Churn Analysis ¶

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
```

In [2]:
```
data = pd.read_excel('view_churn_data.xlsx')
data.head(5)
```

Out[2]:

	Customer_ID	Gender	Age	Married	State	Number_of_Referrals	Tenure_in_Months	Value_Deal	Phone_Service	Multiple_Lines	...	Paym
0	19877-DEL	Male	35	No	Delhi	7	27	None	Yes	No	...	
1	58353-MAH	Female	45	Yes	Maharashtra	14	13	None	Yes	Yes	...	
2	25063-WES	Male	51	No	West Bengal	4	35	Deal 5	Yes	No	...	Ban
3	59787-KAR	Male	79	No	Karnataka	3	21	Deal 4	Yes	No	...	Ban
4	28544-TAM	Female	80	No	Tamil Nadu	3	8	None	Yes	No	...	

5 rows × 32 columns

In [3]:
```
data.shape
```

Out[3]:
```
(6007, 32)
```

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
Customer_ID                    0
Gender                         0
Age                            0
Married                        0
State                          0
Number_of_Referrals            0
Tenure_in_Months               0
Value_Deal                     0
Phone_Service                  0
Multiple_Lines                 0
Internet_Service               0
Internet_Type                  0
Online_Security                0
Online_Backup                  0
Device_Protection_Plan         0
Premium_Support                0
Streaming_TV                   0
Streaming_Movies               0
Streaming_Music                0
Unlimited_Data                 0
Contract                       0
Paperless_Billing              0
Payment_Method                 0
Monthly_Charge                 0
Total_Charges                  0
Total_Refunds                  0
Total_Extra_Data_Charges       0
Total_Long_Distance_Charges    0
Total_Revenue                  0
Customer_Status                0
Churn_Category                 0
Churn_Reason                   0
dtype: int64
```

In [5]:
```
data.describe()
```

Out[5]:

	Age	Number_of_Referrals	Tenure_in_Months	Monthly_Charge	Total_Charges	Total_Refunds	Total_Extra_Data_Charges	Total_Long
count	6007.000000	6007.000000	6007.00000	6007.000000	6007.000000	6007.000000	6007.000000	
mean	47.289163	7.439820	17.39454	65.087598	2430.986173	2.038612	7.015149	
std	16.805110	4.622369	10.59292	31.067808	2267.481294	8.065520	25.405737	
min	18.000000	0.000000	1.00000	-10.000000	19.100000	0.000000	0.000000	
25%	33.000000	3.000000	8.00000	35.950000	539.950000	0.000000	0.000000	
50%	47.000000	7.000000	17.00000	71.100000	1556.850000	0.000000	0.000000	
75%	60.000000	11.000000	27.00000	90.450000	4013.900000	0.000000	0.000000	
max	84.000000	15.000000	36.00000	118.750000	8684.800000	49.790000	150.000000	

Data Preprocessing

In [6]:
```
# Drop columns that won't be used for prediction
data = data.drop(['Customer_ID', 'Churn_Category', 'Churn_Reason'], axis=1)

# List of columns to be label encoded
columns_to_encode = ['Gender', 'Married', 'State', 'Value_Deal', 'Phone_Service', 'Multiple_Lines','Internet_Service',
                     'Internet_Type', 'Online_Security', 'Online_Backup','Device_Protection_Plan', 'Premium_Support',
                     'Streaming_TV', 'Streaming_Movies','Streaming_Music', 'Unlimited_Data', 'Contract',
                     'Paperless_Billing','Payment_Method']
```

In [7]:
```
data.head(5)
```

Out[7]:

	Gender	Age	Married	State	Number_of_Referrals	Tenure_in_Months	Value_Deal	Phone_Service	Multiple_Lines	Internet_Service	...	C
0	Male	35	No	Delhi	7	27	None	Yes	No	Yes	...	
1	Female	45	Yes	Maharashtra	14	13	None	Yes	Yes	Yes	...	tc
2	Male	51	No	West Bengal	4	35	Deal 5	Yes	No	Yes	...	tc
3	Male	79	No	Karnataka	3	21	Deal 4	Yes	No	Yes	...	tc
4	Female	80	No	Tamil Nadu	3	8	None	Yes	No	Yes	...	tc

5 rows × 29 columns

In [8]:
```
# Encode categorical variables except the target variable
label_encoders = {}
for column in columns_to_encode:
    label_encoders[column] = LabelEncoder()
    data[column] = label_encoders[column].fit_transform(data[column])

# Manually encode the target variable 'Customer_Status'
data['Customer_Status'] = data['Customer_Status'].map({'Stayed': 0, 'Churned': 1})
```

