail messages   2666 non-null   int64  
 6   Total day minutes       2666 non-null   float64
 7   Total day calls         2666 non-null   int64  
 8   Total day charge        2666 non-null   float64
 9   Total eve minutes       2666 non-null   float64
 10  Total eve calls         2666 non-null   int64  
 11  Total eve charge        2666 non-null   float64
 12  Total night minutes     2666 non-null   float64
 13  Total night calls       2666 non-null   int64  
```

```
 14  Total night charge      2666 non-null   float64
 15  Total intl minutes      2666 non-null   float64
 16  Total intl calls        2666 non-null   int64
 17  Total intl charge       2666 non-null   float64
 18  Customer service calls  2666 non-null   int64
 19  Churn                   2666 non-null   bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 398.5+ KB
None

Summary statistics:
       Account length    Area code  Number vmail messages  Total day
minutes  \
count     2666.000000  2666.000000            2666.000000
2666.00000
mean       100.620405   437.438860               8.021755
179.48162
std         39.563974    42.521018              13.612277
54.21035
min          1.000000   408.000000               0.000000
0.00000
25%         73.000000   408.000000               0.000000
143.40000
50%        100.000000   415.000000               0.000000
179.95000
75%        127.000000   510.000000              19.000000
215.90000
max        243.000000   510.000000              50.000000
350.80000

       Total day calls  Total day charge  Total eve minutes  Total eve
calls  \
count      2666.000000       2666.000000        2666.000000
2666.000000
mean        100.310203         30.512404         200.386159
100.023631
std          19.988162          9.215733          50.951515
20.161445
min           0.000000          0.000000           0.000000
0.000000
25%          87.000000         24.380000         165.300000
87.000000
50%         101.000000         30.590000         200.900000
100.000000
75%         114.000000         36.700000         235.100000
114.000000
max         160.000000         59.640000         363.700000
170.000000

       Total eve charge  Total night minutes  Total night calls  \
```

```
count       2666.000000          2666.000000        2666.000000
mean          17.033072           201.168942         100.106152
std            4.330864            50.780323          19.418459
min            0.000000            43.700000          33.000000
25%           14.050000           166.925000          87.000000
50%           17.080000           201.150000         100.000000
75%           19.980000           236.475000         113.000000
max           30.910000           395.000000         166.000000

       Total night charge  Total intl minutes  Total intl calls  \
count         2666.000000         2666.000000       2666.000000
mean             9.052689           10.237022          4.467367
std              2.285120            2.788349          2.456195
min              1.970000            0.000000          0.000000
25%              7.512500            8.500000          3.000000
50%              9.050000           10.200000          4.000000
75%             10.640000           12.100000          6.000000
max             17.770000           20.000000         20.000000

       Total intl charge  Customer service calls
count        2666.000000             2666.000000
mean            2.764490                1.562641
std             0.752812                1.311236
min             0.000000                0.000000
25%             2.300000                1.000000
50%             2.750000                1.000000
75%             3.270000                2.000000
max             5.400000                9.000000
```

```python
# Check for missing values
print("\nMissing values:")
print(table.isnull().sum())
```

```
Missing values:
State                     0
Account length            0
Area code                 0
International plan        0
Voice mail plan           0
Number vmail messages     0
Total day minutes         0
Total day calls           0
Total day charge          0
Total eve minutes         0
Total eve calls           0
Total eve charge          0
Total night minutes       0
Total night calls         0
Total night charge        0
```

```
Total intl minutes        0
Total intl calls          0
Total intl charge         0
Customer service calls    0
Churn                     0
dtype: int64
```

```
# Check which columns are numerical
numerical_cols = table.select_dtypes(include=['number']).columns

# Fill missing values with median for numerical columns
for col in numerical_cols:
    if table[col].isnull().any():  # Only fill if there are missing
values
        median_value = table[col].median()
        table[col].fillna(median_value, inplace=True)


print(table.dtypes)
```

```
State                      object
Account length              int64
Area code                   int64
International plan         object
Voice mail plan            object
Number vmail messages       int64
Total day minutes         float64
Total day calls             int64
Total day charge          float64
Total eve minutes         float64
Total eve calls             int64
Total eve charge          float64
Total night minutes       float64
Total night calls           int64
Total night charge        float64
Total intl minutes        float64
Total intl calls            int64
Total intl charge         float64
Customer service calls      int64
Churn                        bool
dtype: object
```

```
#here we replace missing values in categorical columns
# Check which columns are categorical
categorical_cols = table.select_dtypes(include=['object']).columns

# Fill missing values with mode for categorical columns
for col in categorical_cols:
    if table[col].isnull().any():  # Only fill if there are missing
values
```

```
        mode_value = table[col].mode()[0]  # Get the mode value
        table[col].fillna(mode_value, inplace=True)

# Check if there are any missing values remaining
missing_values_after = table.isnull().sum()
print("\nMissing values after imputation:")
print(missing_values_after)

# Verify that there are no missing values left
if missing_values_after.sum() == 0:
    print("All missing values have been successfully filled.")
else:
    print("Some missing values are still present.")
```

