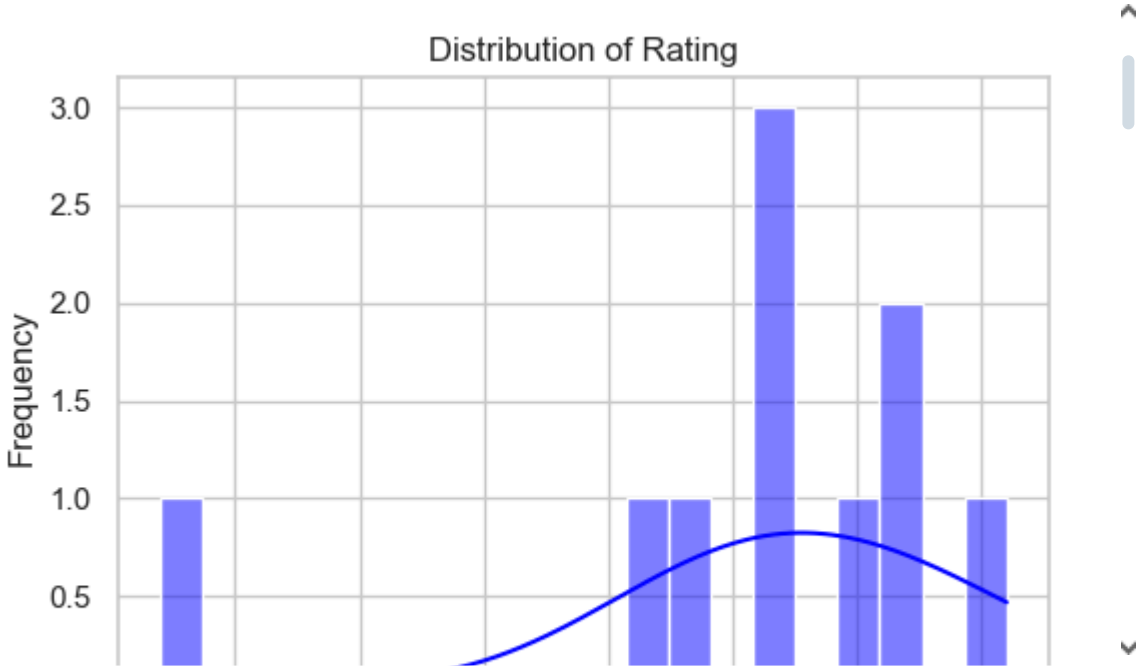
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Set visualization style
sns.set(style="whitegrid", palette="muted")
```

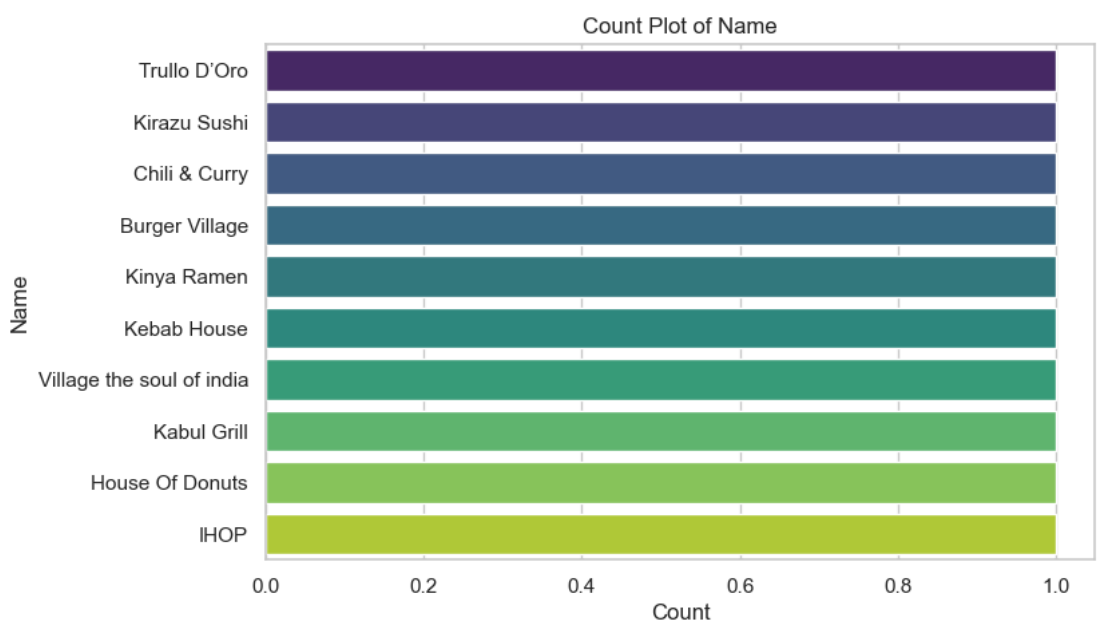
In [2]:
```
# Load dataset (replace 'file_path' with your file's path)
data = pd.read_csv("results1.csv")

# Display the first few rows
print("First 5 Rows of Data:")
print(data.head())

# Basic info and data types
print("\nDataset Info:")
print(data.info())

# Check for missing values
print("\nMissing Values:")
print(data.isnull().sum())

# Summary statistics
print("\nSummary Statistics:")
print(data.describe(include="all"))
```

```
NaN
4                   https://kinya.us (https://kinya.us)          NaN

                                  Instagram  \
0                                       NaN
1                                       NaN
2                                       NaN
3  https://www.instagram.com/burgervillage (https://www.instagram.com/bu
rgervillage)
4     https://www.instagram.com/kinyaramen (https://www.instagram.com/ki
nyaramen)

                                           Facebook Twitter Linkedin
...  \
0      https://www.facebook.com/trullodorolongisland (https://www.facebo
ok.com/trullodorolongisland)     NaN      NaN  ...
1                    https://www.facebook.com/pages (https://www.facebo
ok.com/pages)     NaN      NaN  ...
2                                                    NaN     NaN      NaN
...
```

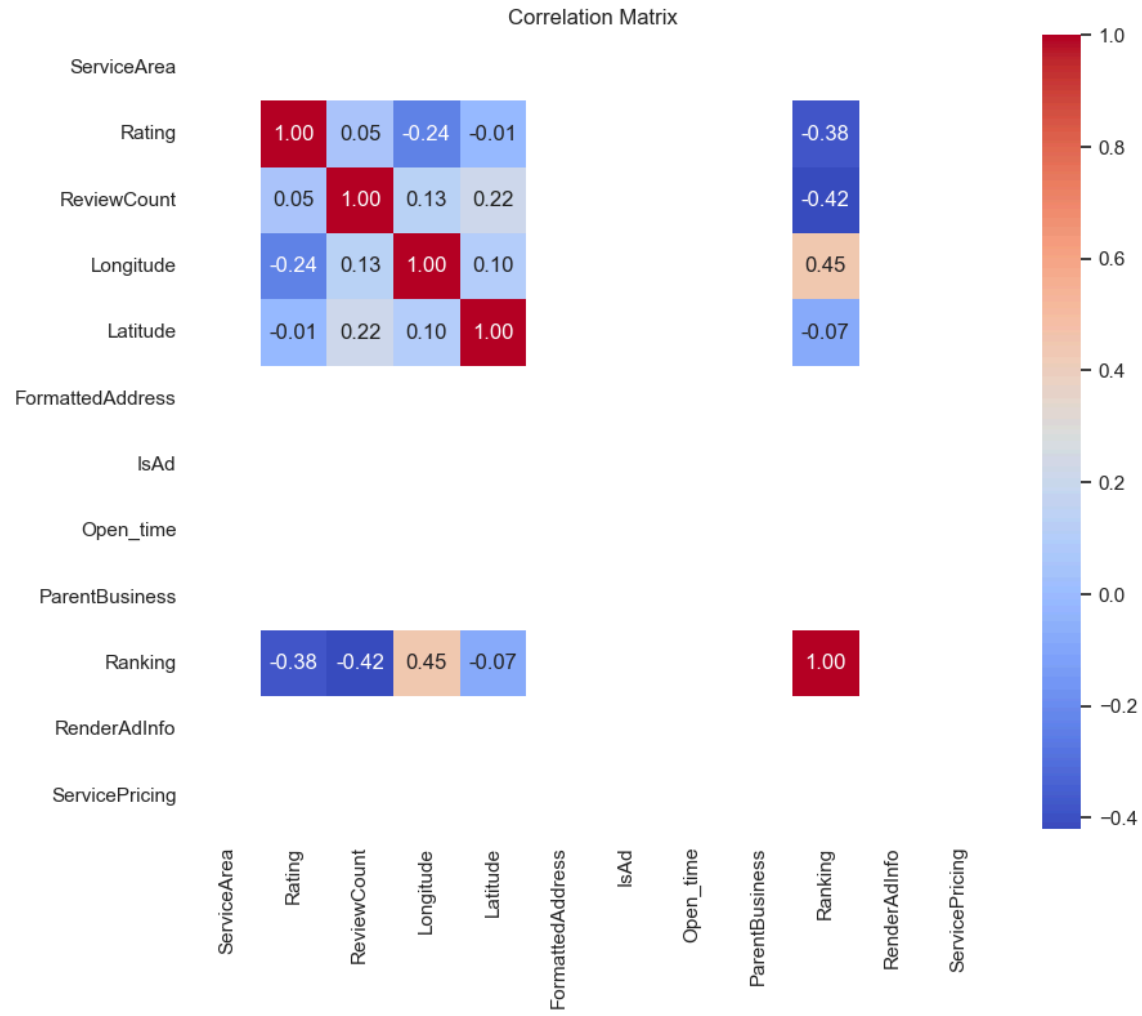
In [3]:
```
# Identify numerical columns
numerical_cols = data.select_dtypes(include=["float64", "int64"]).columns

# Histograms
for col in numerical_cols:
    plt.figure(figsize=(6, 4))
    sns.histplot(data[col], kde=True, bins=20, color="blue")
    plt.title(f"Distribution of {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.show()

# Boxplots
for col in numerical_cols:
    plt.figure(figsize=(6, 4))
    sns.boxplot(x=data[col], color="green")
    plt.title(f"Boxplot of {col}")
    plt.xlabel(col)
    plt.show()
```



In [4]:
```
# Identify categorical columns
categorical_cols = data.select_dtypes(include=["object"]).columns

# Count plots
for col in categorical_cols:
    plt.figure(figsize=(8, 5))
    sns.countplot(y=data[col], order=data[col].value_counts().index, palett
    plt.title(f"Count Plot of {col}")
    plt.xlabel("Count")
    plt.ylabel(col)
    plt.show()
```



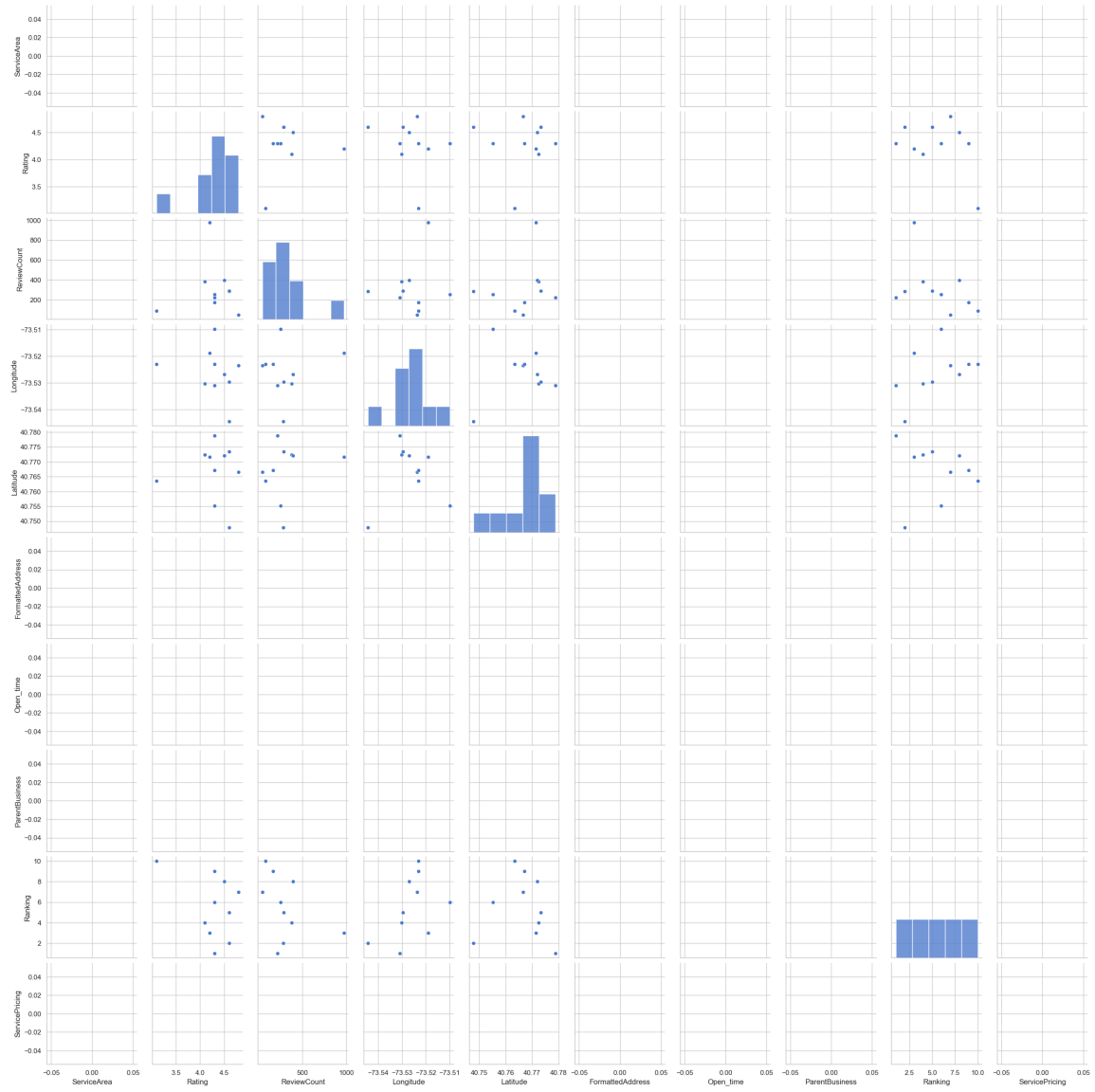
In [5]:
```
# Compute correlation matrix
correlation_matrix = data.corr()