In [9]:
```
# Split data into features and target

X = data.drop('Customer_Status', axis=1)
y = data['Customer_Status']

# Split data 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)
```

Train Random Forest Model

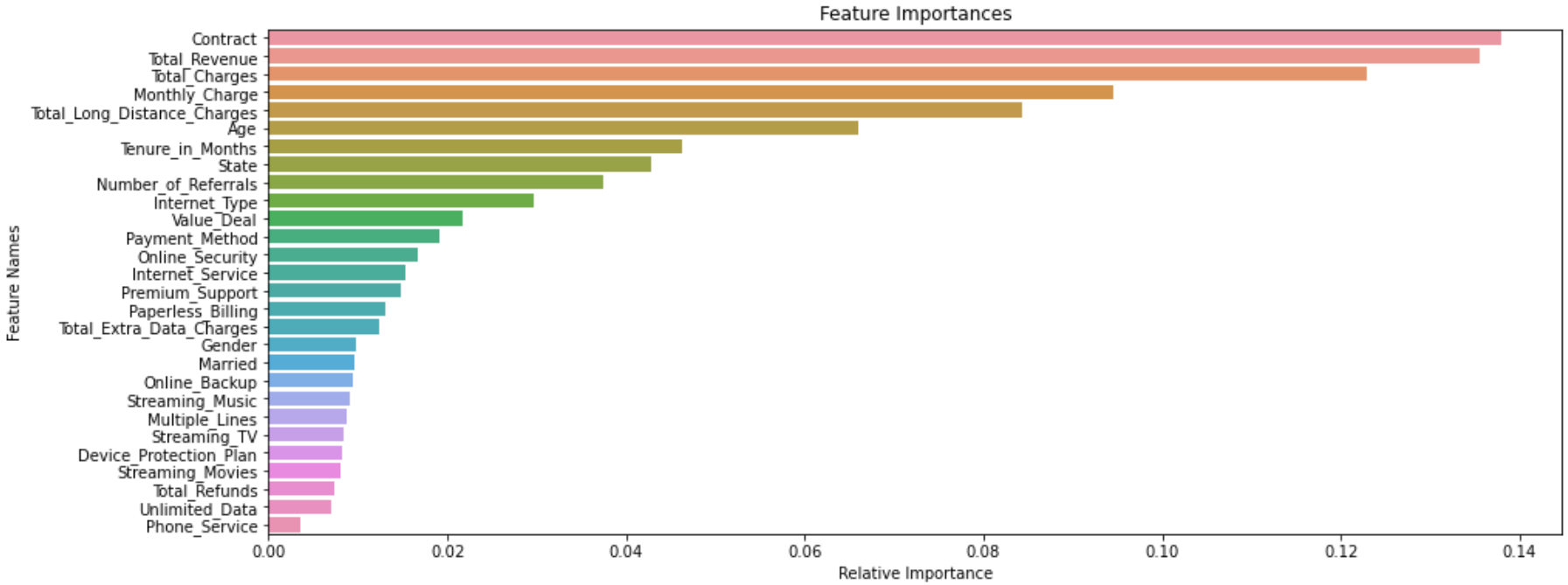
In [10]:
```
# Initialize the Random Forest Classifier
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

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

Out[10]:
```
RandomForestClassifier(random_state=42)
```

Evaluate Model

In [11]:
```
# Make predictions
y_pred = rf_model.predict(X_test)

# Evaluate the model
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

# Feature Selection using Feature Importance
importances = rf_model.feature_importances_
indices = np.argsort(importances)[::-1]

# Plot the feature importances

plt.figure(figsize=(15, 6))
sns.barplot(x=importances[indices], y=X.columns[indices])
plt.title('Feature Importances')
plt.xlabel('Relative Importance')
plt.ylabel('Feature Names')
plt.show()
```

```
Confusion Matrix:
[[791  50]
 [125 236]]