```
Missing values after imputation:
State                     0
Account length            0
Area code                 0
International plan        0
Voice mail plan           0
Number vmail messages     0
Total day minutes         0
Total day calls           0
Total day charge          0
Total eve minutes         0
Total eve calls           0
Total eve charge          0
Total night minutes       0
Total night calls         0
Total night charge        0
Total intl minutes        0
Total intl calls          0
Total intl charge         0
Customer service calls    0
Churn                     0
dtype: int64
All missing values have been successfully filled.
```

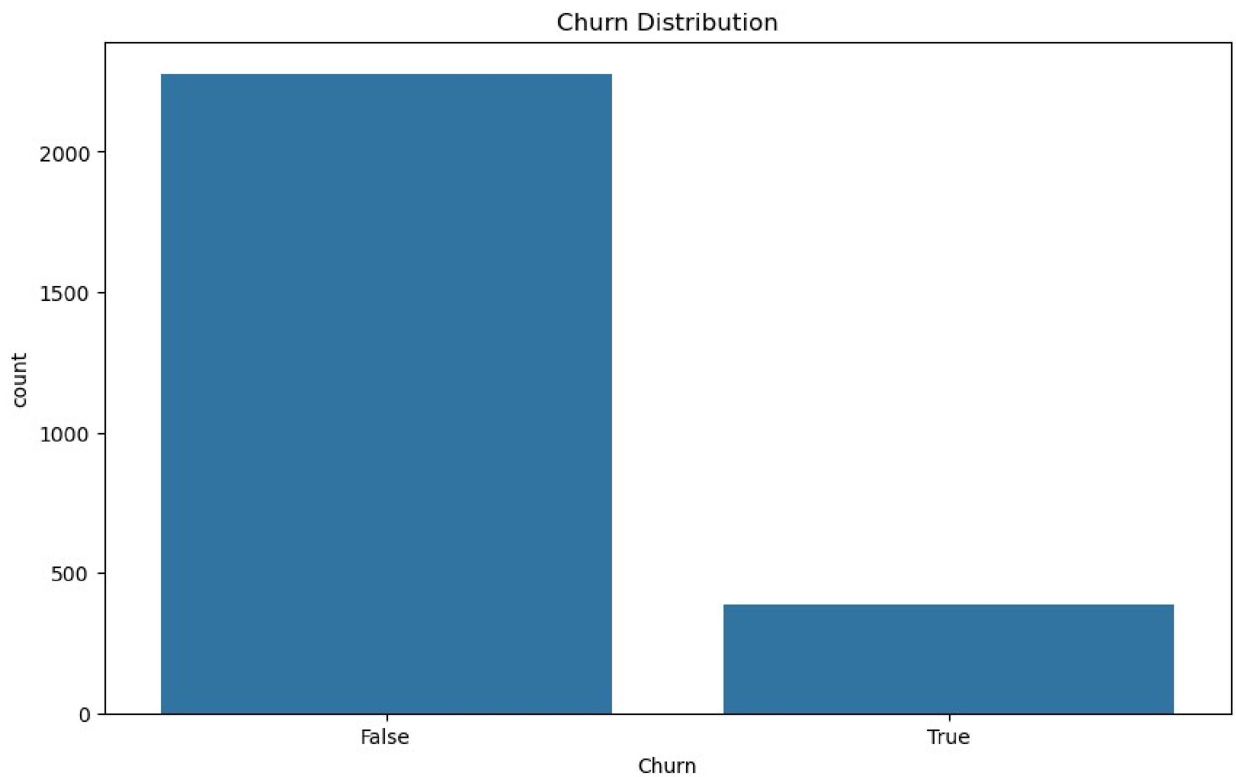
```
print("\nDuplicate rows:")
print(table.duplicated().sum())
```

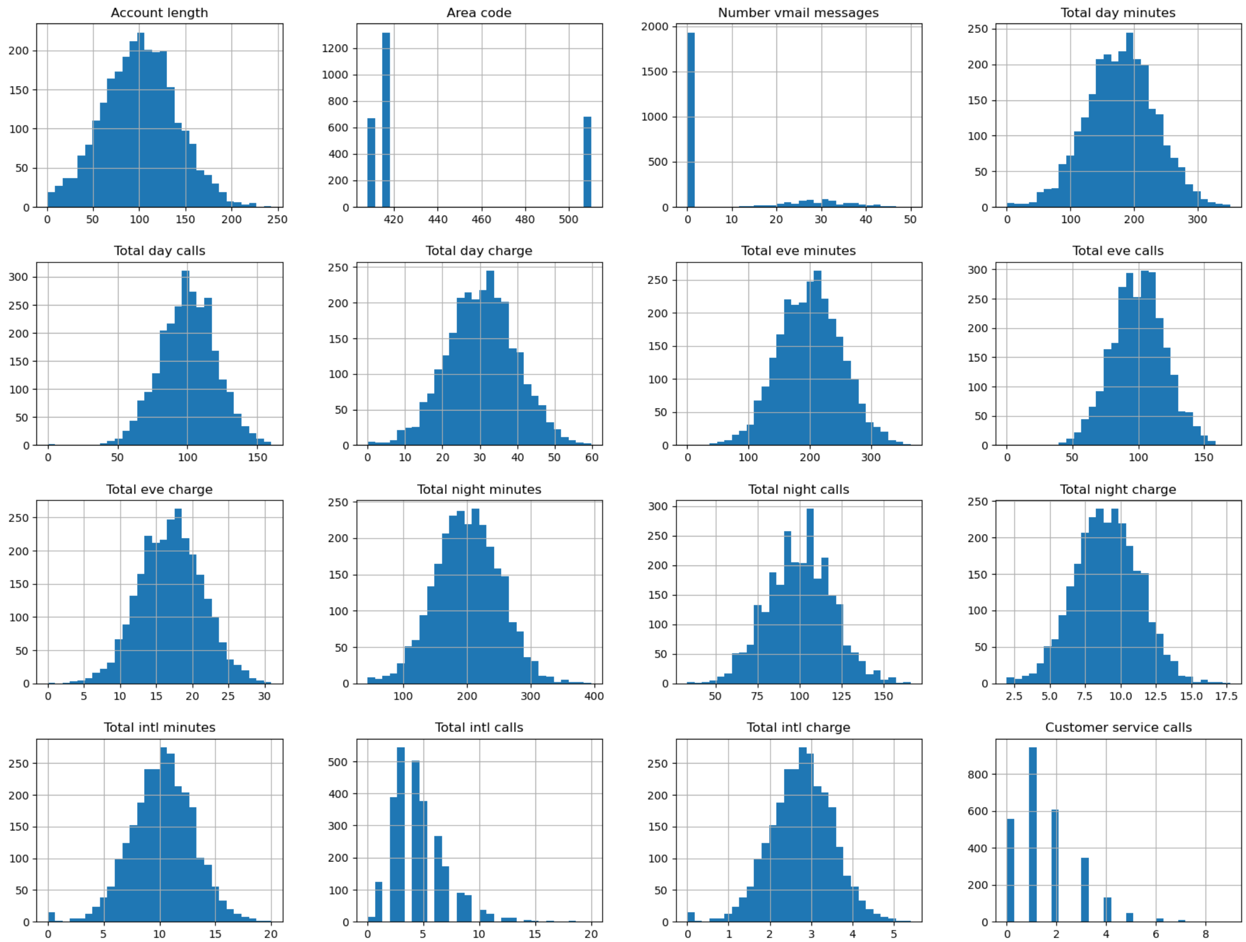
```
Duplicate rows:
0
```

```
table = table.drop_duplicates()
```

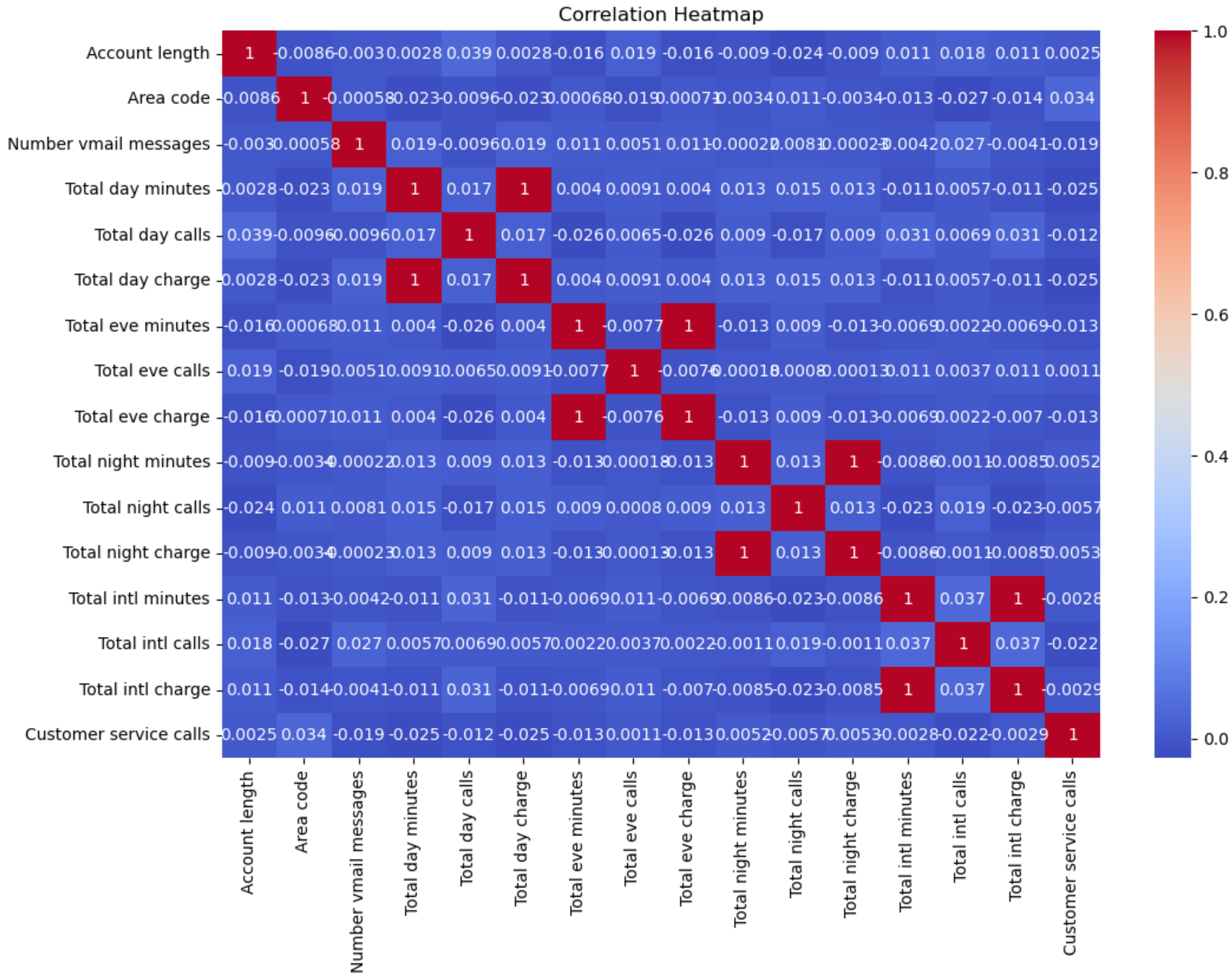
Univariate Analysis

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

# Distribution of the target variable
plt.figure(figsize=(10, 6))
sns.countplot(x='Churn', data=table)
plt.title('Churn Distribution')
plt.show()

# Distribution of numerical features
table.hist(bins=30, figsize=(20, 15))
plt.show()
```






```python
numeric_table = table.select_dtypes(include=[np.number])
#Bivariate Analysis


numeric_table = table.select_dtypes(include=[np.number])
plt.figure(figsize=(12, 8))
sns.heatmap(numeric_table.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```




```
print(table.columns)
table['MonthlyCharges'] = (table['Total day charge'] +
                           table['Total eve charge'] +
                           table['Total night charge'] +
                           table['Total intl charge']) /
table['Account length']