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



In [6]:
```
# Fill missing values with mean (numerical) or mode (categorical)
for col in data.columns:
    if data[col].isnull().sum() > 0:
        if data[col].dtype in ["float64", "int64"]:
            data[col].fillna(data[col].mean(), inplace=True)
        else:
            data[col].fillna(data[col].mode()[0], inplace=True)

# Verify missing values are handled
print("\nMissing Values After Handling:")
print(data.isnull().sum())
```

```
Missing Values After Handling:
Name                                0
Phone                               0
Address                             0
Email                               0
Website                             0
ServiceArea                        10
Instagram                           0
Facebook                            0
Twitter                             0
Linkedin                            0
Youtube                             0
BusinessUrl                         0
Rating                              0
ReviewCount                         0
PriceRange                          0
Longitude                           0
Latitude                            0
Alias                               0
BizId                               0
BusinessSectionUrls_open_hours      0
BusinessSectionUrls_reviews         0
Categories_0_title                  0
Categories_0_url                    0
Categories_1_title                  0
Categories_1_url                    0
Categories_2_title                  0
Categories_2_url                    0
Categories_3_title                  0
Categories_3_url                    0
Categories_4_title                  0
Categories_4_url                    0
FormattedAddress                   10
IsAd                                0
Open_time                          10
ParentBusiness                     10
Ranking                             0
RenderAdInfo                        0
ServicePricing                     10
Snippet                             0
dtype: int64
```

In [7]:
```
# Pairplot for selected numerical columns
sns.pairplot(data[numerical_cols])
plt.show()
```



In [8]:
```
# Example: Price Range vs Rating
if 'PriceRange' in data.columns and 'Rating' in data.columns:
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=data['PriceRange'], y=data['Rating'], palette="muted")
    plt.title("Price Range vs Rating")
    plt.xlabel("Price Range")
    plt.ylabel("Rating")
    plt.show()

# Example: Review Count vs Ranking
if 'ReviewCount' in data.columns and 'Ranking' in data.columns:
    plt.figure(figsize=(8, 6))
    sns.scatterplot(x=data['ReviewCount'], y=data['Ranking'], color="purple
    plt.title("Review Count vs Ranking")
    plt.xlabel("Review Count")
    plt.ylabel("Ranking")
    plt.show()
```

```
---------------------------------------------------------------------------
ParseException                            Traceback (most recent call la
st)
C:\ProgramData\Anaconda3\lib\site-packages\matplotlib\_mathtext.py in pa
rse(self, s, fonts_object, fontsize, dpi)
   2236         try:
-> 2237             result = self._expression.parseString(s)
   2238         except ParseBaseException as err:

C:\ProgramData\Anaconda3\lib\site-packages\pyparsing\core.py in parse_st
ring(self, instring, parse_all, parseAll)
   1140                 # catch and re-raise exception from here, cleari
ng out pyparsing internal stack trace
-> 1141                 raise exc.with_traceback(None)
   1142         else:

ParseException: Expected end of text, found '$'  (at char 0), (line:1, c
ol:1)
```

In [9]:
```
# Save cleaned data to a new CSV file
data.to_csv("cleaned_data.csv", index=False)
```

In [11]:
```
pip install geopy
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Collecting geopy
  Downloading geopy-2.4.1-py3-none-any.whl (125 kB)
     ---------------------------------------- 125.4/125.4 kB 3.6 MB/s eta 0:
00:00
Collecting geographiclib<3,>=1.52
  Downloading geographiclib-2.0-py3-none-any.whl (40 kB)
     ---------------------------------------- 40.3/40.3 kB 2.0 MB/s eta 0:
00:00
Installing collected packages: geographiclib, geopy
Successfully installed geographiclib-2.0 geopy-2.4.1
Note: you may need to restart the kernel to use updated packages.
```

In [12]:
```
from sklearn.preprocessing import StandardScaler, LabelEncoder
from geopy.distance import geodesic
import numpy as np

# Create a copy of the dataset for preprocessing
preprocessed_data = data.copy()

# Step 1: Handle Missing Values
for col in preprocessed_data.columns:
    if preprocessed_data[col].isnull().sum() > 0:
        if preprocessed_data[col].dtype in ["float64", "int64"]:
            preprocessed_data[col].fillna(preprocessed_data[col].mean(), inp
        else:
            preprocessed_data[col].fillna("Unknown", inplace=True)

# Step 2: Normalize Text Data
preprocessed_data["Name"] = preprocessed_data["Name"].str.strip().str.lower

# Step 3: Convert Categorical Data
if 'PriceRange' in preprocessed_data.columns:
    price_mapping = {"Low": 1, "Medium": 2, "High": 3, "Very High": 4}
    preprocessed_data["PriceRangeNumeric"] = preprocessed_data["PriceRange"

# Step 4: Handle Date/Time (if applicable)
if 'Open_time' in preprocessed_data.columns:
    preprocessed_data["Open_time"] = pd.to_datetime(preprocessed_data["Open_

# Step 5: Feature Engineering
# Example: Compute Distance from Village restaurant
village_coords = (40.7683, -73.5258)  # Replace with Village's latitude and
if "Latitude" in preprocessed_data.columns and "Longitude" in preprocessed_
    preprocessed_data["DistanceFromVillage"] = preprocessed_data.apply(
        lambda row: geodesic(village_coords, (row["Latitude"], row["Longitu
    )

# Example: Adjust prices (normalize prices for competitors)
if "Price" in preprocessed_data.columns:
    preprocessed_data["NormalizedPrice"] = (
        preprocessed_data["Price"] / preprocessed_data["Price"].max()
    )

# Step 6: Standardize Numerical Features
scaler = StandardScaler()
numerical_cols = preprocessed_data.select_dtypes(include=["float64", "int64
preprocessed_data[numerical_cols] = scaler.fit_transform(preprocessed_data[

# Step 7: Encode Target or Label Data
if 'PriceRangeNumeric' in preprocessed_data.columns:
    encoder = LabelEncoder()
    preprocessed_data['PriceRangeEncoded'] = encoder.fit_transform(preproce

# Display the first few rows of the preprocessed data
print(preprocessed_data.head())
```

```
             Name           Phone  \
0    trullo d’oro  (516) 719-0070
1    kirazu sushi  (516) 933-8038
2   chili & curry  (516) 932-9180
3  burger village  (516) 597-5336
4     kinya ramen  (516) 719-0388

                                          Address  \
0          294 N Broadway, Hicksville, NY, 11801, US
1  107 Stewart Ave, Ste 9, Hicksville, NY, 11801, US
2         106 Woodbury Rd, Hicksville, NY, 11801, US
3       216 Broadway Mall, Hicksville, NY, 11801, US
4  100 West Old Country Rd, Hicksville, NY, 11801...

                        Email                              Website  ServiceA
rea  \
0  eat@villagesoulofindia.com         http://www.trullo-doro.com (http://w
ww.trullo-doro.com)          NaN
1  eat@villagesoulofindia.com         http://www.kirazusushi.com (http://w
ww.kirazusushi.com)          NaN
2  eat@villagesoulofindia.com  https://chiliandcurry.square.site (https://
chiliandcurry.square.site)          NaN
3  eat@villagesoulofindia.com            http://burgervillage.com (http://b
urgervillage.com)          NaN
4               info@kinya.us                    https://kinya.us (https://
kinya.us)          NaN

                                  Instagram  \
0  https://www.instagram.com/burgervillage (https://www.instagram.com/burg
ervillage)
1  https://www.instagram.com/burgervillage (https://www.instagram.com/burg
ervillage)
2  https://www.instagram.com/burgervillage (https://www.instagram.com/burg
ervillage)
3  https://www.instagram.com/burgervillage (https://www.instagram.com/burg
ervillage)
4     https://www.instagram.com/kinyaramen (https://www.instagram.com/kiny
aramen)

                                           Facebook  \
0      https://www.facebook.com/trullodorolongisland (https://www.faceboo
k.com/trullodorolongisland)
1                    https://www.facebook.com/pages (https://www.faceboo
k.com/pages)
2                     https://www.facebook.com/IHOP (https://www.faceboo
k.com/IHOP)
3             https://www.facebook.com/burgervillage (https://www.faceboo
k.com/burgervillage)
4  https://www.facebook.com/kinyaramensushibar.of... (https://www.faceboo
k.com/kinyaramensushibar.of...)

                    Twitter                                Linkedin  ...  \
0  https://twitter.com/IHOP (https://twitter.com/IHOP)  https://www.linked
in.com/company/ihop (https://www.linkedin.com/company/ihop)  ...
1  https://twitter.com/IHOP (https://twitter.com/IHOP)  https://www.linked
in.com/company/ihop (https://www.linkedin.com/company/ihop)  ...
2  https://twitter.com/IHOP (https://twitter.com/IHOP)  https://www.linked
in.com/company/ihop (https://www.linkedin.com/company/ihop)  ...
3  https://twitter.com/IHOP (https://twitter.com/IHOP)  https://www.linked
in.com/company/ihop (https://www.linkedin.com/company/ihop)  ...
4  https://twitter.com/IHOP (https://twitter.com/IHOP)  https://www.linked
```

```
in.com/company/ihop (https://www.linkedin.com/company/ihop)  ...

    IsAd Open_time  ParentBusiness   Ranking RenderAdInfo  ServicePricing
\
0  False       NaT             NaN -1.566699        False             NaN
1  False       NaT             NaN -1.218544        False             NaN
2  False       NaT             NaN -0.870388        False             NaN
3  False       NaT             NaN -0.522233        False             NaN
4  False       NaT             NaN -0.174078        False             NaN

                                             Snippet PriceRangeNumeric  \
0  We just had a [[HIGHLIGHT]]wonderful meal at t...               NaN
1  [[HIGHLIGHT]]The BEST sushi in town[[ENDHIGHLI...               NaN
2  I'm actually not new to Chili & Curry, I disco...               NaN
3  Came here on Saturday night and the place did ...               NaN
4  Food was good for the price. And nick was very...               NaN

  DistanceFromVillage PriceRangeEncoded
0            0.387001                 0
1            2.330559                 0
2           -0.332792                 0
3           -0.457517                 0
4           -0.377297                 0

[5 rows x 42 columns]

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\extmath.py:985: R
untimeWarning: invalid value encountered in true_divide
  updated_mean = (last_sum + new_sum) / updated_sample_count
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\extmath.py:990: R
untimeWarning: invalid value encountered in true_divide
  T = new_sum / new_sample_count
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\extmath.py:1020:
RuntimeWarning: invalid value encountered in true_divide
  new_unnormalized_variance -= correction ** 2 / new_sample_count
```

In [13]:

```
# Drop columns irrelevant to ML task
columns_to_drop = [
    "Phone", "Email", "Website", "Instagram", "Facebook", "Twitter",
    "Linkedin", "ParentBusiness", "Snippet", "IsAd", "RenderAdInfo"
]
preprocessed_data = preprocessed_data.drop(columns=columns_to_drop, errors=

# Handle NaN values in important columns
# Fill numerical columns with mean
numerical_cols = preprocessed_data.select_dtypes(include=["float64", "int64
preprocessed_data[numerical_cols] = preprocessed_data[numerical_cols].fillna

# Fill categorical columns with mode
categorical_cols = preprocessed_data.select_dtypes(include=["object"]).colu
preprocessed_data[categorical_cols] = preprocessed_data[categorical_cols].f
```

In [14]:
```
# Parse 'Open_time' if it's essential to the task
if "Open_time" in preprocessed_data.columns:
    preprocessed_data["Open_time"] = pd.to_datetime(preprocessed_data["Open_

# Extract useful time features if required
if "Open_time" in preprocessed_data.columns:
    preprocessed_data["OpeningHour"] = preprocessed_data["Open_time"].dt.hou
    preprocessed_data["OpeningDay"] = preprocessed_data["Open_time"].dt.day(
```