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.94      0.90       841
           1       0.83      0.65      0.73       361

    accuracy                           0.85      1202
   macro avg       0.84      0.80      0.81      1202
weighted avg       0.85      0.85      0.85      1202
```


Feature Importances
Contract
Total_Revenue
Total_Charges
Monthly_Charge
Total_Long_Distance_Charges
Age
Tenure_in_Months
State
Number_of_Referrals
Internet_Type
Value_Deal
Payment_Method
Online_Security
Internet_Service
Premium_Support
Paperless_Billing
Total_Extra_Data_Charges
Gender
Married
Online_Backup
Streaming_Music
Multiple_Lines
Streaming_TV
Device_Protection_Plan
Streaming_Movies
Total_Refunds
Unlimited_Data
Phone_Service
Feature Names
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0.14
Relative Importance

Use Model for Prediction on New Data

In [12]:
```
new_data = pd.read_excel('view_churn_joined.xlsx')
new_data.head(5)
```

Out[12]:

	Customer_ID	Gender	Age	Married	State	Number_of_Referrals	Tenure_in_Months	Value_Deal	Phone_Service	Multiple_Lines	...	Paymer
0	93520-GUJ	Female	67	No	Gujarat	13	19	Deal 5	Yes	Yes	...	Bank
1	57256-BIH	Female	18	No	Bihar	9	7	None	Yes	No	...	C
2	72357-MAD	Female	53	No	Madhya Pradesh	14	12	Deal 5	Yes	No	...	C
3	66612-KAR	Female	58	Yes	Karnataka	11	18	None	Yes	No	...	C
4	22119-WES	Male	31	Yes	West Bengal	5	5	None	Yes	No	...	C

5 rows × 32 columns

In [13]:
```
new_data.shape
```

Out[13]: (411, 32)

In [14]:
```
# Retain the original DataFrame to preserve unencoded columns
original_data = new_data.copy()
```

In [15]:
```
original_data.head(5)
```

Out[15]:

	Customer_ID	Gender	Age	Married	State	Number_of_Referrals	Tenure_in_Months	Value_Deal	Phone_Service	Multiple_Lines	...	Paymer
0	93520-GUJ	Female	67	No	Gujarat	13	19	Deal 5	Yes	Yes	...	Bank
1	57256-BIH	Female	18	No	Bihar	9	7	None	Yes	No	...	C
2	72357-MAD	Female	53	No	Madhya Pradesh	14	12	Deal 5	Yes	No	...	C
3	66612-KAR	Female	58	Yes	Karnataka	11	18	None	Yes	No	...	C
4	22119-WES	Male	31	Yes	West Bengal	5	5	None	Yes	No	...	C

5 rows × 32 columns

In [16]:
```
# Retain the Customer_ID column
customer_ids = new_data['Customer_ID']
```

In [17]:
```
# Drop columns that won't be used for prediction in the encoded DataFrame

new_data = new_data.drop(['Customer_ID', 'Customer_Status', 'Churn_Category', 'Churn_Reason'], axis=1)
```

In [18]:
```
# Encode categorical variables using the saved label encoders
for column in new_data.select_dtypes(include=['object']).columns:
    new_data[column] = label_encoders[column].fit_transform(new_data[column])
```

In [19]:
```
# Make predictions
new_predictions = rf_model.predict(new_data)

# Add predictions to the original DataFrame
original_data['Customer_Status_Predicted'] = new_predictions

# Filter the DataFrame to include only records predicted as "Churned"
original_data = original_data[original_data['Customer_Status_Predicted'] == 1]
```

In [20]:
```
original_data
```

Out[20]:

	Customer_ID	Gender	Age	Married	State	Number_of_Referrals	Tenure_in_Months	Value_Deal	Phone_Service	Multiple_Lines	...	Mont
0	93520-GUJ	Female	67	No	Gujarat	13	19	Deal 5	Yes	Yes	...	
1	57256-BIH	Female	18	No	Bihar	9	7	None	Yes	No	...	
2	72357-MAD	Female	53	No	Madhya Pradesh	14	12	Deal 5	Yes	No	...	
3	66612-KAR	Female	58	Yes	Karnataka	11	18	None	Yes	No	...	
4	22119-WES	Male	31	Yes	West Bengal	5	5	None	Yes	No	...	
...	...	...	...	...	...	...	...	...	...	...	...	...
405	21065-HAR	Male	27	No	Haryana	5	10	None	Yes	No	...	
406	31412-HAR	Female	81	Yes	Haryana	14	29	None	Yes	No	...	
407	54997-UTT	Female	55	No	Uttar Pradesh	7	23	None	Yes	No	...	
408	56728-RAJ	Male	40	No	Rajasthan	0	1	None	Yes	No	...	
409	47624-TAM	Female	62	Yes	Tamil Nadu	7	29	None	Yes	No	...	

376 rows × 33 columns

In [21]:
```
# Save the results
original_data.to_csv(r"C:\Users\prira\Downloads\Power BI\Churn\Predictions.csv", index=False)
```

In [ ]:
```

```