Index(['State', 'Account length', 'Area code', 'International plan',
       'Voice mail plan', 'Number vmail messages', 'Total day
minutes',
       'Total day calls', 'Total day charge', 'Total eve minutes',
       'Total eve calls', 'Total eve charge', 'Total night minutes',
       'Total night calls', 'Total night charge', 'Total intl
minutes',
       'Total intl calls', 'Total intl charge', 'Customer service
calls',
       'Churn'],
      dtype='object')
```

```python
# Interaction between day charge and international charge
table['Day_Intl_Charge_Interaction'] = table['Total day charge'] *
table['Total intl charge']

# Example: Total charges per call (sum of all charges divided by the
total number of calls)
table['TotalCharges_Per_Call'] = (table['Total day charge'] +
                                  table['Total eve charge'] +
                                  table['Total night charge'] +
                                  table['Total intl charge']) /
(table['Total day calls'] +

table['Total eve calls'] +

table['Total night calls'] +

table['Total intl calls'])

print(table.columns)

Index(['State', 'Account length', 'Area code', 'International plan',
       'Voice mail plan', 'Number vmail messages', 'Total day
minutes',
       'Total day calls', 'Total day charge', 'Total eve minutes',
       'Total eve calls', 'Total eve charge', 'Total night minutes',
       'Total night calls', 'Total night charge', 'Total intl
minutes',
       'Total intl calls', 'Total intl charge', 'Customer service
calls',
       'Churn', 'MonthlyCharges', 'Day_Intl_Charge_Interaction',
       'TotalCharges_Per_Call'],
      dtype='object')

# Example: Convert 'International plan' to binary (1 if 'yes', else 0)
table['International plan'] = table['International plan'].map({'yes':
1, 'no': 0})

# Example: Log transform total day minutes to reduce skewness
table['Log_Total_Day_Minutes'] = np.log1p(table['Total day minutes'])
# Using log1p to handle zero values

from sklearn.preprocessing import PolynomialFeatures

# Example: Polynomial features for 'Total day charge'
poly = PolynomialFeatures(degree=2, include_bias=False)
poly_features = poly.fit_transform(table[['Total day charge']])

# Adding polynomial features back to the DataFrame
poly_df = pd.DataFrame(poly_features, columns=['Total_Day_Charge',
'Total_Day_Charge^2'])
table = pd.concat([table, poly_df], axis=1)
```

```
# Example: One-hot encoding 'State' column
table = pd.get_dummies(table, columns=['State'], drop_first=True)

# Convert categorical variables to numerical format using one-hot
encoding
table_encoded = pd.get_dummies(table, drop_first=True)

# Calculate the correlation matrix on the encoded DataFrame
correlation_matrix = table_encoded.corr()

# Select only the numeric columns from the DataFrame
numeric_table = table.select_dtypes(include=[np.number])

# Calculate the correlation matrix on the numeric columns
correlation_matrix = numeric_table.corr()

high_corr_pairs = correlation_matrix[correlation_matrix >
0.8].stack().reset_index()
high_corr_pairs = high_corr_pairs[high_corr_pairs['level_0'] !=
high_corr_pairs['level_1']]
print(high_corr_pairs)
```

```
                 level_0               level_1         0
4      Total day minutes      Total day charge  1.000000
5      Total day minutes  Log_Total_Day_Minutes  0.908101
6      Total day minutes      Total_Day_Charge  1.000000
7      Total day minutes    Total_Day_Charge^2  0.977451
9       Total day charge     Total day minutes  1.000000
11      Total day charge  Log_Total_Day_Minutes  0.908101
12      Total day charge      Total_Day_Charge  1.000000
13      Total day charge    Total_Day_Charge^2  0.977452
15     Total eve minutes      Total eve charge  1.000000
17      Total eve charge     Total eve minutes  1.000000
20   Total night minutes    Total night charge  0.999999
22    Total night charge   Total night minutes  0.999999
25    Total intl minutes     Total intl charge  0.999993
27     Total intl charge    Total intl minutes  0.999993
33  Log_Total_Day_Minutes     Total day minutes  0.908101
34  Log_Total_Day_Minutes      Total day charge  0.908101
36  Log_Total_Day_Minutes      Total_Day_Charge  0.908101
37  Log_Total_Day_Minutes    Total_Day_Charge^2  0.822917
38      Total_Day_Charge     Total day minutes  1.000000
39      Total_Day_Charge      Total day charge  1.000000
40      Total_Day_Charge  Log_Total_Day_Minutes  0.908101
42      Total_Day_Charge    Total_Day_Charge^2  0.977452
43    Total_Day_Charge^2     Total day minutes  0.977451
44    Total_Day_Charge^2      Total day charge  0.977452
45    Total_Day_Charge^2  Log_Total_Day_Minutes  0.822917
46    Total_Day_Charge^2      Total_Day_Charge  0.977452
```

```python
# Identify pairs of features with high correlation
high_corr_pairs = correlation_matrix[correlation_matrix.abs() >
0.8].stack().reset_index()
high_corr_pairs = high_corr_pairs[high_corr_pairs['level_0'] !=
high_corr_pairs['level_1']]
high_corr_pairs.columns = ['Feature1', 'Feature2', 'Correlation']
print(high_corr_pairs)
```

```
                 Feature1                Feature2  Correlation
4       Total day minutes       Total day charge     1.000000
5       Total day minutes  Log_Total_Day_Minutes     0.908101
6       Total day minutes       Total_Day_Charge     1.000000
7       Total day minutes     Total_Day_Charge^2     0.977451
9        Total day charge      Total day minutes     1.000000
11       Total day charge  Log_Total_Day_Minutes     0.908101
12       Total day charge       Total_Day_Charge     1.000000
13       Total day charge     Total_Day_Charge^2     0.977452
15      Total eve minutes       Total eve charge     1.000000
17       Total eve charge      Total eve minutes     1.000000
20    Total night minutes     Total night charge     0.999999
22     Total night charge    Total night minutes     0.999999
25     Total intl minutes      Total intl charge     0.999993
27      Total intl charge     Total intl minutes     0.999993
33  Log_Total_Day_Minutes      Total day minutes     0.908101
34  Log_Total_Day_Minutes       Total day charge     0.908101
36  Log_Total_Day_Minutes       Total_Day_Charge     0.908101
37  Log_Total_Day_Minutes     Total_Day_Charge^2     0.822917
38       Total_Day_Charge      Total day minutes     1.000000
39       Total_Day_Charge       Total day charge     1.000000
40       Total_Day_Charge  Log_Total_Day_Minutes     0.908101
42       Total_Day_Charge     Total_Day_Charge^2     0.977452
43     Total_Day_Charge^2      Total day minutes     0.977451
44     Total_Day_Charge^2       Total day charge     0.977452
45     Total_Day_Charge^2  Log_Total_Day_Minutes     0.822917
46     Total_Day_Charge^2       Total_Day_Charge     0.977452
```

```python
# Remove highly correlated features
# Example: Drop one feature from each pair with high correlation
features_to_drop = set()

for feature1, feature2, corr in
high_corr_pairs.itertuples(index=False):
    if feature1 not in features_to_drop:
        features_to_drop.add(feature2)

table_reduced = table.drop(columns=features_to_drop)

print(table.columns)
```

```
Index(['Account length', 'Area code', 'International plan', 'Voice
mail plan',
       'Number vmail messages', 'Total day minutes', 'Total day
calls',
       'Total day charge', 'Total eve minutes', 'Total eve calls',
       'Total eve charge', 'Total night minutes', 'Total night calls',
       'Total night charge', 'Total intl minutes', 'Total intl calls',
       'Total intl charge', 'Customer service calls', 'Churn',
       'MonthlyCharges', 'Day_Intl_Charge_Interaction',
       'TotalCharges_Per_Call', 'Log_Total_Day_Minutes',
'Total_Day_Charge',
       'Total_Day_Charge^2', 'State_AL', 'State_AR', 'State_AZ',
'State_CA',
       'State_CO', 'State_CT', 'State_DC', 'State_DE', 'State_FL',
'State_GA',
       'State_HI', 'State_IA', 'State_ID', 'State_IL', 'State_IN',
'State_KS',
       'State_KY', 'State_LA', 'State_MA', 'State_MD', 'State_ME',
'State_MI',
       'State_MN', 'State_MO', 'State_MS', 'State_MT', 'State_NC',
'State_ND',
       'State_NE', 'State_NH', 'State_NJ', 'State_NM', 'State_NV',
'State_NY',
       'State_OH', 'State_OK', 'State_OR', 'State_PA', 'State_RI',
'State_SC',
       'State_SD', 'State_TN', 'State_TX', 'State_UT', 'State_VA',
'State_VT',
       'State_WA', 'State_WI', 'State_WV', 'State_WY'],
      dtype='object')
```