In [15]:
```
from sklearn.preprocessing import StandardScaler

# Standardize numerical features
scaler = StandardScaler()
numerical_cols = preprocessed_data.select_dtypes(include=["float64", "int64
preprocessed_data[numerical_cols] = scaler.fit_transform(preprocessed_data[n
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\extmath.py:985: R
untimeWarning: invalid value encountered in true_divide
  updated_mean = (last_sum + new_sum) / updated_sample_count
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\extmath.py:990: R
untimeWarning: invalid value encountered in true_divide
  T = new_sum / new_sample_count
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\extmath.py:1020:
RuntimeWarning: invalid value encountered in true_divide
  new_unnormalized_variance -= correction ** 2 / new_sample_count
```

In [16]:
```
from sklearn.preprocessing import LabelEncoder

# Encode categorical features
encoder = LabelEncoder()
categorical_cols = preprocessed_data.select_dtypes(include=["object"]).colu
for col in categorical_cols:
    preprocessed_data[col] = encoder.fit_transform(preprocessed_data[col])
```

In [17]:
```
# Verify the final preprocessed data
print("Final Preprocessed Data Overview:")
print(preprocessed_data.info())

# Display sample rows
print(preprocessed_data.head())
```

```
Final Preprocessed Data Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 33 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   Name                              10 non-null     int32
 1   Address                           10 non-null     int32
 2   ServiceArea                       0 non-null      float64
 3   Youtube                           10 non-null     int32
 4   BusinessUrl                       10 non-null     int32
 5   Rating                            10 non-null     float64
 6   ReviewCount                       10 non-null     float64
 7   PriceRange                        10 non-null     int32
 8   Longitude                         10 non-null     float64
 9   Latitude                          10 non-null     float64
 10  Alias                             10 non-null     int32
 11  BizId                             10 non-null     int32
 12  BusinessSectionUrls_open_hours    10 non-null     int32
 13  BusinessSectionUrls_reviews       10 non-null     int32
 14  Categories_0_title                10 non-null     int32
 15  Categories_0_url                  10 non-null     int32
 16  Categories_1_title                10 non-null     int32
 17  Categories_1_url                  10 non-null     int32
 18  Categories_2_title                10 non-null     int32
 19  Categories_2_url                  10 non-null     int32
 20  Categories_3_title                10 non-null     int32
 21  Categories_3_url                  10 non-null     int32
 22  Categories_4_title                10 non-null     int32
 23  Categories_4_url                  10 non-null     int32
 24  FormattedAddress                  0 non-null      float64
 25  Open_time                         0 non-null      datetime64[ns]
 26  Ranking                           10 non-null     float64
 27  ServicePricing                    0 non-null      float64
 28  PriceRangeNumeric                 0 non-null      float64
 29  DistanceFromVillage               10 non-null     float64
 30  PriceRangeEncoded                 10 non-null     float64
 31  OpeningHour                       0 non-null      float64
 32  OpeningDay                        0 non-null      float64
dtypes: datetime64[ns](1), float64(13), int32(19)
memory usage: 2.0 KB
None
   Name  Address  ServiceArea  Youtube  BusinessUrl    Rating  ReviewCount
\
0     8        6          NaN        0            8  0.045222    -0.366696
1     7        2          NaN        0            7  0.723545    -0.106513
2     1        1          NaN        0            1 -0.180886     2.702654
3     0        5          NaN        0            0 -0.406994     0.287828
4     6        0          NaN        0            6  0.723545    -0.086186

   PriceRange  Longitude  Latitude  ...  Categories_4_url  FormattedAddres
s  \
0           1  -0.576548  1.361496  ...                 1               Na
N
1           1  -2.145819 -2.159696  ...                 1               Na
N
2           1   0.839482  0.534358  ...                 1               Na
N
3           1  -0.489585  0.633888  ...                 2               Na
N
4           1  -0.422302  0.745089  ...                 2               Na
```

```
N

   Open_time   Ranking  ServicePricing  PriceRangeNumeric  \
0        NaT -1.566699             NaN                NaN
1        NaT -1.218544             NaN                NaN
2        NaT -0.870388             NaN                NaN
3        NaT -0.522233             NaN                NaN
4        NaT -0.174078             NaN                NaN

   DistanceFromVillage  PriceRangeEncoded  OpeningHour  OpeningDay
0             0.387001                0.0          NaN         NaN
1             2.330559                0.0          NaN         NaN
2            -0.332792                0.0          NaN         NaN
3            -0.457517                0.0          NaN         NaN
4            -0.377297                0.0          NaN         NaN

[5 rows x 33 columns]
```

In [18]:
```
columns_to_drop = [
    "ServiceArea", "FormattedAddress", "ServicePricing",
    "PriceRangeNumeric", "Open_time", "OpeningHour", "OpeningDay"
]
preprocessed_data = preprocessed_data.drop(columns=columns_to_drop, errors=
```

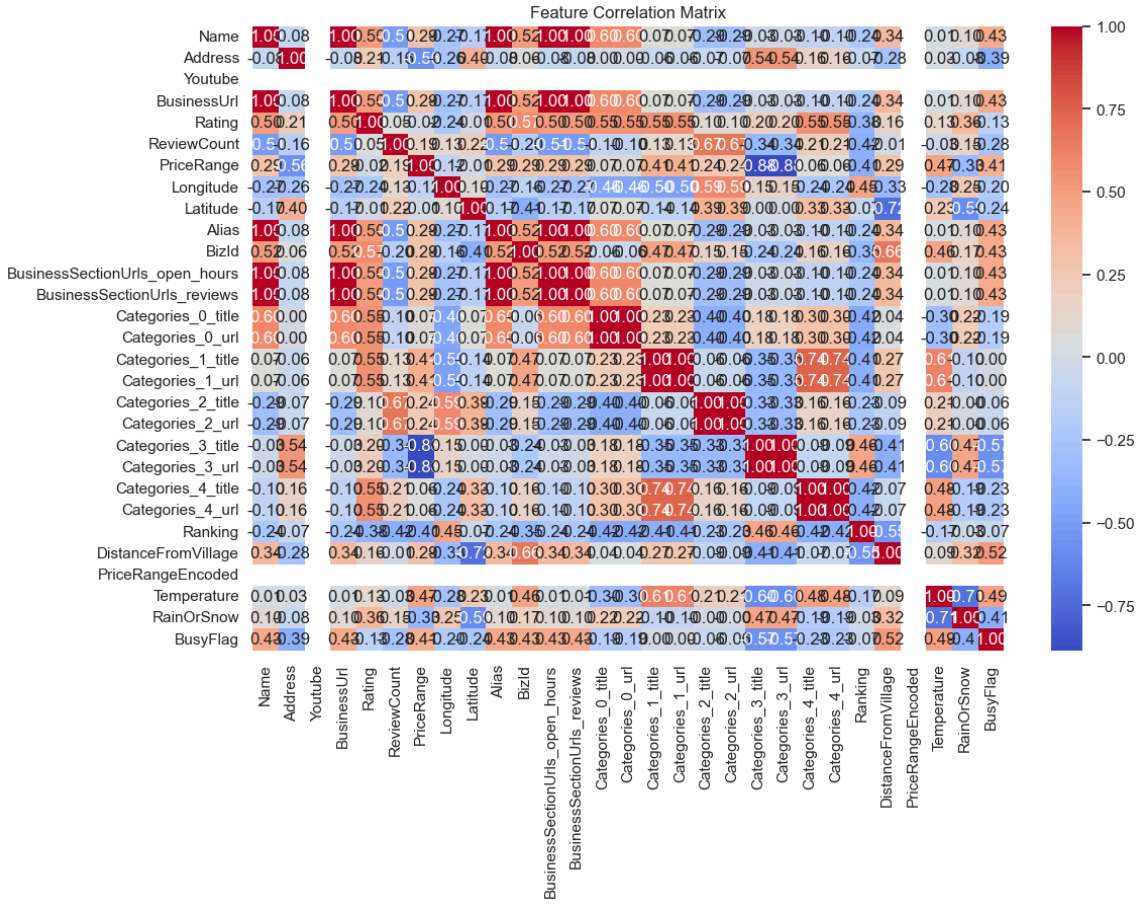
In [19]:
```
# Example placeholders (replace with actual data)
preprocessed_data["Temperature"] = [50, 45, 32, 60, 55, 48, 35, 65, 30, 40]
preprocessed_data["RainOrSnow"] = [0, 1, 1, 0, 0, 1, 1, 0, 1, 0]  # 0 = No,
preprocessed_data["BusyFlag"] = [1, 1, 0, 0, 1, 1, 0, 1, 0, 1]  # 0 = Not b
```

In [20]:
```
print(preprocessed_data.info())
print(preprocessed_data.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 29 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Name                             10 non-null     int32  
 1   Address                          10 non-null     int32  
 2   Youtube                          10 non-null     int32  
 3   BusinessUrl                      10 non-null     int32  
 4   Rating                           10 non-null     float64
 5   ReviewCount                      10 non-null     float64
 6   PriceRange                       10 non-null     int32  
 7   Longitude                        10 non-null     float64
 8   Latitude                         10 non-null     float64
 9   Alias                            10 non-null     int32  
 10  BizId                            10 non-null     int32  
 11  BusinessSectionUrls_open_hours   10 non-null     int32  
 12  BusinessSectionUrls_reviews      10 non-null     int32  
 13  Categories_0_title               10 non-null     int32  
 14  Categories_0_url                 10 non-null     int32  
 15  Categories_1_title               10 non-null     int32  
 16  Categories_1_url                 10 non-null     int32  
 17  Categories_2_title               10 non-null     int32  
 18  Categories_2_url                 10 non-null     int32  
 19  Categories_3_title               10 non-null     int32  
 20  Categories_3_url                 10 non-null     int32  
 21  Categories_4_title               10 non-null     int32  
 22  Categories_4_url                 10 non-null     int32  
 23  Ranking                          10 non-null     float64
 24  DistanceFromVillage              10 non-null     float64
 25  PriceRangeEncoded                10 non-null     float64
 26  Temperature                      10 non-null     int64  
 27  RainOrSnow                       10 non-null     int64  
 28  BusyFlag                         10 non-null     int64  
dtypes: float64(7), int32(19), int64(3)
memory usage: 1.6 KB
None
   Name  Address  Youtube  BusinessUrl    Rating  ReviewCount  PriceRange
\
0     8        6        0            8  0.045222    -0.366696           1
1     7        2        0            7  0.723545    -0.106513           1
2     1        1        0            1 -0.180886     2.702654           1
3     0        5        0            0 -0.406994     0.287828           1
4     6        0        0            6  0.723545    -0.086186           1

   Longitude  Latitude  Alias  ...  Categories_3_title  Categories_3_url
\
0  -0.576548  1.361496      8  ...                   0                 0
1  -2.145819 -2.159696      7  ...                   0                 0
2   0.839482  0.534358      1  ...                   0                 0
3  -0.489585  0.633888      0  ...                   0                 0
4  -0.422302  0.745089      6  ...                   0                 0

   Categories_4_title  Categories_4_url   Ranking  DistanceFromVillage  \
0                   1                 1 -1.566699             0.387001   
1                   1                 1 -1.218544             2.330559   
2                   1                 1 -0.870388            -0.332792   
3                   2                 2 -0.522233            -0.457517   
4                   2                 2 -0.174078            -0.377297   

   PriceRangeEncoded  Temperature  RainOrSnow  BusyFlag
```

```
0                0.0                50              0          1
1                0.0                45              1          1
2                0.0                32              1          0
3                0.0                60              0          0
4                0.0                55              0          1

[5 rows x 29 columns]
```

In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt

# Compute correlation matrix
correlation_matrix = preprocessed_data.corr()

# Plot heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title("Feature Correlation Matrix")
plt.show()
```



In [22]:
```
from sklearn.feature_selection import VarianceThreshold

# Set a threshold for variance (e.g., 0.01)
selector = VarianceThreshold(threshold=0.01)
high_variance_data = selector.fit_transform(preprocessed_data)

# Get selected feature names
selected_features = preprocessed_data.columns[selector.get_support()]
print("Selected Features (Variance Threshold):", selected_features)
```

```
Selected Features (Variance Threshold): Index(['Name', 'Address', 'Busines
sUrl', 'Rating', 'ReviewCount', 'PriceRange',
       'Longitude', 'Latitude', 'Alias', 'BizId',
       'BusinessSectionUrls_open_hours', 'BusinessSectionUrls_reviews',
       'Categories_0_title', 'Categories_0_url', 'Categories_1_title',
       'Categories_1_url', 'Categories_2_title', 'Categories_2_url',
       'Categories_3_title', 'Categories_3_url', 'Categories_4_title',
       'Categories_4_url', 'Ranking', 'DistanceFromVillage', 'Temperatur
e',
       'RainOrSnow', 'BusyFlag'],
      dtype='object')
```

In [24]:
```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestRegressor

# Replace 'target_column' with the actual name of your target variable
target_column = "ActualTargetColumnName"  # Update this to match your data

# Define the model
model = RandomForestRegressor(random_state=42)

# Check if the target column exists
if target_column not in preprocessed_data.columns:
    raise ValueError(f"Target column '{target_column}' not found in dataset