```
print(table.describe)
```

```
<bound method NDFrame.describe of       Account length  Area code
International plan Voice mail plan  \
0                 128        415                NaN            Yes
1                 107        415                NaN            Yes
2                 137        415                NaN             No
3                  84        408                NaN             No
4                  75        415                NaN             No
...               ...        ...                ...            ...
2661               79        415                NaN             No
2662              192        415                NaN            Yes
2663               68        415                NaN             No
2664               28        510                NaN             No
2665               74        415                NaN            Yes

      Number vmail messages  Total day minutes  Total day calls  \
0                        25              265.1              110
1                        26              161.6              123
2                         0              243.4              114
```

```
3                              0               299.4               71
4                              0               166.7              113
...                          ...                 ...              ...
2661                           0               134.7               98
2662                          36               156.2               77
2663                           0               231.1               57
2664                           0               180.8              109
2665                          25               234.4              113

      Total day charge  Total eve minutes  Total eve calls  ...
State_SD  \
0                45.07              197.4               99  ...
False
1                27.47              195.5              103  ...
False
2                41.38              121.2              110  ...
False
3                50.90               61.9               88  ...
False
4                28.34              148.3              122  ...
False
...                ...                ...              ...  ...
...
2661             22.90              189.7               68  ...
False
2662             26.55              215.5              126  ...
False
2663             39.29              153.4               55  ...
False
2664             30.74              288.8               58  ...
False
2665             39.85              265.9               82  ...
False

      State_TN  State_TX  State_UT  State_VA  State_VT  State_WA
State_WI  \
0        False     False     False     False     False     False
False
1        False     False     False     False     False     False
False
2        False     False     False     False     False     False
False
3        False     False     False     False     False     False
False
4        False     False     False     False     False     False
False
...        ...       ...       ...       ...       ...       ...
...
2661     False     False     False     False     False     False
```

```
False
2662     False     False     False     False     False     False
False
2663     False     False     False     False     False     False
False
2664     False     False     False     False     False     False
False
2665      True     False     False     False     False     False
False

      State_WV  State_WY
0        False     False
1        False     False
2        False     False
3        False     False
4        False     False
...        ...       ...
2661     False     False
2662     False     False
2663      True     False
2664     False     False
2665     False     False

[2666 rows x 75 columns]>
```

```
# Convert categorical variables to numerical format using one-hot
encoding
table_encoded = pd.get_dummies(table, drop_first=True)

# Calculate the correlation matrix on the encoded DataFrame
correlation_matrix = table_encoded.corr()

# Select numeric columns only
numeric_table = table_reduced.select_dtypes(include=[float, int])

# Calculate the correlation matrix for numeric columns
new_correlation_matrix = numeric_table.corr()

# Display the new correlation matrix
print(new_correlation_matrix)
```

```
                             Account length  Area code  International
plan  \
Account length                     1.000000  -0.008620
NaN
Area code                         -0.008620   1.000000
NaN
International plan                      NaN        NaN
NaN
Number vmail messages             -0.002996  -0.000584
NaN
```

```
Total day minutes                     0.002847  -0.023134
NaN
Total day calls                       0.038862  -0.009629
NaN
Total eve minutes                    -0.015923   0.000679
NaN
Total eve calls                       0.018552  -0.018602
NaN
Total night minutes                  -0.008994  -0.003353
NaN
Total night calls                    -0.024007   0.011455
NaN
Total intl minutes                    0.011369  -0.013418
NaN
Total intl calls                      0.017627  -0.027423
NaN
Customer service calls                0.002455   0.034442
NaN
MonthlyCharges                       -0.295137  -0.021622
NaN
Day_Intl_Charge_Interaction           0.012564  -0.020865
NaN
TotalCharges_Per_Call                -0.018291  -0.011560
NaN

                             Number vmail messages  Total day minutes  \
Account length                           -0.002996           0.002847

Area code                                -0.000584          -0.023134

International plan                             NaN                NaN

Number vmail messages                     1.000000           0.019027

Total day minutes                         0.019027           1.000000

Total day calls                          -0.009622           0.016780

Total eve minutes                         0.011401           0.003999

Total eve calls                           0.005131           0.009059

Total night minutes                      -0.000224           0.013491

Total night calls                         0.008124           0.015054

Total intl minutes                       -0.004156          -0.011042

Total intl calls                          0.027013           0.005687
```

Customer service calls	-0.018787	-0.024543
MonthlyCharges	0.015001	0.031773
Day_Intl_Charge_Interaction	0.016413	0.726057
TotalCharges_Per_Call	0.015363	0.724796

	Total day calls	Total eve minutes \
Account length	0.038862	-0.015923
Area code	-0.009629	0.000679
International plan	NaN	NaN
Number vmail messages	-0.009622	0.011401
Total day minutes	0.016780	0.003999
Total day calls	1.000000	-0.026003
Total eve minutes	-0.026003	1.000000
Total eve calls	0.006473	-0.007654
Total night minutes	0.008986	-0.013414
Total night calls	-0.016776	0.009017
Total intl minutes	0.031036	-0.006915
Total intl calls	0.006928	0.002160
Customer service calls	-0.011945	-0.013192
MonthlyCharges	-0.006308	0.036526
Day_Intl_Charge_Interaction	0.028860	-0.000182
TotalCharges_Per_Call	-0.309617	0.351571

	Total eve calls	Total night minutes \
Account length	0.018552	-0.008994
Area code	-0.018602	-0.003353
International plan	NaN	NaN
Number vmail messages	0.005131	-0.000224
Total day minutes	0.009059	0.013491
Total day calls	0.006473	0.008986
Total eve minutes	-0.007654	-0.013414
Total eve calls	1.000000	-0.000175
Total night minutes	-0.000175	1.000000
Total night calls	0.000797	0.012736
Total intl minutes	0.011012	-0.008607
Total intl calls	0.003710	-0.001110
Customer service calls	0.001058	0.005236
MonthlyCharges	-0.029454	-0.001365
Day_Intl_Charge_Interaction	0.015680	0.002490
TotalCharges_Per_Call	-0.320093	0.181503

	Total night calls	Total intl minutes \
Account length	-0.024007	0.011369
Area code	0.011455	-0.013418
International plan	NaN	NaN

```
Number vmail messages                      0.008124           -0.004156
Total day minutes                          0.015054           -0.011042
Total day calls                           -0.016776            0.031036
Total eve minutes                          0.009017           -0.006915
Total eve calls                            0.000797            0.011012
Total night minutes                        0.012736           -0.008607
Total night calls                          1.000000           -0.023447
Total intl minutes                        -0.023447            1.000000
Total intl calls                           0.019367            0.037315
Customer service calls                    -0.005677           -0.002826
MonthlyCharges                             0.006215           -0.040017
Day_Intl_Charge_Interaction               -0.002598            0.650203
TotalCharges_Per_Call                     -0.289248            0.040373

                             Total intl calls  Customer service calls
\
Account length                       0.017627                0.002455

Area code                           -0.027423                0.034442

International plan                        NaN                     NaN

Number vmail messages                0.027013               -0.018787

Total day minutes                    0.005687               -0.024543

Total day calls                      0.006928               -0.011945

Total eve minutes                    0.002160               -0.013192

Total eve calls                      0.003710                0.001058

Total night minutes                 -0.001110                0.005236

Total night calls                    0.019367               -0.005677

Total intl minutes                   0.037315               -0.002826

Total intl calls                     1.000000               -0.022143

Customer service calls              -0.022143                1.000000

MonthlyCharges                      -0.027235               -0.003609

Day_Intl_Charge_Interaction          0.036489               -0.018071

TotalCharges_Per_Call               -0.040077               -0.017781


                             MonthlyCharges
Day_Intl_Charge_Interaction  \
```

```
Account length                   -0.295137
0.012564
Area code                        -0.021622                      -
0.020865
International plan                     NaN
NaN
Number vmail messages             0.015001
0.016413
Total day minutes                 0.031773
0.726057
Total day calls                  -0.006308
0.028860
Total eve minutes                 0.036526                      -
0.000182
Total eve calls                  -0.029454
0.015680
Total night minutes              -0.001365
0.002490
Total night calls                 0.006215                      -
0.002598
Total intl minutes               -0.040017
0.650203
Total intl calls                 -0.027235
0.036489
Customer service calls           -0.003609                      -
0.018071
MonthlyCharges                    1.000000                      -
0.001303
Day_Intl_Charge_Interaction      -0.001303
1.000000
TotalCharges_Per_Call             0.043313
0.556862