# Perform RFE
X = preprocessed_data.drop(target_column, axis=1)
y = preprocessed_data[target_column]
rfe = RFE(estimator=model, n_features_to_select=10)  # Select top 10 featur
rfe.fit(X, y)

# Get selected features
rfe_selected_features = X.columns[rfe.support_]
print("Selected Features (RFE):", rfe_selected_features)
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\417814623.py in <module>
     10 # Check if the target column exists
     11 if target_column not in preprocessed_data.columns:
---> 12     raise ValueError(f"Target column '{target_column}' not found i
n dataset.")
     13
     14 # Perform RFE

ValueError: Target column 'ActualTargetColumnName' not found in dataset.
```

In [25]:
```
target_column = "Price"  # Replace 'Price' with the actual target column nar
```

In [26]:
```
print(preprocessed_data.columns)
```

```
Index(['Name', 'Address', 'Youtube', 'BusinessUrl', 'Rating', 'ReviewCoun
t',
       'PriceRange', 'Longitude', 'Latitude', 'Alias', 'BizId',
       'BusinessSectionUrls_open_hours', 'BusinessSectionUrls_reviews',
       'Categories_0_title', 'Categories_0_url', 'Categories_1_title',
       'Categories_1_url', 'Categories_2_title', 'Categories_2_url',
       'Categories_3_title', 'Categories_3_url', 'Categories_4_title',
       'Categories_4_url', 'Ranking', 'DistanceFromVillage',
       'PriceRangeEncoded', 'Temperature', 'RainOrSnow', 'BusyFlag'],
      dtype='object')
```

In [27]:
```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestRegressor

# Replace 'Price' with your actual target column name
target_column = "Price"  # Update this to match your actual data

# Define the model
model = RandomForestRegressor(random_state=42)

# Check if the target column exists
if target_column not in preprocessed_data.columns:
    raise ValueError(f"Target column '{target_column}' not found in dataset

# Perform RFE
X = preprocessed_data.drop(target_column, axis=1)
y = preprocessed_data[target_column]
rfe = RFE(estimator=model, n_features_to_select=10)  # Select top 10 featur
rfe.fit(X, y)

# Get selected features
rfe_selected_features = X.columns[rfe.support_]
print("Selected Features (RFE):", rfe_selected_features)
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\1492160400.py in <module>
     10 # Check if the target column exists
     11 if target_column not in preprocessed_data.columns:
---> 12     raise ValueError(f"Target column '{target_column}' not found i
n dataset.")
     13
     14 # Perform RFE

ValueError: Target column 'Price' not found in dataset.
```

In [28]:
```
print(preprocessed_data.columns)
```

```
Index(['Name', 'Address', 'Youtube', 'BusinessUrl', 'Rating', 'ReviewCoun
t',
       'PriceRange', 'Longitude', 'Latitude', 'Alias', 'BizId',
       'BusinessSectionUrls_open_hours', 'BusinessSectionUrls_reviews',
       'Categories_0_title', 'Categories_0_url', 'Categories_1_title',
       'Categories_1_url', 'Categories_2_title', 'Categories_2_url',
       'Categories_3_title', 'Categories_3_url', 'Categories_4_title',
       'Categories_4_url', 'Ranking', 'DistanceFromVillage',
       'PriceRangeEncoded', 'Temperature', 'RainOrSnow', 'BusyFlag'],
      dtype='object')
```

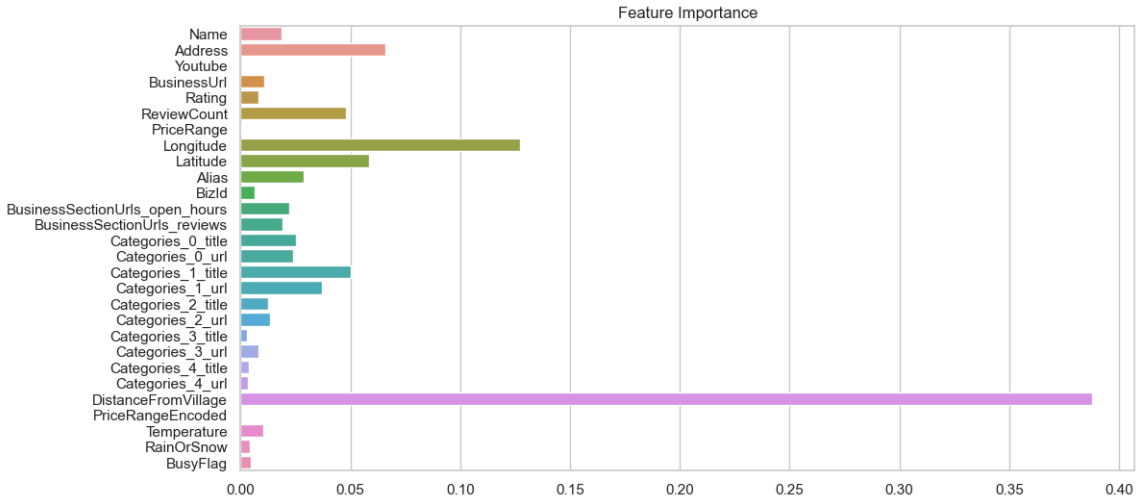
In [29]:
```
# Print the column names to verify the target column exists
print("Column names in the dataset:", preprocessed_data.columns)

# Replace 'Price' with the actual name of your target column after verifying
target_column = "ActualTargetColumnName"  # Replace with correct column name

# Define the model
model = RandomForestRegressor(random_state=42)

# Check if the target column exists
if target_column not in preprocessed_data.columns:
    raise ValueError(f"Target column '{target_column}' not found in dataset

# Perform RFE
X = preprocessed_data.drop(target_column, axis=1)
y = preprocessed_data[target_column]
rfe = RFE(estimator=model, n_features_to_select=10)  # Select top 10 feature
rfe.fit(X, y)

# Get selected features
rfe_selected_features = X.columns[rfe.support_]
print("Selected Features (RFE):", rfe_selected_features)
```

```
Column names in the dataset: Index(['Name', 'Address', 'Youtube', 'Busines
sUrl', 'Rating', 'ReviewCount',
       'PriceRange', 'Longitude', 'Latitude', 'Alias', 'BizId',
       'BusinessSectionUrls_open_hours', 'BusinessSectionUrls_reviews',
       'Categories_0_title', 'Categories_0_url', 'Categories_1_title',
       'Categories_1_url', 'Categories_2_title', 'Categories_2_url',
       'Categories_3_title', 'Categories_3_url', 'Categories_4_title',
       'Categories_4_url', 'Ranking', 'DistanceFromVillage',
       'PriceRangeEncoded', 'Temperature', 'RainOrSnow', 'BusyFlag'],
      dtype='object')

---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\271399286.py in <module>
     10 # Check if the target column exists
     11 if target_column not in preprocessed_data.columns:
---> 12     raise ValueError(f"Target column '{target_column}' not found i
n dataset.")
     13
     14 # Perform RFE

ValueError: Target column 'ActualTargetColumnName' not found in dataset.
```

In [39]:
```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestRegressor

# Replace with the actual target column, for example 'Ranking' or another re
target_column = "Ranking"  # Replace with the actual target column name afte

# Define the model
model = RandomForestRegressor(random_state=42)

# Check if the target column exists
if target_column not in preprocessed_data.columns:
    raise ValueError(f"Target column '{target_column}' not found in dataset

# Perform RFE
X = preprocessed_data.drop(target_column, axis=1)
y = preprocessed_data[target_column]
rfe = RFE(estimator=model, n_features_to_select=10)  # Select top 10 featur
rfe.fit(X, y)

# Get selected features
rfe_selected_features = X.columns[rfe.support_]
print("Selected Features (RFE):", rfe_selected_features)
```

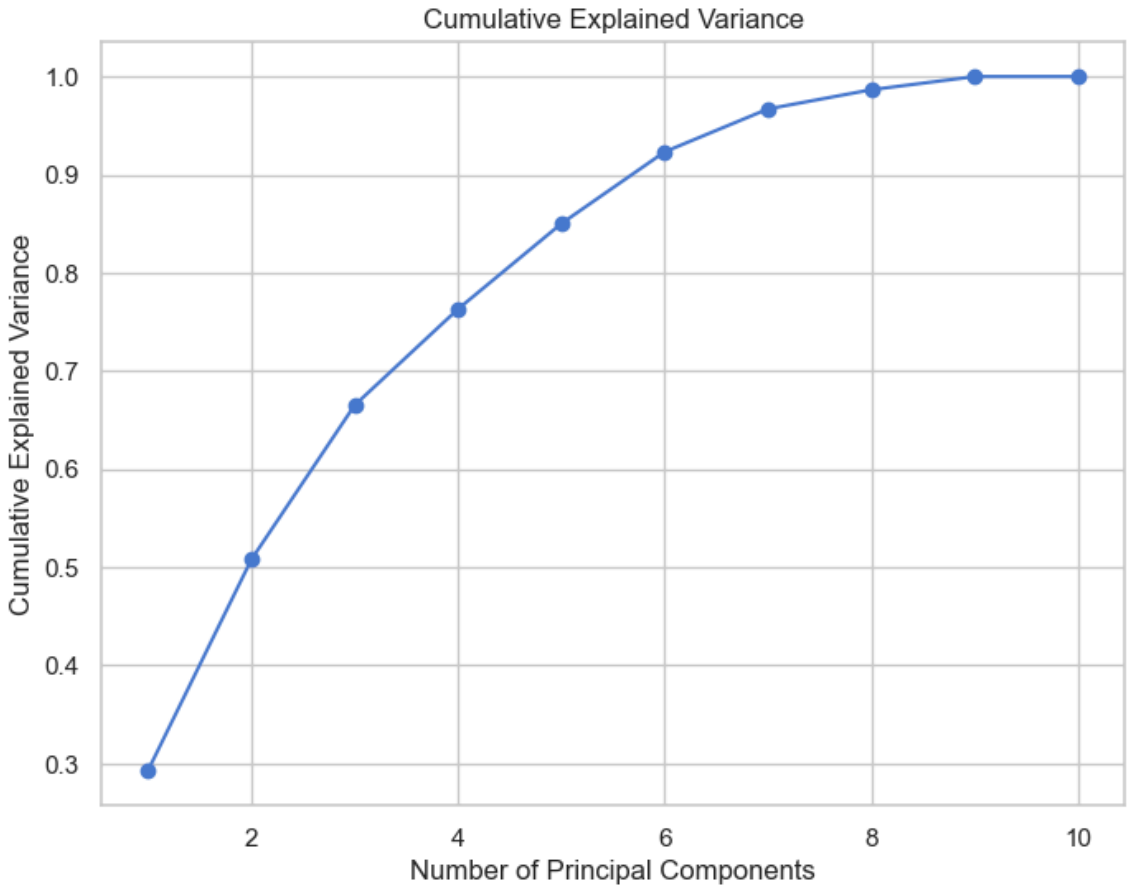
```
Selected Features (RFE): Index(['Address', 'BusinessUrl', 'ReviewCount',
'Longitude', 'Latitude',
       'BusinessSectionUrls_open_hours', 'Categories_0_url',
       'Categories_1_url', 'Categories_2_url', 'DistanceFromVillage'],
      dtype='object')
```

In [44]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor

# Define the actual target column name
target_column = "Ranking"  # Replace this with your actual target column nar

# Fit RandomForestRegressor for feature importance
model = RandomForestRegressor(random_state=42)
model.fit(preprocessed_data.drop(target_column, axis=1), preprocessed_data[

# Plot feature importance
importances = model.feature_importances_
features = preprocessed_data.drop(target_column, axis=1).columns

plt.figure(figsize=(12, 6))
sns.barplot(x=importances, y=features)
plt.title("Feature Importance")
plt.show()
```



In [43]:
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(preprocessed_data)

# Apply PCA
pca = PCA(n_components=10)  # Select the number of principal components
principal_components = pca.fit_transform(scaled_data)

# Explained variance ratio
print("Explained Variance Ratio:", pca.explained_variance_ratio_)

# Plot cumulative explained variance
plt.figure(figsize=(8, 6))
plt.plot(range(1, len(pca.explained_variance_ratio_) + 1),
         pca.explained_variance_ratio_.cumsum(), marker='o')
plt.title("Cumulative Explained Variance")
plt.xlabel("Number of Principal Components")
plt.ylabel("Cumulative Explained Variance")
plt.show()
```

```
Explained Variance Ratio: [2.92648505e-01 2.15024833e-01 1.57425072e-01 9.
79498905e-02
 8.70924126e-02 7.31684814e-02 4.37174845e-02 1.95767373e-02
 1.33965834e-02 5.07082154e-33]
```



In [46]:
```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset
file_path = r"C:\Users\shubh\Downloads\results1.csv"  # Use the raw string j
data = pd.read_csv(file_path)

# Specify the target and features
target_column = "Ranking"  # Replace this with your actual target column nar
features = data.drop(target_column, axis=1)  # Drop the target column
target = data[target_column]  # Select the target column

# Perform train-test split
X_train, X_test, y_train, y_test = train_test_split(
    features, target, test_size=0.2, random_state=42
)