                             TotalCharges_Per_Call
Account length                           -0.018291
Area code                                -0.011560
International plan                             NaN
Number vmail messages                     0.015363
Total day minutes                         0.724796
Total day calls                          -0.309617
Total eve minutes                         0.351571
Total eve calls                          -0.320093
Total night minutes                       0.181503
Total night calls                        -0.289248
Total intl minutes                        0.040373
Total intl calls                         -0.040077
Customer service calls                   -0.017781
MonthlyCharges                            0.043313
```

```
Day_Intl_Charge_Interaction                    0.556862
TotalCharges_Per_Call                          1.000000
```

```
# Convert categorical variables to numeric (if not already done)
table_reduced['International plan'] = table_reduced['International
plan'].map({'yes': 1, 'no': 0})
table_reduced['Voice mail plan'] = table_reduced['Voice mail
plan'].map({'yes': 1, 'no': 0})

# Check data types of columns
print(table_reduced.dtypes)
```

```
Account length              int64
Area code                   int64
International plan        float64
Voice mail plan           float64
Number vmail messages       int64
                          ...
State_VT                     bool
State_WA                     bool
State_WI                     bool
State_WV                     bool
State_WY                     bool
Length: 68, dtype: object
```

```
# Fill NaNs with a default value, e.g., 0
table_reduced = table_reduced.fillna(0)

# Convert boolean columns to numeric (0 and 1)
table_reduced = table_reduced.astype(int)

# Select only numeric columns (optional, as all columns are now
numeric)
numeric_table = table_reduced.select_dtypes(include=[float, int])

# Calculate the correlation matrix
new_correlation_matrix = numeric_table.corr()

# Display the new correlation matrix
print(new_correlation_matrix)
```

```
                       Account length  Area code  International
plan  \
Account length               1.000000  -0.008620                 NaN

Area code                   -0.008620   1.000000                 NaN

International plan                NaN        NaN                 NaN

Voice mail plan                   NaN        NaN                 NaN
```

```
Number vmail messages      -0.002996  -0.000584                 NaN

...                              ...        ...                 ...

State_VT                    0.012432   0.008722                 NaN

State_WA                    0.008645  -0.004185                 NaN

State_WI                   -0.014259  -0.003704                 NaN

State_WV                   -0.028746   0.023971                 NaN

State_WY                    0.019166  -0.003746                 NaN


                       Voice mail plan  Number vmail messages  \
Account length                     NaN              -0.002996
Area code                          NaN              -0.000584
International plan                 NaN                    NaN
Voice mail plan                    NaN                    NaN
Number vmail messages              NaN               1.000000
...                                ...                    ...
State_VT                           NaN              -0.016241
State_WA                           NaN              -0.038355
State_WI                           NaN               0.001967
State_WV                           NaN               0.013126
State_WY                           NaN              -0.011607

                       Total day minutes  Total day calls  Total eve
minutes  \
Account length                  0.002840         0.038862           -
0.015735
Area code                      -0.023115        -0.009629
0.000768
International plan                   NaN              NaN
NaN
Voice mail plan                      NaN              NaN
NaN
Number vmail messages           0.019007        -0.009622
0.011484
...                                  ...              ...
...
State_VT                        0.009839         0.005361
0.005533
State_WA                       -0.010045         0.001427
0.009697
State_WI                       -0.010438        -0.025347           -
0.007930
State_WV                       -0.025475         0.038735           -
```

```
0.052270
State_WY                        0.001393         0.008157
0.030341

                       Total eve calls  Total night minutes  ...
State_SD  \
Account length                0.018552            -0.008946  ...
0.021433
Area code                    -0.018602            -0.003247  ...
0.016323
International plan                 NaN                  NaN  ...
NaN
Voice mail plan                    NaN                  NaN  ...
NaN
Number vmail messages         0.005131            -0.000281  ...
0.013529
...                                ...                  ...  ...
...
State_VT                     -0.022300             0.017478  ... -
0.020225
State_WA                     -0.028707            -0.005577  ... -
0.018528
State_WI                     -0.007149            -0.008582  ... -
0.020939
State_WV                     -0.006049            -0.005997  ... -
0.025281
State_WY                     -0.006773            -0.009094  ... -
0.021801

                       State_TN  State_TX  State_UT  State_VA
State_VT  \
Account length        -0.032553 -0.030161 -0.002763  0.021230
0.012432
Area code             -0.002366 -0.018258  0.011402 -0.002390
0.008722
International plan          NaN       NaN       NaN       NaN
NaN
Voice mail plan             NaN       NaN       NaN       NaN
NaN
Number vmail messages -0.006023 -0.018458  0.012765 -0.016104 -
0.016241
...                         ...       ...       ...       ...       ..
.
State_VT              -0.018473 -0.021453 -0.022428 -0.023732
1.000000
State_WA              -0.016922 -0.019652 -0.020546 -0.021741 -
0.020014
State_WI              -0.019124 -0.022210 -0.023219 -0.024569 -
0.022618
```

```
State_WV                -0.023090 -0.026815 -0.028034 -0.029664 -
0.027309
State_WY                -0.019912 -0.023124 -0.024175 -0.025581 -
0.023550

                         State_WA  State_WI  State_WV  State_WY
Account length           0.008645 -0.014259 -0.028746  0.019166
Area code               -0.004185 -0.003704  0.023971 -0.003746
International plan            NaN       NaN       NaN       NaN
Voice mail plan               NaN       NaN       NaN       NaN
Number vmail messages   -0.038355  0.001967  0.013126 -0.011607
...                           ...       ...       ...       ...
State_VT                -0.020014 -0.022618 -0.027309 -0.023550
State_WA                 1.000000 -0.020720 -0.025017 -0.021574
State_WI                -0.020720  1.000000 -0.028272 -0.024381
State_WV                -0.025017 -0.028272  1.000000 -0.029436
State_WY                -0.021574 -0.024381 -0.029436  1.000000

[68 rows x 68 columns]
```

```
# Identify remaining highly correlated pairs
remaining_high_corr_pairs =
new_correlation_matrix[new_correlation_matrix.abs() >
0.8].stack().reset_index()
remaining_high_corr_pairs =
remaining_high_corr_pairs[remaining_high_corr_pairs['level_0'] !=
remaining_high_corr_pairs['level_1']]
remaining_high_corr_pairs.columns = ['Feature1', 'Feature2',
'Correlation']

# Display remaining high correlation pairs
print("Remaining highly correlated feature pairs:")
print(remaining_high_corr_pairs)
```

```
Remaining highly correlated feature pairs:
Empty DataFrame
Columns: [Feature1, Feature2, Correlation]
Index: []
```

Training and Testing The data set

```
from sklearn.model_selection import train_test_split

X = table_reduced.drop('Churn', axis=1)  # Features
y = table_reduced['Churn']  # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

```python
#Random Forest Model
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```python
#Model Evaluation
from sklearn.metrics import classification_report, confusion_matrix

y_pred = model.predict(X_test)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[454   1]
 [ 53  26]]
              precision    recall  f1-score   support

           0       0.90      1.00      0.94       455
           1       0.96      0.33      0.49        79

    accuracy                           0.90       534
   macro avg       0.93      0.66      0.72       534
weighted avg       0.91      0.90      0.88       534
```

```python
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20]
}

grid_search = GridSearchCV(estimator=model, param_grid=param_grid,
cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

print("Best parameters found: ", grid_search.best_params_)
```

```
Best parameters found:  {'max_depth': 20, 'n_estimators': 200}
```

```python
importances = model.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns,
'Importance': importances})
feature_importance_df =
feature_importance_df.sort_values(by='Importance', ascending=False)

print(feature_importance_df)
```

```
                       Feature  Importance
5            Total day minutes    0.158508
15  Day_Intl_Charge_Interaction    0.106663
13      Customer service calls    0.100856
7            Total eve minutes    0.076711
9          Total night minutes    0.057647
..                         ...         ...
23                    State_DC    0.000730
46                    State_NH    0.000683
3              Voice mail plan    0.000000
2           International plan    0.000000
16       TotalCharges_Per_Call    0.000000

[67 rows x 2 columns]
```

```
import joblib

joblib.dump(model, 'churn_prediction_model.pkl')
```

```
['churn_prediction_model.pkl']
```

```
# Example of checking feature names from a trained model
feature_names = model.feature_names_in_
print("Feature names used during training:")
print(feature_names)
```

```
Feature names used during training:
['Account length' 'Area code' 'International plan' 'Voice mail plan'
 'Number vmail messages' 'Total day minutes' 'Total day calls'
 'Total eve minutes' 'Total eve calls' 'Total night minutes'
 'Total night calls' 'Total intl minutes' 'Total intl calls'
 'Customer service calls' 'MonthlyCharges'
'Day_Intl_Charge_Interaction'
 'TotalCharges_Per_Call' 'State_AL' 'State_AR' 'State_AZ' 'State_CA'
 'State_CO' 'State_CT' 'State_DC' 'State_DE' 'State_FL' 'State_GA'
 'State_HI' 'State_IA' 'State_ID' 'State_IL' 'State_IN' 'State_KS'
 'State_KY' 'State_LA' 'State_MA' 'State_MD' 'State_ME' 'State_MI'
 'State_MN' 'State_MO' 'State_MS' 'State_MT' 'State_NC' 'State_ND'
 'State_NE' 'State_NH' 'State_NJ' 'State_NM' 'State_NV' 'State_NY'
 'State_OH' 'State_OK' 'State_OR' 'State_PA' 'State_RI' 'State_SC'
 'State_SD' 'State_TN' 'State_TX' 'State_UT' 'State_VA' 'State_VT'
 'State_WA' 'State_WI' 'State_WV' 'State_WY']
```

```
import pandas as pd
import joblib

# Load the trained model
model = joblib.load('churn_prediction_model.pkl')

# Define the feature names used during training
training_feature_names = ['Account length', 'Area code',
```

```
'International plan', 'Voice mail plan',
                           'Number vmail messages', 'Total day
minutes', 'Total day calls',
                           'Total eve minutes', 'Total eve calls',
'Total night minutes',
                           'Total night calls', 'Total intl minutes',
'Total intl calls',
                           'Customer service calls', 'MonthlyCharges',
'Day_Intl_Charge_Interaction',
                           'TotalCharges_Per_Call', 'State_AL',
'State_AR', 'State_AZ', 'State_CA',
                           'State_CO', 'State_CT', 'State_DC',
'State_DE', 'State_FL', 'State_GA',
                           'State_HI', 'State_IA', 'State_ID',
'State_IL', 'State_IN', 'State_KS',
                           'State_KY', 'State_LA', 'State_MA',
'State_MD', 'State_ME', 'State_MI',
                           'State_MN', 'State_MO', 'State_MS',
'State_MT', 'State_NC', 'State_ND',
                           'State_NE', 'State_NH', 'State_NJ',
'State_NM', 'State_NV', 'State_NY',
                           'State_OH', 'State_OK', 'State_OR',
'State_PA', 'State_RI', 'State_SC',
                           'State_SD', 'State_TN', 'State_TX',
'State_UT', 'State_VA', 'State_VT',
                           'State_WA', 'State_WI', 'State_WV',
'State_WY']

# Prepare new data with all required features
new_data = pd.DataFrame({
    'Account length': [100],
    'Area code': [415],
    'International plan': [1],
    'Voice mail plan': [0],
    'Number vmail messages': [25],
    'Total day minutes': [200.0],
    'Total day calls': [100],
    'Total eve minutes': [150.0],
    'Total eve calls': [120],
    'Total night minutes': [100.0],
    'Total night calls': [100],
    'Total intl minutes': [10.0],
    'Total intl calls': [5],
    'Customer service calls': [3],
    'MonthlyCharges': [70.0],
    'Day_Intl_Charge_Interaction': [15.0],
    'TotalCharges_Per_Call': [2.5],
    'State_AL': [0],
    'State_AR': [0],
```

```
    'State_AZ': [0],
    'State_CA': [1],
    'State_CO': [0],
    'State_CT': [0],
    'State_DC': [0],
    'State_DE': [0],
    'State_FL': [0],
    'State_GA': [0],
    'State_HI': [0],
    'State_IA': [0],
    'State_ID': [0],
    'State_IL': [0],
    'State_IN': [0],
    'State_KS': [0],
    'State_KY': [0],
    'State_LA': [0],
    'State_MA': [0],
    'State_MD': [0],
    'State_ME': [0],
    'State_MI': [0],
    'State_MN': [0],
    'State_MO': [0],
    'State_MS': [0],
    'State_MT': [0],
    'State_NC': [0],
    'State_ND': [0],
    'State_NE': [0],
    'State_NH': [0],
    'State_NJ': [0],
    'State_NM': [0],
    'State_NV': [0],
    'State_NY': [0],
    'State_OH': [0],
    'State_OK': [0],
    'State_OR': [0],
    'State_PA': [0],
    'State_RI': [0],
    'State_SC': [0],
    'State_SD': [0],
    'State_TN': [0],
    'State_TX': [0],
    'State_UT': [0],
    'State_VA': [0],
    'State_VT': [0],
    'State_WA': [0],
    'State_WI': [0],
    'State_WV': [0],
    'State_WY': [0]
})
```

```python
# Ensure new data has the same columns as the model expects
new_data = new_data.reindex(columns=training_feature_names,
fill_value=0)

# Make predictions
predictions = model.predict(new_data)

# Print the prediction
print(predictions)
```

```
[0]
```

```python
# Example prediction output
prediction = model.predict(new_data)

# Convert prediction to a human-readable message
if prediction[0] == 0:
    message = "The customer is not likely to churn."
else:
    message = "The customer is likely to churn."

print(message)
```

```
The customer is not likely to churn.
```

```python
# Get the probability of each class
probabilities = model.predict_proba(new_data)

# Probability of churn (1) for the first sample
churn_probability = probabilities[0][1]

print(f"Probability of churn: {churn_probability:.2f}")
```

```
Probability of churn: 0.09
```

```python
from sklearn.metrics import accuracy_score

# Assuming X_test and y_test are your test features and labels
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

print(f"Model accuracy: {accuracy:.2f}")
```

```
Model accuracy: 0.90
```

```python
import pandas as pd
import joblib

# Load the trained model
model = joblib.load('churn_prediction_model.pkl')
```

```
# Define the feature names used during training
training_feature_names = ['Account length', 'Area code',
'International plan', 'Voice mail plan',
                           'Number vmail messages', 'Total day
minutes', 'Total day calls',
                           'Total eve minutes', 'Total eve calls',
'Total night minutes',
                           'Total night calls', 'Total intl minutes',
'Total intl calls',
                           'Customer service calls', 'MonthlyCharges',
'Day_Intl_Charge_Interaction',
                           'TotalCharges_Per_Call', 'State_AL',
'State_AR', 'State_AZ', 'State_CA',
                           'State_CO', 'State_CT', 'State_DC',
'State_DE', 'State_FL', 'State_GA',
                           'State_HI', 'State_IA', 'State_ID',
'State_IL', 'State_IN', 'State_KS',
                           'State_KY', 'State_LA', 'State_MA',
'State_MD', 'State_ME', 'State_MI',
                           'State_MN', 'State_MO', 'State_MS',