# Display the shapes of the splits
print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
X_train shape: (8, 38)
X_test shape: (2, 38)
y_train shape: (8,)
y_test shape: (2,)
```

In [49]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load the dataset
file_path = r"C:\Users\shubh\Downloads\results1.csv"
data = pd.read_csv(file_path)

# Inspect the data
print(data.info())

# Convert categorical columns to numerical values
categorical_columns = data.select_dtypes(include=['object']).columns
print(f"Categorical Columns: {categorical_columns}")

# Apply Label Encoding for simplicity (or use OneHotEncoder for more flexib
label_encoders = {}
for col in categorical_columns:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Check for missing values
print(data.isnull().sum())

# Fill or drop missing values if needed
data = data.fillna(0)  # Example: Replace NaNs with 0

# Define features and target
target_column = "Ranking"  # Replace this with your actual target column nam
features = data.drop(target_column, axis=1)
target = data[target_column]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_s

print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 39 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   Name                              10 non-null     object 
 1   Phone                             10 non-null     object 
 2   Address                           8 non-null      object 
 3   Email                             4 non-null      object 
 4   Website                           9 non-null      object 
 5   ServiceArea                       0 non-null      float64
 6   Instagram                         3 non-null      object 
 7   Facebook                          7 non-null      object 
 8   Twitter                           1 non-null      object 
 9   Linkedin                          1 non-null      object 
 10  Youtube                           1 non-null      object 
 11  BusinessUrl                       10 non-null     object 
 12  Rating                            10 non-null     float64
 13  ReviewCount                       10 non-null     int64  
 14  PriceRange                        9 non-null      object 
 15  Longitude                         10 non-null     float64
 16  Latitude                          10 non-null     float64
 17  Alias                             10 non-null     object 
 18  BizId                             10 non-null     object 
 19  BusinessSectionUrls_open_hours    10 non-null     object 
 20  BusinessSectionUrls_reviews       10 non-null     object 
 21  Categories_0_title                10 non-null     object 
 22  Categories_0_url                  10 non-null     object 
 23  Categories_1_title                10 non-null     object 
 24  Categories_1_url                  10 non-null     object 
 25  Categories_2_title                10 non-null     object 
 26  Categories_2_url                  10 non-null     object 
 27  Categories_3_title                9 non-null      object 
 28  Categories_3_url                  9 non-null      object 
 29  Categories_4_title                7 non-null      object 
 30  Categories_4_url                  7 non-null      object 
 31  FormattedAddress                  0 non-null      float64
 32  IsAd                              10 non-null     bool   
 33  Open_time                         0 non-null      float64
 34  ParentBusiness                    0 non-null      float64
 35  Ranking                           10 non-null     int64  
 36  RenderAdInfo                      10 non-null     bool   
 37  ServicePricing                    0 non-null      float64
 38  Snippet                           10 non-null     object 
dtypes: bool(2), float64(8), int64(2), object(27)
memory usage: 3.0+ KB
None
Categorical Columns: Index(['Name', 'Phone', 'Address', 'Email', 'Websit
e', 'Instagram', 'Facebook',
       'Twitter', 'Linkedin', 'Youtube', 'BusinessUrl', 'PriceRange', 'Ali
as',
       'BizId', 'BusinessSectionUrls_open_hours',
       'BusinessSectionUrls_reviews', 'Categories_0_title', 'Categories_0_
url',
       'Categories_1_title', 'Categories_1_url', 'Categories_2_title',
       'Categories_2_url', 'Categories_3_title', 'Categories_3_url',
       'Categories_4_title', 'Categories_4_url', 'Snippet'],
      dtype='object')
Name                              0
Phone                             0
Address                           0
```

```
Email                              0
Website                            0
ServiceArea                       10
Instagram                          0
Facebook                           0
Twitter                            0
Linkedin                           0
Youtube                            0
BusinessUrl                        0
Rating                             0
ReviewCount                        0
PriceRange                         0
Longitude                          0
Latitude                           0
Alias                              0
BizId                              0
BusinessSectionUrls_open_hours     0
BusinessSectionUrls_reviews        0
Categories_0_title                 0
Categories_0_url                   0
Categories_1_title                 0
Categories_1_url                   0
Categories_2_title                 0
Categories_2_url                   0
Categories_3_title                 0
Categories_3_url                   0
Categories_4_title                 0
Categories_4_url                   0
FormattedAddress                  10
IsAd                               0
Open_time                         10
ParentBusiness                    10
Ranking                            0
RenderAdInfo                       0
ServicePricing                    10
Snippet                            0
dtype: int64
X_train shape: (8, 38)
X_test shape: (2, 38)
```

In [50]:
```
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Train the Support Vector Regressor
svm_model = SVR(kernel='rbf', C=1.0, epsilon=0.1)
svm_model.fit(X_train, y_train)

# Make predictions
y_pred_svm = svm_model.predict(X_test)

# Evaluate the model
mae_svm = mean_absolute_error(y_test, y_pred_svm)
mse_svm = mean_squared_error(y_test, y_pred_svm)
r2_svm = r2_score(y_test, y_pred_svm)

print("SVM Performance:")
print(f"Mean Absolute Error: {mae_svm}")
print(f"Mean Squared Error: {mse_svm}")
print(f"R² Score: {r2_svm}")
```

```
SVM Performance:
Mean Absolute Error: 3.3016845814084763
Mean Squared Error: 10.92403679265615
R² Score: 0.10824189447704902
```

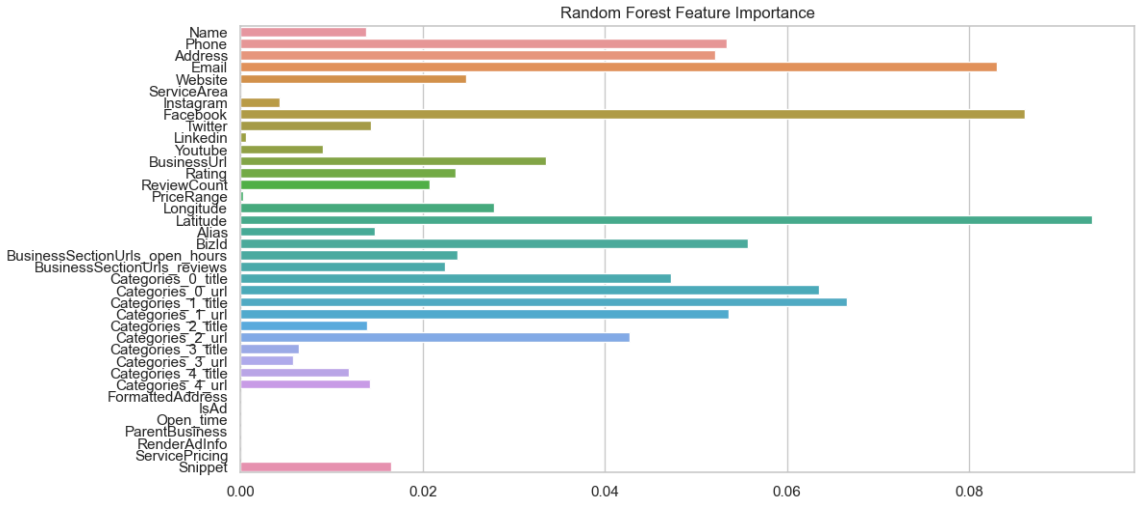
In [51]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVR

# Define the parameter grid
param_grid = {
    'C': [0.1, 1, 10, 100],
    'gamma': ['scale', 'auto', 0.1, 0.01],
    'epsilon': [0.1, 0.2, 0.5]
}

# GridSearchCV for SVR
grid_search = GridSearchCV(SVR(kernel='rbf'), param_grid, cv=5, scoring='r2
grid_search.fit(X_train, y_train)

# Best parameters and performance
print("Best Parameters:", grid_search.best_params_)
print("Best R² Score on Training Data:", grid_search.best_score_)

# Evaluate on test data
best_svr = grid_search.best_estimator_
y_pred_tuned = best_svr.predict(X_test)

print("Tuned SVM Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_tuned))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_tuned))
print("R² Score:", r2_score(y_test, y_pred_tuned))
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
[CV] END .....................C=0.1, epsilon=0.1, gamma=scale; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=scale; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=scale; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=scale; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=scale; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=auto; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=auto; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=auto; total time
=   0.0s
[CV] END .....................C=0.1, epsilon=0.1, gamma=auto; total time
=   0.0s
```

In [52]:
```
from sklearn.ensemble import RandomForestRegressor

# Train Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=200, max_depth=10, random_sta
rf_model.fit(X_train, y_train)

# Evaluate Random Forest
y_pred_rf = rf_model.predict(X_test)
print("Random Forest Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_rf))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_rf))
print("R² Score:", r2_score(y_test, y_pred_rf))
```

```
Random Forest Performance:
Mean Absolute Error: 3.8775
Mean Squared Error: 15.368512500000001
R² Score: -0.25457244897959197
```

In [53]:
```
# Plot feature importance
import matplotlib.pyplot as plt
import seaborn as sns

importances = rf_model.feature_importances_
features = X_train.columns

plt.figure(figsize=(12, 6))
sns.barplot(x=importances, y=features)
plt.title("Random Forest Feature Importance")
plt.show()
```



In [58]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [5, 10, 20, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Perform grid search
grid_search_rf = GridSearchCV(RandomForestRegressor(random_state=42), param_
grid_search_rf.fit(X_train, y_train)

# Best parameters and performance
print("Best Parameters:", grid_search_rf.best_params_)
print("Best R² Score on Training Data:", grid_search_rf.best_score_)

# Evaluate on test data
best_rf_model = grid_search_rf.best_estimator_
y_pred_rf_tuned = best_rf_model.predict(X_test)

print("Tuned Random Forest Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_rf_tuned))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_rf_tuned))
print("R² Score:", r2_score(y_test, y_pred_rf_tuned))
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100; total time=   0.0s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100; total time=   0.0s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100; total time=   0.0s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100; total time=   0.0s
[CV] END max_depth=5, min_samples_leaf=1, min_samples_split=2, n_estimat
ors=100; total time=   0.0s

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.p
y:796: UndefinedMetricWarning: R^2 score is not well-defined with less t
han two samples.
  warnings.warn(msg, UndefinedMetricWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.p
y:796: UndefinedMetricWarning: R^2 score is not well-defined with less t
han two samples.
  warnings.warn(msg, UndefinedMetricWarning)
```

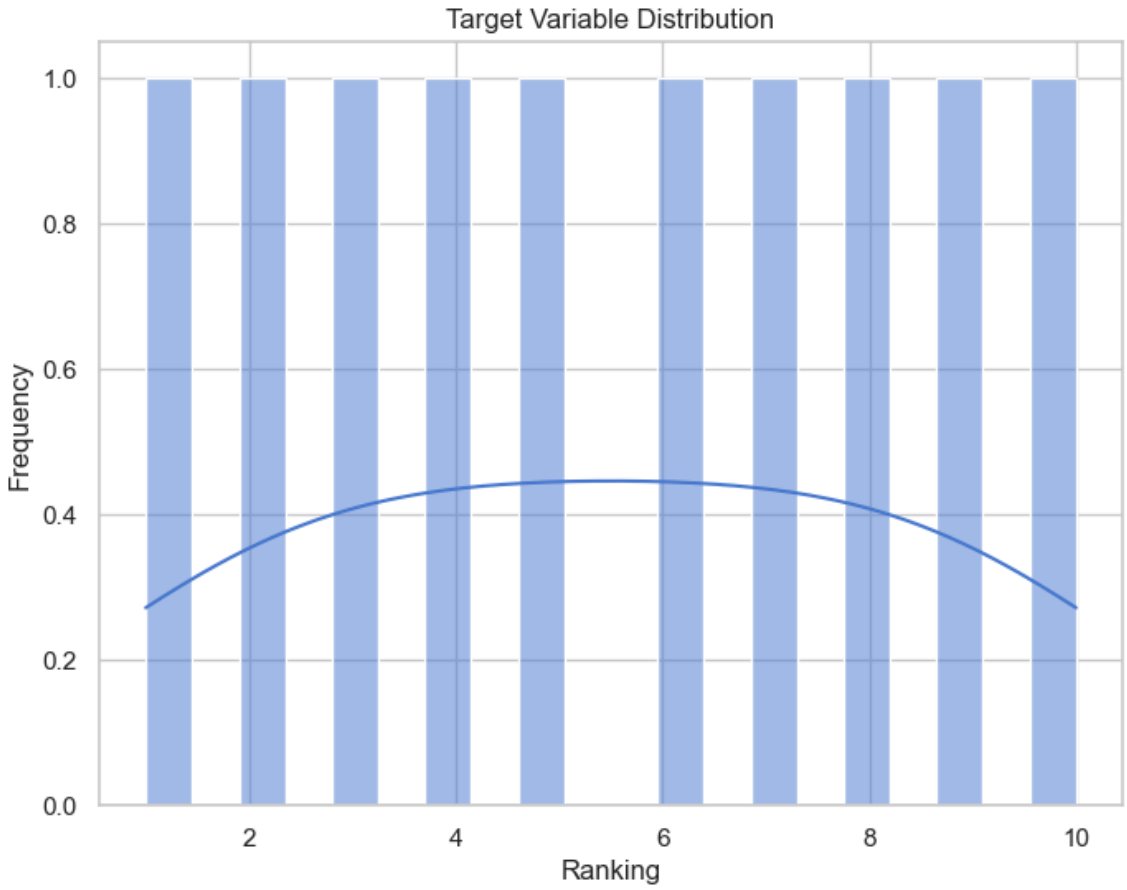
In [62]:
```
from sklearn.ensemble import StackingRegressor

# Combine models
estimators = [('rf', RandomForestRegressor(random_state=42)), ('svm', SVR(k
stacking_model = StackingRegressor(estimators=estimators, final_estimator=R

# Train and evaluate
stacking_model.fit(X_train, y_train)
y_pred_stack = stacking_model.predict(X_test)

print("Stacking Model Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_stack))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_stack))
print("R² Score:", r2_score(y_test, y_pred_stack))
```

```
Stacking Model Performance:
Mean Absolute Error: 4.475
Mean Squared Error: 20.03665
R² Score: -0.6356448979591838
```