'State_MT', 'State_NC', 'State_ND',
                           'State_NE', 'State_NH', 'State_NJ',
'State_NM', 'State_NV', 'State_NY',
                           'State_OH', 'State_OK', 'State_OR',
'State_PA', 'State_RI', 'State_SC',
                           'State_SD', 'State_TN', 'State_TX',
'State_UT', 'State_VA', 'State_VT',
                           'State_WA', 'State_WI', 'State_WV',
'State_WY']

# Prepare new data with all required features
new_data = pd.DataFrame({
    'Account length': [50],  # Example value
    'Area code': [415],      # Example value
    'International plan': [1],  # Assuming 1 indicates 'yes'
    'Voice mail plan': [0],     # Assuming 0 indicates 'no'
    'Number vmail messages': [25],  # Example value
    'Total day minutes': [30],  # Example value
    'Total day calls': [100],  # Example value
    'Total eve minutes': [200],  # Example value
    'Total eve calls': [80],  # Example value
    'Total night minutes': [10],  # Example value
    'Total night calls': [40],  # Example value
    'Total intl minutes': [10],  # Example value
    'Total intl calls': [10],  # Example value
    'Customer service calls': [5],  # Example value
    'MonthlyCharges': [100],  # Example value
    'Day_Intl_Charge_Interaction': [1],  # Example value
    'TotalCharges_Per_Call': [5],
```

```
    'State_AL': [0],
    'State_AR': [0],
    'State_AZ': [0],
    'State_CA': [1],
    'State_CO': [0],
    'State_CT': [0],
    'State_DC': [0],
    'State_DE': [0],
    'State_FL': [0],
    'State_GA': [0],
    'State_HI': [0],
    'State_IA': [0],
    'State_ID': [0],
    'State_IL': [0],
    'State_IN': [0],
    'State_KS': [0],
    'State_KY': [0],
    'State_LA': [0],
    'State_MA': [0],
    'State_MD': [0],
    'State_ME': [0],
    'State_MI': [0],
    'State_MN': [0],
    'State_MO': [0],
    'State_MS': [0],
    'State_MT': [0],
    'State_NC': [0],
    'State_ND': [0],
    'State_NE': [0],
    'State_NH': [0],
    'State_NJ': [0],
    'State_NM': [0],
    'State_NV': [0],
    'State_NY': [0],
    'State_OH': [0],
    'State_OK': [0],
    'State_OR': [0],
    'State_PA': [0],
    'State_RI': [0],
    'State_SC': [0],
    'State_SD': [0],
    'State_TN': [0],
    'State_TX': [0],
    'State_UT': [0],
    'State_VA': [0],
    'State_VT': [0],
    'State_WA': [0],
    'State_WI': [0],
    'State_WV': [0],
```

```
    'State_WY': [0]
})

# Ensure new data has the same columns as the model expects
new_data = new_data.reindex(columns=training_feature_names,
fill_value=0)

# Make predictions
predictions = model.predict(new_data)

# Print the prediction
print(predictions)
```

```
[0]
```

```
import pandas as pd
import joblib

# Load the trained model
model = joblib.load('churn_prediction_model.pkl')

# Create a DataFrame for a customer likely to churn
sample_data = pd.DataFrame({
    'Account length': [120],
    'Area code': [415],
    'International plan': [1],
    'Voice mail plan': [1],
    'Number vmail messages': [30],
    'Total day minutes': [500],
    'Total day calls': [150],
    'Total eve minutes': [400],
    'Total eve calls': [120],
    'Total night minutes': [300],
    'Total night calls': [90],
    'Total intl minutes': [50],
    'Total intl calls': [20],
    'Customer service calls': [10],
    'MonthlyCharges': [100],
    'Day_Intl_Charge_Interaction': [1],
    'TotalCharges_Per_Call': [6],
   'State_AL': [0],
    'State_AR': [0],
    'State_AZ': [0],
    'State_CA': [1],
    'State_CO': [0],
    'State_CT': [0],
    'State_DC': [0],
    'State_DE': [0],
    'State_FL': [0],
    'State_GA': [0],
```

```python
    'State_HI': [0],
    'State_IA': [0],
    'State_ID': [0],
    'State_IL': [0],
    'State_IN': [0],
    'State_KS': [0],
    'State_KY': [0],
    'State_LA': [0],
    'State_MA': [0],
    'State_MD': [0],
    'State_ME': [0],
    'State_MI': [0],
    'State_MN': [0],
    'State_MO': [0],
    'State_MS': [0],
    'State_MT': [0],
    'State_NC': [0],
    'State_ND': [0],
    'State_NE': [0],
    'State_NH': [0],
    'State_NJ': [0],
    'State_NM': [0],
    'State_NV': [0],
    'State_NY': [0],
    'State_OH': [0],
    'State_OK': [0],
    'State_OR': [0],
    'State_PA': [0],
    'State_RI': [0],
    'State_SC': [0],
    'State_SD': [0],
    'State_TN': [0],
    'State_TX': [0],
    'State_UT': [0],
    'State_VA': [0],
    'State_VT': [0],
    'State_WA': [0],
    'State_WI': [0],
    'State_WV': [0],
    'State_WY': [0]
})

def print_colored_box(text, color_code, box_width=50):
    # Create the box
    border = "+" + "-" * (box_width - 2) + "+"
    padding = (box_width - len(text) - 2) // 2
    box_text = "|" + " " * padding + text + " " * (box_width -
len(text) - padding - 2) + "|"
```

```python
    # Print the box with color
    print(f"\033[{color_code}m{border}\n{box_text}\n{border}\033[0m")

# Example usage

# Predict churn
predictions = model.predict(sample_data)
probabilities = model.predict_proba(sample_data)

# Output the results
print_colored_box("Prediction: " + ("Churn" if predictions[0] == 1 else "Not Churn"), "31")  # Red for Churn
print_colored_box(f"Probability of churn: {probabilities[0][1]:.2f}", "32")  # Green for Probability

# Model accuracy
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print_colored_box(f"Model accuracy: {accuracy:.2f}", "35")  # Magenta for Accuracy

# Model evaluation
print_colored_box("Confusion Matrix", "33")  # Yellow for Confusion Matrix
print(confusion_matrix(y_test, y_pred))

print_colored_box("Classification Report", "36")  # Cyan for Classification Report
print(classification_report(y_test, y_pred))
```

```
+-------------------------------------------------+
|                Prediction: Churn                |
+-------------------------------------------------+
+-------------------------------------------------+
|            Probability of churn: 0.54           |
+-------------------------------------------------+
+-------------------------------------------------+
|               Model accuracy: 0.90              |
+-------------------------------------------------+
+-------------------------------------------------+
|                 Confusion Matrix                |
+-------------------------------------------------+
[[454   1]
 [ 53  26]]
+-------------------------------------------------+
|              Classification Report              |
+-------------------------------------------------+
              precision    recall  f1-score   support
```

```
           0       0.90      1.00      0.94       455
           1       0.96      0.33      0.49        79

    accuracy                           0.90       534
   macro avg       0.93      0.66      0.72       534
weighted avg       0.91      0.90      0.88       534
```

```python
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report
import joblib

# Initialize the SVM model
svm_model = SVC(probability=True, kernel='linear')  # Use linear
kernel for simplicity

# Train the model
svm_model.fit(X_train, y_train)
```

```
SVC(kernel='linear', probability=True)
```

```python
# Predict churn using the SVM model
svm_predictions = svm_model.predict(X_test)
svm_probabilities = svm_model.predict_proba(X_test)

# Output the results
print("SVM Model:")
print("Prediction:", "Churn" if svm_predictions[0] == 1 else "Not
Churn")
print("Probability of churn:", svm_probabilities[0][1])

# Model accuracy
svm_accuracy = accuracy_score(y_test, svm_predictions)
print(f"Model accuracy: {svm_accuracy:.2f}")

# Model evaluation
print(confusion_matrix(y_test, svm_predictions))
print(classification_report(y_test, svm_predictions))
```

```
SVM Model:
Prediction: Not Churn
Probability of churn: 0.3407298387731522
Model accuracy: 0.85
[[455   0]
 [ 79   0]]
              precision    recall  f1-score   support

           0       0.85      1.00      0.92       455
           1       0.00      0.00      0.00        79
```

```
    accuracy                           0.85       534
   macro avg       0.43      0.50      0.46       534
weighted avg       0.73      0.85      0.78       534


C:\Users\anura\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
C:\Users\anura\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
C:\Users\anura\anaconda3\Lib\site-packages\sklearn\metrics\
_classification.py:1509: UndefinedMetricWarning: Precision is ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline

# Create a pipeline with scaling and logistic regression
pipeline = make_pipeline(StandardScaler(),
LogisticRegression(max_iter=3000))

# Fit the model
pipeline.fit(X_train, y_train)
```