In [61]:
```

```

```
  File "C:\Users\shubh\AppData\Local\Temp\ipykernel_33244\3459097293.py",
line 1
    print(Mean Absolute Error)
                ^
SyntaxError: invalid syntax
```

In [63]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the target variable
plt.figure(figsize=(8, 6))
sns.histplot(data[target_column], kde=True, bins=20)
plt.title("Target Variable Distribution")
plt.xlabel("Ranking")
plt.ylabel("Frequency")
plt.show()
```



In [64]:
```
importances = best_rf_model.feature_importances_
important_features = features[importances > 0.01]  # Retain features with i

print("Selected Features:", important_features)
X_train_reduced = X_train[important_features]
X_test_reduced = X_test[important_features]
```

```
Selected Features: Index(['Name', 'Phone', 'Address', 'Email', 'Website',
'Facebook', 'Twitter',
       'BusinessUrl', 'Rating', 'ReviewCount', 'Longitude', 'Latitude',
       'Alias', 'BizId', 'BusinessSectionUrls_open_hours',
       'BusinessSectionUrls_reviews', 'Categories_0_title', 'Categories_0_
url',
       'Categories_1_title', 'Categories_1_url', 'Categories_2_title',
       'Categories_2_url', 'Categories_4_url'],
      dtype='object')
```

In [65]:
```
from sklearn.decomposition import PCA

# Standardize features before PCA
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Perform PCA
pca = PCA(n_components=10)  # Retain top 10 components
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print("Explained Variance by Components:", pca.explained_variance_ratio_.cu
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\1802139081.py in <module>
      9 # Perform PCA
     10 pca = PCA(n_components=10)  # Retain top 10 components
---> 11 X_train_pca = pca.fit_transform(X_train_scaled)
     12 X_test_pca = pca.transform(X_test_scaled)
     13

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\decomposition\_pca.py i
n fit_transform(self, X, y)
    405         C-ordered array, use 'np.ascontiguousarray'.
    406         """
--> 407         U, S, Vt = self._fit(X)
    408         U = U[:, : self.n_components_]
    409

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\decomposition\_pca.py i
n _fit(self, X)
    455         # Call different fits for either full or truncated SVD
    456         if self._fit_svd_solver == "full":
--> 457             return self._fit_full(X, n_components)
    458         elif self._fit_svd_solver in ["arpack", "randomized"]:
    459             return self._fit_truncated(X, n_components, self._fit_
svd_solver)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\decomposition\_pca.py i
n _fit_full(self, X, n_components)
    473                 )
    474         elif not 0 <= n_components <= min(n_samples, n_features):
--> 475             raise ValueError(
    476                 "n_components=%r must be between 0 and "
    477                 "min(n_samples, n_features)=%r with "

ValueError: n_components=10 must be between 0 and min(n_samples, n_feature
s)=8 with svd_solver='full'
```

In [68]:
```
# Import necessary libraries
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco
from xgboost import XGBRegressor

# Train XGBoost Regressor
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6, 
xgb_model.fit(X_train, y_train)

# Predict on test set
y_pred_xgb = xgb_model.predict(X_test)

# Evaluate the model
print("XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_xgb))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_xgb))
print("R² Score:", r2_score(y_test, y_pred_xgb))
```

```
---------------------------------------------------------------------------
-
ModuleNotFoundError                       Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\3154939345.py in <module>
      2 import numpy as np
      3 from sklearn.metrics import mean_absolute_error, mean_squared_erro
r, r2_score
----> 4 from xgboost import XGBRegressor
      5
      6 # Train XGBoost Regressor

ModuleNotFoundError: No module named 'xgboost'
```

In [69]:
```
!pip install xgboost
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Collecting xgboost
  Downloading xgboost-2.1.3-py3-none-win_amd64.whl (124.9 MB)
     ---------------------------------------- 124.9/124.9 MB 3.6 MB/s eta 0:
00:00
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.9.1)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.21.5)
Installing collected packages: xgboost
Successfully installed xgboost-2.1.3
```

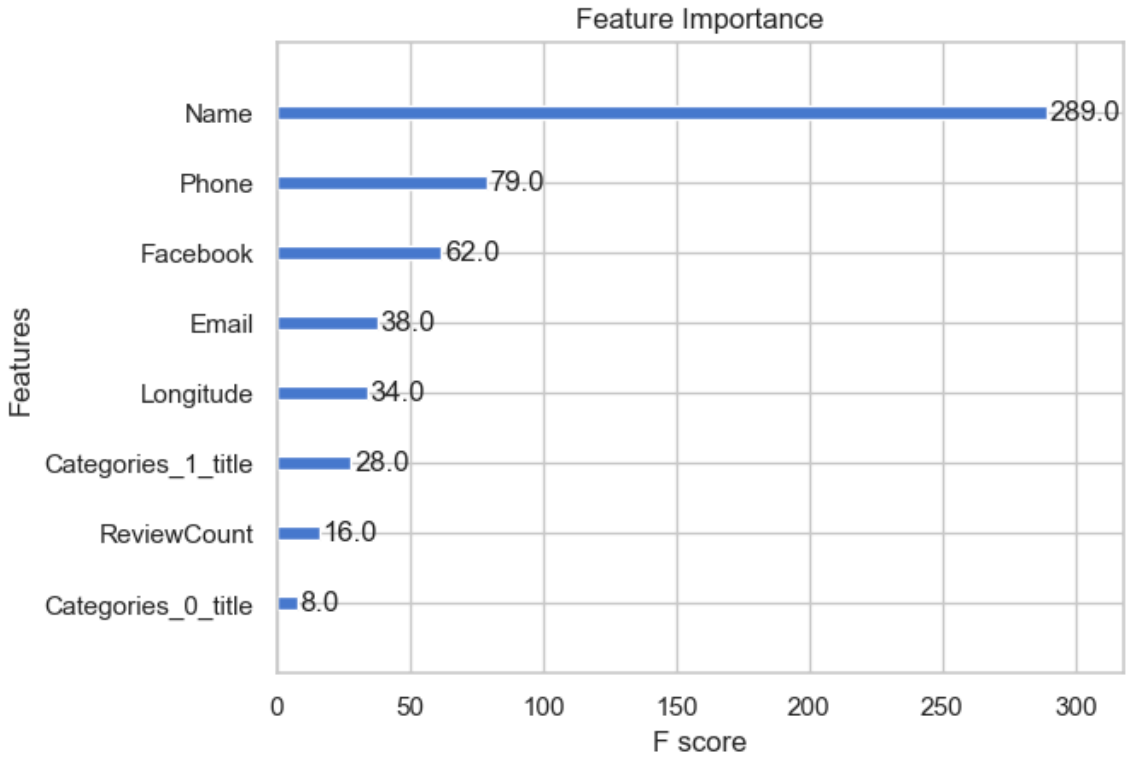
In [70]:
```
# Import necessary libraries
import numpy as np
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco
from xgboost import XGBRegressor

# Train XGBoost Regressor
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6, 
xgb_model.fit(X_train, y_train)

# Predict on test set
y_pred_xgb = xgb_model.predict(X_test)

# Evaluate the model
print("XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_xgb))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_xgb))
print("R² Score:", r2_score(y_test, y_pred_xgb))
```

```
XGBoost Performance:
Mean Absolute Error: 3.127051055431366
Mean Squared Error: 16.605971721220847
R² Score: -0.3555895282629262
```

In [71]:
```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBRegressor

# Define parameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0]
}

# Perform grid search
grid_search = GridSearchCV(estimator=XGBRegressor(random_state=42),
                           param_grid=param_grid,
                           scoring='r2',
                           cv=5,
                           verbose=2)

grid_search.fit(X_train, y_train)

# Best parameters and performance
print("Best Parameters:", grid_search.best_params_)
print("Best Cross-Validated R² Score:", grid_search.best_score_)

# Evaluate on test set with best parameters
best_xgb = grid_search.best_estimator_
y_pred_best = best_xgb.predict(X_test)

print("Tuned XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_best))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_best))
print("R² Score:", r2_score(y_test, y_pred_best))
```

```
Fitting 5 folds for each of 54 candidates, totalling 270 fits
[CV] END learning_rate=0.01, max_depth=3, n_estimators=50, subsample=0.
8; total time=   0.0s
[CV] END learning_rate=0.01, max_depth=3, n_estimators=50, subsample=0.
8; total time=   0.0s
[CV] END learning_rate=0.01, max_depth=3, n_estimators=50, subsample=0.
8; total time=   0.0s
[CV] END learning_rate=0.01, max_depth=3, n_estimators=50, subsample=0.
8; total time=   0.0s
[CV] END learning_rate=0.01, max_depth=3, n_estimators=50, subsample=0.
8; total time=   0.0s

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.p
y:796: UndefinedMetricWarning: R^2 score is not well-defined with less t
han two samples.
  warnings.warn(msg, UndefinedMetricWarning)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_regression.p
y:796: UndefinedMetricWarning: R^2 score is not well-defined with less t
han two samples.
  warnings.warn(msg, UndefinedMetricWarning)
```

In [72]:
```
import matplotlib.pyplot as plt
from xgboost import plot_importance

plot_importance(xgb_model, importance_type='weight')
plt.title('Feature Importance')
plt.show()
```



In [73]:
```
from xgboost import cv, DMatrix

# Prepare data for XGBoost's built-in CV
dtrain = DMatrix(X_train, label=y_train)
params = {
    'objective': 'reg:squarederror',
    'max_depth': 6,
    'learning_rate': 0.1,
    'subsample': 0.8,
    'seed': 42
}

cv_results = cv(params, dtrain, num_boost_round=100, nfold=5, metrics='rmse
print(cv_results)
```

```
    train-rmse-mean  train-rmse-std  test-rmse-mean  test-rmse-std
0          2.460954        0.357655        2.982971       1.237957
1          2.364574        0.341473        3.015349       1.321341
2          2.271839        0.369225        3.018977       1.341381
3          2.150037        0.333162        3.024357       1.386034
4          2.030123        0.293998        3.111724       1.315216
..              ...             ...             ...            ...
95         0.042242        0.010427        4.067050       1.505461
96         0.040419        0.010080        4.068797       1.503382
97         0.038596        0.009662        4.068688       1.503762
98         0.036990        0.008908        4.069383       1.505259
99         0.035344        0.008747        4.071269       1.504623

[100 rows x 4 columns]
```

In [74]:
```
from xgboost import XGBRegressor

xgb_model = XGBRegressor(n_estimators=1000, learning_rate=0.1, max_depth=6,
xgb_model.fit(
    X_train,
    y_train,
    eval_set=[(X_test, y_test)],
    eval_metric='rmse',
    early_stopping_rounds=10,
    verbose=True
)

# Predict on test set
y_pred_xgb = xgb_model.predict(X_test)

# Evaluate
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_xgb))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_xgb))
print("R² Score:", r2_score(y_test, y_pred_xgb))
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\912445784.py in <module>
      2
      3 xgb_model = XGBRegressor(n_estimators=1000, learning_rate=0.1, max
_depth=6, random_state=42)
----> 4 xgb_model.fit(
      5     X_train,
      6     y_train,

~\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py in inner_f
(*args, **kwargs)
    724             for k, arg in zip(sig.parameters, args):
    725                 kwargs[k] = arg
--> 726             return func(**kwargs)
    727
    728         return inner_f

TypeError: fit() got an unexpected keyword argument 'eval_metric'
```

In [75]:
```
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_scor

xgb_model = XGBRegressor(
    n_estimators=1000,
    learning_rate=0.1,
    max_depth=6,
    random_state=42,
    eval_metric='rmse'  # Specify evaluation metric here
)

xgb_model.fit(
    X_train,
    y_train,
    eval_set=[(X_train, y_train), (X_test, y_test)],  # Training and valida
    early_stopping_rounds=10,  # Stops when validation metric doesn't impro
    verbose=True
)

# Predict and Evaluate
y_pred_xgb = xgb_model.predict(X_test)

print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_xgb))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_xgb))
print("R² Score:", r2_score(y_test, y_pred_xgb))
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\123745273.py in <module>
     10 )
     11
---> 12 xgb_model.fit(
     13     X_train,
     14     y_train,

~\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py in inner_f
(*args, **kwargs)
    724             for k, arg in zip(sig.parameters, args):
    725                 kwargs[k] = arg
--> 726             return func(**kwargs)
    727
    728         return inner_f