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression',
LogisticRegression(max_iter=3000))])
```

```
from sklearn.linear_model import LogisticRegression

# Using liblinear solver
model = LogisticRegression(max_iter=3000, solver='liblinear')
model.fit(X_train, y_train)

# Using saga solver
model = LogisticRegression(max_iter=3000, solver='saga')
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=3000, solver='saga')
```

```python
from sklearn.linear_model import LogisticRegression

# Adjusting regularization strength
model = LogisticRegression(max_iter=3000, C=0.1, solver='liblinear')
model.fit(X_train, y_train)
```

```
LogisticRegression(C=0.1, max_iter=3000, solver='liblinear')
```

```python
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# Create and fit the model with scaling
pipeline = make_pipeline(StandardScaler(),
LogisticRegression(max_iter=3000, solver='liblinear', C=0.1))

# Fit the model
pipeline.fit(X_train, y_train)

# Predict and evaluate
predictions = pipeline.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
print(f"Model accuracy: {accuracy:.2f}")
```

```
Model accuracy: 0.85
```

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

# Example new data
new_data = pd.DataFrame({
    'Account length': [100],
    'Area code': [415],
    'International plan': [0],
    'Voice mail plan': [1],
    'Number vmail messages': [25],
    'Total day minutes': [200],
    'Total day calls': [100],
    'Total eve minutes': [150],
    'Total eve calls': [80],
    'Total night minutes': [180],
    'Total night calls': [90],
    'Total intl minutes': [10],
    'Total intl calls': [5],
    'Customer service calls': [1],
    'MonthlyCharges': [75],
    'Day_Intl_Charge_Interaction': [2.5],
    'TotalCharges_Per_Call': [0.3],
```

```
    'State_AL': [0],
    'State_AR': [0],
    'State_AZ': [0],
    'State_CA': [1],
    'State_CO': [0],
    'State_CT': [0],
    'State_DC': [0],
    'State_DE': [0],
    'State_FL': [0],
    'State_GA': [0],
    'State_HI': [0],
    'State_IA': [0],
    'State_ID': [0],
    'State_IL': [0],
    'State_IN': [0],
    'State_KS': [0],
    'State_KY': [0],
    'State_LA': [0],
    'State_MA': [0],
    'State_MD': [0],
    'State_ME': [0],
    'State_MI': [0],
    'State_MN': [0],
    'State_MO': [0],
    'State_MS': [0],
    'State_MT': [0],
    'State_NC': [0],
    'State_ND': [0],
    'State_NE': [0],
    'State_NH': [0],
    'State_NJ': [0],
    'State_NM': [0],
    'State_NV': [0],
    'State_NY': [0],
    'State_OH': [0],
    'State_OK': [0],
    'State_OR': [0],
    'State_PA': [0],
    'State_RI': [0],
    'State_SC': [0],
    'State_SD': [0],
    'State_TN': [0],
    'State_TX': [0],
    'State_UT': [0],
    'State_VA': [0],
    'State_VT': [0],
    'State_WA': [0],
    'State_WI': [0],
    'State_WV': [0],
```

```
    'State_WY': [0]
})

# Assuming 'pipeline' is the Logistic Regression model with scaling
predictions = pipeline.predict(new_data)
probabilities = pipeline.predict_proba(new_data)

# Output the results
print("Prediction:", "Churn" if predictions[0] == 1 else "Not Churn")
print("Probability of churn:", probabilities[0][1])
```

```
Prediction: Not Churn
Probability of churn: 0.036263571005464
```

```
pip install streamlit
```

```
Requirement already satisfied: streamlit in c:\users\anura\anaconda3\
lib\site-packages (1.32.0)
Requirement already satisfied: altair<6,>=4.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (5.0.1)
Requirement already satisfied: blinker<2,>=1.0.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (1.6.2)
Requirement already satisfied: cachetools<6,>=4.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (5.3.3)
Requirement already satisfied: click<9,>=7.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (8.1.7)
Requirement already satisfied: numpy<2,>=1.19.3 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (1.26.4)
Requirement already satisfied: packaging<24,>=16.8 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (23.2)
Requirement already satisfied: pandas<3,>=1.3.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (2.2.2)
Requirement already satisfied: pillow<11,>=7.1.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (10.3.0)
Requirement already satisfied: protobuf<5,>=3.20 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (3.20.3)
Requirement already satisfied: pyarrow>=7.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (14.0.2)
Requirement already satisfied: requests<3,>=2.27 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (2.32.2)
Requirement already satisfied: rich<14,>=10.14.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (13.3.5)
Requirement already satisfied: tenacity<9,>=8.1.0 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (8.2.2)
Requirement already satisfied: toml<2,>=0.10.1 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.3.0 in c:\
users\anura\anaconda3\lib\site-packages (from streamlit) (4.11.0)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in c:\
users\anura\anaconda3\lib\site-packages (from streamlit) (3.1.37)
```

```
Requirement already satisfied: pydeck<1,>=0.8.0b4 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (0.8.0)
Requirement already satisfied: tornado<7,>=6.0.3 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (6.4.1)
Requirement already satisfied: watchdog>=2.1.5 in c:\users\anura\
anaconda3\lib\site-packages (from streamlit) (4.0.1)
Requirement already satisfied: jinja2 in c:\users\anura\anaconda3\lib\
site-packages (from altair<6,>=4.0->streamlit) (3.1.4)
Requirement already satisfied: jsonschema>=3.0 in c:\users\anura\
anaconda3\lib\site-packages (from altair<6,>=4.0->streamlit) (4.19.2)
Requirement already satisfied: toolz in c:\users\anura\anaconda3\lib\
site-packages (from altair<6,>=4.0->streamlit) (0.12.0)
Requirement already satisfied: colorama in c:\users\anura\anaconda3\
lib\site-packages (from click<9,>=7.0->streamlit) (0.4.6)
Requirement already satisfied: gitdb<5,>=4.0.1 in c:\users\anura\
anaconda3\lib\site-packages (from gitpython!=3.1.19,<4,>=3.0.7-
>streamlit) (4.0.7)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\
anura\anaconda3\lib\site-packages (from pandas<3,>=1.3.0->streamlit)
(2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\anura\
anaconda3\lib\site-packages (from pandas<3,>=1.3.0->streamlit)
(2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\anura\
anaconda3\lib\site-packages (from pandas<3,>=1.3.0->streamlit)
(2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
anura\anaconda3\lib\site-packages (from requests<3,>=2.27->streamlit)
(2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anura\
anaconda3\lib\site-packages (from requests<3,>=2.27->streamlit) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\anura\
anaconda3\lib\site-packages (from requests<3,>=2.27->streamlit)
(2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\anura\
anaconda3\lib\site-packages (from requests<3,>=2.27->streamlit)
(2024.7.4)
Requirement already satisfied: markdown-it-py<3.0.0,>=2.2.0 in c:\
users\anura\anaconda3\lib\site-packages (from rich<14,>=10.14.0-
>streamlit) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in c:\users\
anura\anaconda3\lib\site-packages (from rich<14,>=10.14.0->streamlit)
(2.15.1)
Requirement already satisfied: smmap<5,>=3.0.1 in c:\users\anura\
anaconda3\lib\site-packages (from gitdb<5,>=4.0.1->gitpython!
=3.1.19,<4,>=3.0.7->streamlit) (4.0.0)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\anura\
anaconda3\lib\site-packages (from jinja2->altair<6,>=4.0->streamlit)
(2.1.3)
```

```
Requirement already satisfied: attrs>=22.2.0 in c:\users\anura\anaconda3\lib\site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\users\anura\anaconda3\lib\site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (2023.7.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\anura\anaconda3\lib\site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\anura\anaconda3\lib\site-packages (from jsonschema>=3.0->altair<6,>=4.0->streamlit) (0.10.6)
Requirement already satisfied: mdurl~=0.1 in c:\users\anura\anaconda3\lib\site-packages (from markdown-it-py<3.0.0,>=2.2.0->rich<14,>=10.14.0->streamlit) (0.1.0)
Requirement already satisfied: six>=1.5 in c:\users\anura\anaconda3\lib\site-packages (from python-dateutil>=2.8.2->pandas<3,>=1.3.0->streamlit) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

```python
import streamlit as st
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline

# Load your model
# Replace with your own model and scaler
pipeline = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(max_iter=3000))
])

# Function to predict churn
def predict_churn(data):
    return pipeline.predict(data)

# Streamlit UI
st.title('Customer Churn Prediction')

# Input fields
account_length = st.number_input('Account Length', min_value=0)
area_code = st.number_input('Area Code', min_value=0)
international_plan = st.selectbox('International Plan', ['No', 'Yes'])
voice_mail_plan = st.selectbox('Voice Mail Plan', ['No', 'Yes'])
number_vmail_messages = st.number_input('Number of Voicemail Messages', min_value=0)
total_day_minutes = st.number_input('Total Day Minutes', min_value=0.0)
# Add other features similarly
```

```
# Create a DataFrame from inputs
input_data = pd.DataFrame({
    'Account length': [account_length],
    'Area code': [area_code],
    'International plan': [1 if international_plan == 'Yes' else 0],
    'Voice mail plan': [1 if voice_mail_plan == 'Yes' else 0],
    'Number vmail messages': [number_vmail_messages],
    'Total day minutes': [total_day_minutes],
    # Add other features similarly
})

# Predict
if st.button('Predict'):
    prediction = predict_churn(input_data)
    st.write("Prediction:", "Churn" if prediction[0] == 1 else "Not
Churn")

2024-09-04 11:27:51.328
  Warning: to view this Streamlit app on a browser, run it with the
following
  command:

    streamlit run C:\Users\anura\anaconda3\Lib\site-packages\
ipykernel_launcher.py [ARGUMENTS]
```