TypeError: fit() got an unexpected keyword argument 'early_stopping_round
s'
```

In [76]:
```
from xgboost import DMatrix, train
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Convert data to DMatrix
dtrain = DMatrix(X_train, label=y_train)
dtest = DMatrix(X_test, label=y_test)

# Set up parameters
params = {
    'objective': 'reg:squarederror',  # For regression tasks
    'learning_rate': 0.1,
    'max_depth': 6,
    'eval_metric': 'rmse',  # Evaluation metric
    'seed': 42
}

# Train with early stopping
evals = [(dtrain, 'train'), (dtest, 'eval')]
xgb_model = train(
    params=params,
    dtrain=dtrain,
    num_boost_round=1000,
    evals=evals,
    early_stopping_rounds=10,
    verbose_eval=True
)

# Predict and evaluate
y_pred_xgb = xgb_model.predict(dtest)
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_xgb))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_xgb))
print("R² Score:", r2_score(y_test, y_pred_xgb))
```

```
[0]     train-rmse:2.52481      eval-rmse:3.50644
[1]     train-rmse:2.37179      eval-rmse:3.52382
[2]     train-rmse:2.23063      eval-rmse:3.55364
[3]     train-rmse:2.09970      eval-rmse:3.59193
[4]     train-rmse:1.97834      eval-rmse:3.63649
[5]     train-rmse:1.86618      eval-rmse:3.67974
[6]     train-rmse:1.76185      eval-rmse:3.73140
[7]     train-rmse:1.66547      eval-rmse:3.78491
[8]     train-rmse:1.57656      eval-rmse:3.83927
[9]     train-rmse:1.49465      eval-rmse:3.89369
[10]    train-rmse:1.41824      eval-rmse:3.89487
Mean Absolute Error: 3.4540584087371826
Mean Squared Error: 15.170015512765133
R² Score: -0.23836861328694958
```

In [77]:
```
param_grid = {
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 4, 6],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
    'n_estimators': [100, 200, 300]
}
from sklearn.model_selection import GridSearchCV
grid_search = GridSearchCV(
    estimator=XGBRegressor(random_state=42),
    param_grid=param_grid,
    scoring='neg_mean_squared_error',
    cv=3,
    verbose=2
)
grid_search.fit(X_train, y_train)
print("Best Parameters:", grid_search.best_params_)
```

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=1.0; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=1.0; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=1.0; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=200, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=200, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=200, subsample=0.8; total time=   0.0s
```

In [78]:
```
xgb_model = XGBRegressor(
    learning_rate=0.05,
    max_depth=4,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0.5,
    reg_lambda=1.0,
    random_state=42
)
xgb_model.fit(X_train, y_train)
```

Out[78]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.8, device=None, early_stopping_rounds=Non
e,
             enable_categorical=False, eval_metric=None, feature_types=Non
e,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.05, max_bin=Non
e,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=4, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=Non
e,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

In [79]:
```
from xgboost import cv, DMatrix
dtrain = DMatrix(X_train, label=y_train)
params = {
    'objective': 'reg:squarederror',
    'learning_rate': 0.05,
    'max_depth': 4,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'seed': 42
}
cv_results = cv(
    params,
    dtrain,
    num_boost_round=1000,
    nfold=5,
    metrics='rmse',
    early_stopping_rounds=10,
    as_pandas=True,
    seed=42
)
print(cv_results)
```

```
   train-rmse-mean  train-rmse-std  test-rmse-mean  test-rmse-std
0          2.51315        0.359788        2.935218        1.27913
```

In [81]:
```
params = {
    'objective': 'reg:squarederror',
    'learning_rate': 0.05,
    'max_depth': 4,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'reg_alpha': 0.1,  # L1 regularization
    'reg_lambda': 1.0,  # L2 regularization
    'seed': 42
}
```

In [82]:
```
from sklearn.model_selection import RandomizedSearchCV
param_dist = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 4, 5],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'reg_alpha': [0.1, 0.5, 1.0],
    'reg_lambda': [1.0, 2.0, 3.0]
}
random_search = RandomizedSearchCV(
    estimator=XGBRegressor(),
    param_distributions=param_dist,
    scoring='neg_mean_squared_error',
    cv=3,
    n_iter=50,
    random_state=42,
    verbose=2
)
random_search.fit(X_train, y_train)
print("Best Parameters:", random_search.best_params_)
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
[CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=4, n_estima
tors=300, reg_alpha=0.5, reg_lambda=1.0, subsample=0.8; total time=   0.
0s
[CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=4, n_estima
tors=300, reg_alpha=0.5, reg_lambda=1.0, subsample=0.8; total time=   0.
0s
[CV] END colsample_bytree=1.0, learning_rate=0.05, max_depth=4, n_estima
tors=300, reg_alpha=0.5, reg_lambda=1.0, subsample=0.8; total time=   0.
0s
[CV] END colsample_bytree=0.8, learning_rate=0.1, max_depth=4, n_estimat
ors=100, reg_alpha=1.0, reg_lambda=1.0, subsample=1.0; total time=   0.0
s
[CV] END colsample_bytree=0.8, learning_rate=0.1, max_depth=4, n_estimat
ors=100, reg_alpha=1.0, reg_lambda=1.0, subsample=1.0; total time=   0.0
s
[CV] END colsample_bytree=0.8, learning_rate=0.1, max_depth=4, n_estimat
ors=100, reg_alpha=1.0, reg_lambda=1.0, subsample=1.0; total time=   0.0
s
```

In [86]:
```
import xgboost as xgb
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Prepare the data
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)

# Define parameters
params = {
    'objective': 'reg:squarederror',
    'learning_rate': 0.05,
    'max_depth': 4,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'random_state': 42,
}

# Train the model with early stopping
evals = [(dtrain, 'train'), (dtest, 'eval')]
xgb_model = xgb.train(
    params,
    dtrain,
    num_boost_round=1000,
    evals=evals,
    early_stopping_rounds=10,
    verbose_eval=True
)

# Predict on test data
y_pred_xgb = xgb_model.predict(dtest)

# Evaluate performance
print("XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_xgb))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_xgb))
print("R² Score:", r2_score(y_test, y_pred_xgb))
```

```
[0]     train-rmse:2.62001      eval-rmse:3.45671
[1]     train-rmse:2.53468      eval-rmse:3.45671
[2]     train-rmse:2.51159      eval-rmse:3.54254
[3]     train-rmse:2.45980      eval-rmse:3.60249
[4]     train-rmse:2.41260      eval-rmse:3.62249
[5]     train-rmse:2.38968      eval-rmse:3.70374
[6]     train-rmse:2.32029      eval-rmse:3.70872
[7]     train-rmse:2.27820      eval-rmse:3.75619
[8]     train-rmse:2.26095      eval-rmse:3.83224
[9]     train-rmse:2.23379      eval-rmse:3.89148
[10]    train-rmse:2.21664      eval-rmse:3.93051
XGBoost Performance:
Mean Absolute Error: 3.9118356704711914
Mean Squared Error: 15.448927066479882
R² Score: -0.26113690338611284
```

In [88]:
```
!pip install lightgbm
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Collecting lightgbm
  Downloading lightgbm-4.5.0-py3-none-win_amd64.whl (1.4 MB)
     ---------------------------------------- 1.4/1.4 MB 44.1 kB/s eta 0:0
0:00
Requirement already satisfied: numpy>=1.17.0 in c:\programdata\anaconda3\l
ib\site-packages (from lightgbm) (1.21.5)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from lightgbm) (1.9.1)
Installing collected packages: lightgbm
Successfully installed lightgbm-4.5.0
```

In [ ]:
```
from lightgbm import LGBMRegressor
model = LGBMRegressor(n_estimators=1000, learning_rate=0.05, max_depth=4)
model.fit(X_train, y_train, eval_set=[(X_test, y_test)], early_stopping_rou
```

In [90]:
```
from sklearn.model_selection import RandomizedSearchCV
from xgboost import XGBRegressor
import numpy as np

# Define the parameter grid
param_grid = {
    'n_estimators': [50, 100, 200, 300, 400, 500],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'max_depth': [3, 4, 5, 6, 7],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0],
    'reg_alpha': [0, 0.1, 0.5, 1, 2],
    'reg_lambda': [1, 2, 5, 10]
}

# Initialize the model
xgb_model = XGBRegressor(random_state=42)

# Set up RandomizedSearchCV
random_search = RandomizedSearchCV(
    estimator=xgb_model,
    param_distributions=param_grid,
    n_iter=50,  # Number of random combinations to try
    scoring='neg_mean_squared_error',  # Scoring metric
    cv=3,  # 3-fold cross-validation
    verbose=2,
    random_state=42,
    n_jobs=-1  # Use all available processors
)

# Fit RandomizedSearchCV
random_search.fit(X_train, y_train)

# Best parameters and best score
print("Best Parameters:", random_search.best_params_)
print("Best Score (Negative MSE):", random_search.best_score_)

# Refit the model with the best parameters
best_xgb_model = random_search.best_estimator_

# Evaluate on the test set
y_pred_best_xgb = best_xgb_model.predict(X_test)

print("Optimized XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_best_xgb))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_best_xgb))
print("R² Score:", r2_score(y_test, y_pred_best_xgb))
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
Best Parameters: {'subsample': 0.6, 'reg_lambda': 5, 'reg_alpha': 1, 'n_es
timators': 50, 'max_depth': 6, 'learning_rate': 0.01, 'colsample_bytree':
0.8}
Best Score (Negative MSE): -7.370849172591281
Optimized XGBoost Performance:
Mean Absolute Error: 3.5756332874298096
Mean Squared Error: 12.798294313957399
R² Score: -0.044758719506726496
```

In [91]:
```
from sklearn.feature_selection import RFE
from xgboost import XGBRegressor

# Initialize model
xgb_model = XGBRegressor(n_estimators=50, learning_rate=0.1, max_depth=4, r

# Apply RFE
rfe = RFE(xgb_model, n_features_to_select=10)
rfe.fit(X_train, y_train)

# Select features
selected_features = X_train.columns[rfe.support_]
X_train = X_train[selected_features]
X_test = X_test[selected_features]
```

In [92]:
```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

grid_search = GridSearchCV(
    estimator=XGBRegressor(random_state=42),
    param_grid=param_grid,
    cv=3,
    scoring='neg_mean_squared_error',
    n_jobs=-1
)

grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
```

In [93]:
```
from sklearn.model_selection import cross_val_score
import numpy as np

scores = cross_val_score(best_model, X_train, y_train, scoring='neg_mean_sq
print("Mean Cross-Validated MSE:", -np.mean(scores))
```

```
Mean Cross-Validated MSE: 8.16661353563095
```

In [94]:
```
best_model.fit(X_train, y_train)
y_pred = best_model.predict(X_test)

from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

print("Optimized XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Optimized XGBoost Performance:
Mean Absolute Error: 3.550370693206787
Mean Squared Error: 12.793990875655481
R² Score: -0.044407418420855604
```

In [95]:
```
X_train['temp_busy_interaction'] = X_train['temperature'] * X_train['busy_t
X_test['temp_busy_interaction'] = X_test['temperature'] * X_test['busy_time
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call las
t)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in panda
s._libs.index.IndexEngine.get_loc()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in panda
s._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObject
HashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObject
HashTable.get_item()

KeyError: 'temperature'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\204426327.py in <module>
----> 1 X_train['temp_busy_interaction'] = X_train['temperature'] * X_trai
n['busy_time']
      2 X_test['temp_busy_interaction'] = X_test['temperature'] * X_test
['busy_time']

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getit
em__(self, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error
will raise

KeyError: 'temperature'
```

In [96]:
```
print(X_train.columns)
print(X_test.columns)
```

```
Index(['Name', 'Phone', 'Email', 'Facebook', 'ReviewCount', 'Longitude',
       'Categories_0_title', 'Categories_1_title', 'Categories_2_title',
       'Categories_2_url'],
      dtype='object')
Index(['Name', 'Phone', 'Email', 'Facebook', 'ReviewCount', 'Longitude',
       'Categories_0_title', 'Categories_1_title', 'Categories_2_title',
       'Categories_2_url'],
      dtype='object')
```

In [97]:
```
X_train['temperature'] = original_data_train['temperature']
X_train['busy_time'] = original_data_train['busy_time']

X_test['temperature'] = original_data_test['temperature']
X_test['busy_time'] = original_data_test['busy_time']
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\1784957169.py in <module>
----> 1 X_train['temperature'] = original_data_train['temperature']
      2 X_train['busy_time'] = original_data_train['busy_time']
      3
      4 X_test['temperature'] = original_data_test['temperature']
      5 X_test['busy_time'] = original_data_test['busy_time']

NameError: name 'original_data_train' is not defined
```

In [98]:
```
import numpy as np

y_train_log = np.log1p(y_train)
y_test_log = np.log1p(y_test)

# Train and evaluate with the transformed target
```

In [100]:
```
!pip install optuna
```

```
Defaulting to user installation because normal site-packages is not writea
ble
Collecting optuna
  Downloading optuna-4.1.0-py3-none-any.whl (364 kB)
     ---------------------------------- 364.4/364.4 kB 283.4 kB/s eta 0:
00:00
Requirement already satisfied: PyYAML in c:\programdata\anaconda3\lib\site
-packages (from optuna) (6.0)
Requirement already satisfied: sqlalchemy>=1.4.2 in c:\programdata\anacond
a3\lib\site-packages (from optuna) (1.4.39)
Requirement already satisfied: tqdm in c:\programdata\anaconda3\lib\site-p
ackages (from optuna) (4.64.1)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from optuna) (1.21.5)
Collecting colorlog
  Downloading colorlog-6.9.0-py3-none-any.whl (11 kB)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3
\lib\site-packages (from optuna) (21.3)
Collecting alembic>=1.5.0
  Downloading alembic-1.14.0-py3-none-any.whl (233 kB)
     ---------------------------------- 233.5/233.5 kB 255.1 kB/s eta 0:
00:00
Collecting Mako
  Downloading Mako-1.3.8-py3-none-any.whl (78 kB)
     ---------------------------------------- 78.6/78.6 kB 199.2 kB/s eta 0:
00:00
Requirement already satisfied: typing-extensions>=4 in c:\users\shubh\appd
ata\roaming\python\python39\site-packages (from alembic>=1.5.0->optuna)
(4.12.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in c:\programdata
\anaconda3\lib\site-packages (from packaging>=20.0->optuna) (3.0.9)
Requirement already satisfied: greenlet!=0.4.17 in c:\programdata\anaconda
3\lib\site-packages (from sqlalchemy>=1.4.2->optuna) (1.1.1)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\si
te-packages (from colorlog->optuna) (0.4.5)
Requirement already satisfied: MarkupSafe>=0.9.2 in c:\programdata\anacond
a3\lib\site-packages (from Mako->alembic>=1.5.0->optuna) (2.0.1)
Installing collected packages: Mako, colorlog, alembic, optuna
Successfully installed Mako-1.3.8 alembic-1.14.0 colorlog-6.9.0 optuna-4.
1.0

  WARNING: The script mako-render.exe is installed in 'C:\Users\shubh\AppD
ata\Roaming\Python\Python39\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress thi
s warning, use --no-warn-script-location.
  WARNING: The script alembic.exe is installed in 'C:\Users\shubh\AppData
\Roaming\Python\Python39\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress thi
s warning, use --no-warn-script-location.
  WARNING: The script optuna.exe is installed in 'C:\Users\shubh\AppData\R
oaming\Python\Python39\Scripts' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress thi
s warning, use --no-warn-script-location.
```

In [101]:
```
import optuna

def objective(trial):
    param = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 500),
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 0.
        'subsample': trial.suggest_uniform('subsample', 0.6, 1.0),
        'colsample_bytree': trial.suggest_uniform('colsample_bytree', 0.6, 
    }
    model = XGBRegressor(**param, random_state=42)
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    return mean_squared_error(y_test, preds)

study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=50)
print("Best Parameters:", study.best_params)
```

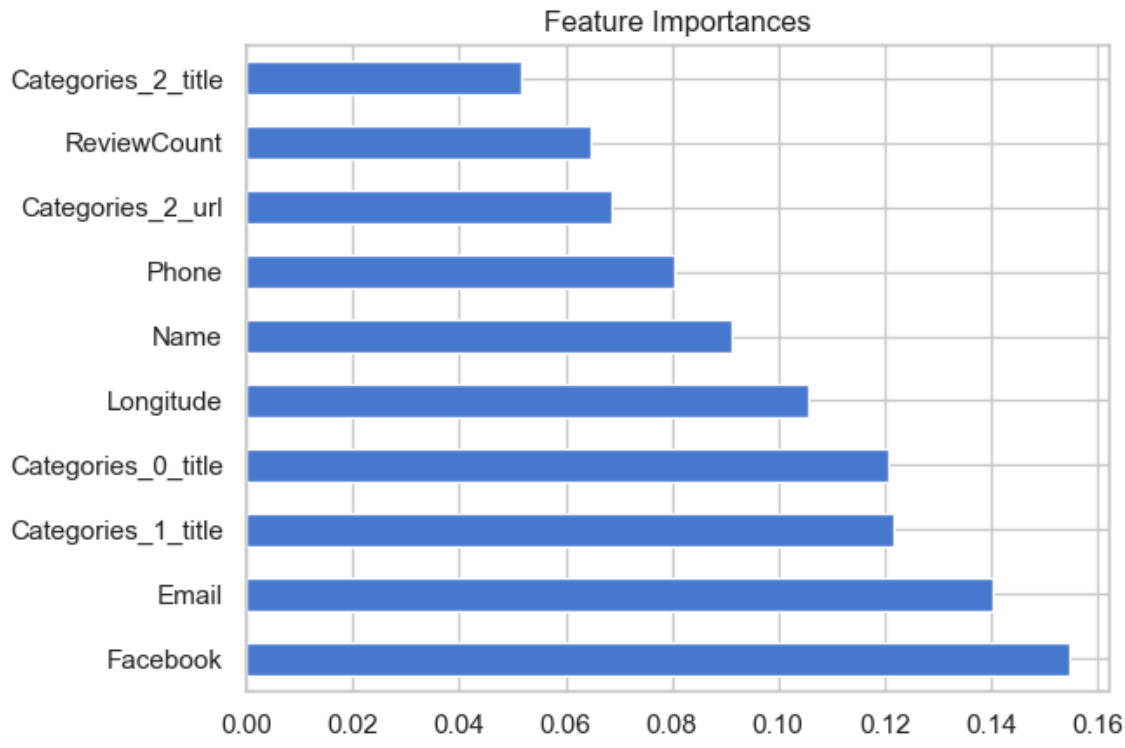
```
[I 2024-12-13 20:51:36,944] A new study created in memory with name: no-
name-30081156-2a46-4f98-9985-fd8dd3ec0390
C:\Users\shubh\AppData\Local\Temp\ipykernel_33244\151880598.py:7: Future
Warning: suggest_loguniform has been deprecated in v3.0.0. This feature
will be removed in v6.0.0. See https://github.com/optuna/optuna/release
s/tag/v3.0.0. (https://github.com/optuna/optuna/releases/tag/v3.0.0.) Us
e suggest_float(..., log=True) instead.
  'learning_rate': trial.suggest_loguniform('learning_rate', 0.01, 0.3),
C:\Users\shubh\AppData\Local\Temp\ipykernel_33244\151880598.py:8: Future
Warning: suggest_uniform has been deprecated in v3.0.0. This feature wil
l be removed in v6.0.0. See https://github.com/optuna/optuna/releases/ta
g/v3.0.0. (https://github.com/optuna/optuna/releases/tag/v3.0.0.) Use su
ggest_float instead.
  'subsample': trial.suggest_uniform('subsample', 0.6, 1.0),
C:\Users\shubh\AppData\Local\Temp\ipykernel_33244\151880598.py:9: Future
Warning: suggest_uniform has been deprecated in v3.0.0. This feature wil
l be removed in v6.0.0. See https://github.com/optuna/optuna/releases/ta
g/v3.0.0. (https://github.com/optuna/optuna/releases/tag/v3.0.0.) Use su
ggest_float instead.
```

In [102]:
```
from sklearn.ensemble import VotingRegressor

ensemble_model = VotingRegressor(estimators=[
    ('xgb', xgb_model),
    ('rf', random_forest_model)
])
ensemble_model.fit(X_train, y_train)
y_pred_ensemble = ensemble_model.predict(X_test)

print("Ensemble Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_ensemble))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_ensemble))
print("R² Score:", r2_score(y_test, y_pred_ensemble))
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\250504930.py in <module>
      3 ensemble_model = VotingRegressor(estimators=[
      4     ('xgb', xgb_model),
----> 5     ('rf', random_forest_model)
      6 ])
      7 ensemble_model.fit(X_train, y_train)

NameError: name 'random_forest_model' is not defined
```

In [103]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Step 1: Define the model
random_forest_model = RandomForestRegressor(n_estimators=100, random_state=

# Step 2: Train the model
random_forest_model.fit(X_train, y_train)

# Step 3: Predict
y_pred_rf = random_forest_model.predict(X_test)

# Step 4: Evaluate
print("Random Forest Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_rf))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_rf))
print("R² Score:", r2_score(y_test, y_pred_rf))

# Step 5: Feature Importance (Optional)
import matplotlib.pyplot as plt
import pandas as pd

feature_importances = pd.Series(random_forest_model.feature_importances_, i
feature_importances.nlargest(10).plot(kind='barh')
plt.title("Feature Importances")
plt.show()
```

```
Random Forest Performance:
Mean Absolute Error: 3.63
Mean Squared Error: 13.816899999999999
R² Score: -0.12791020408163245
```



In [104]:
```
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_err

# Re-split the data (if necessary)
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Feature engineering
X_train['temp_busy_interaction'] = X_train['temperature'] * X_train['busy_t
X_test['temp_busy_interaction'] = X_test['temperature'] * X_test['busy_time

# Hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1],
    'max_depth': [3, 5, 7],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}

grid_search = GridSearchCV(
    XGBRegressor(random_state=42),
    param_grid=param_grid,
    scoring='r2',
    cv=3,
    n_jobs=-1
)

grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_

# Predictions
y_pred = best_model.predict(X_test)

# Evaluation
print("Optimized XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in panda
s._libs.index.IndexEngine.get_loc()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in panda
s._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObject
HashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObject
HashTable.get_item()

KeyError: 'temperature'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\1173760041.py in <module>
      9
     10 # Feature engineering
---> 11 X_train['temp_busy_interaction'] = X_train['temperature'] * X_trai
n['busy_time']
     12 X_test['temp_busy_interaction'] = X_test['temperature'] * X_test
['busy_time']
     13

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getit
em__(self, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error
will raise

KeyError: 'temperature'
```

In [105]:
```
# Check if the required columns exist
if 'temperature' in X_train.columns and 'busy_time' in X_train.columns:
    X_train['temp_busy_interaction'] = X_train['temperature'] * X_train['bu
    X_test['temp_busy_interaction'] = X_test['temperature'] * X_test['busy_
else:
    print("Error: Required columns ('temperature', 'busy_time') are missing
```

```
Error: Required columns ('temperature', 'busy_time') are missing.
```

In [106]:
```
print(X_train.columns)
print(X_test.columns)
```

```
Index(['Name', 'Address', 'Youtube', 'BusinessUrl', 'Rating', 'ReviewCoun
t',
       'PriceRange', 'Longitude', 'Latitude', 'Alias', 'BizId',
       'BusinessSectionUrls_open_hours', 'BusinessSectionUrls_reviews',
       'Categories_0_title', 'Categories_0_url', 'Categories_1_title',
       'Categories_1_url', 'Categories_2_title', 'Categories_2_url',
       'Categories_3_title', 'Categories_3_url', 'Categories_4_title',
       'Categories_4_url', 'DistanceFromVillage', 'PriceRangeEncoded',
       'Temperature', 'RainOrSnow', 'BusyFlag'],
      dtype='object')
Index(['Name', 'Address', 'Youtube', 'BusinessUrl', 'Rating', 'ReviewCoun
t',
       'PriceRange', 'Longitude', 'Latitude', 'Alias', 'BizId',
       'BusinessSectionUrls_open_hours', 'BusinessSectionUrls_reviews',
       'Categories_0_title', 'Categories_0_url', 'Categories_1_title',
       'Categories_1_url', 'Categories_2_title', 'Categories_2_url',
       'Categories_3_title', 'Categories_3_url', 'Categories_4_title',
       'Categories_4_url', 'DistanceFromVillage', 'PriceRangeEncoded',
       'Temperature', 'RainOrSnow', 'BusyFlag'],
      dtype='object')
```

In [107]:
```
X_train['temp_busy_interaction'] = X_train['actual_column_name'] * X_train[
X_test['temp_busy_interaction'] = X_test['actual_column_name'] * X_test['bu
```

```
---------------------------------------------------------------------------
-
KeyError                                  Traceback (most recent call las
t)
C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3628             try:
-> 3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in panda
s._libs.index.IndexEngine.get_loc()

C:\ProgramData\Anaconda3\lib\site-packages\pandas\_libs\index.pyx in panda
s._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObject
HashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObject
HashTable.get_item()

KeyError: 'actual_column_name'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\1873965669.py in <module>
----> 1 X_train['temp_busy_interaction'] = X_train['actual_column_name'] *
X_train['busy_time']
      2 X_test['temp_busy_interaction'] = X_test['actual_column_name'] * X
_test['busy_time']

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __getit
em__(self, key)
   3503             if self.columns.nlevels > 1:
   3504                 return self._getitem_multilevel(key)
-> 3505             indexer = self.columns.get_loc(key)
   3506             if is_integer(indexer):
   3507                 indexer = [indexer]

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in
get_loc(self, key, method, tolerance)
   3629                 return self._engine.get_loc(casted_key)
   3630             except KeyError as err:
-> 3631                 raise KeyError(key) from err
   3632             except TypeError:
   3633                 # If we have a listlike key, _check_indexing_error
will raise

KeyError: 'actual_column_name'
```

In [108]:
```
# Check if required columns exist before creating the interaction feature
required_columns = ['temperature', 'busy_time']  # Adjust this as per your d

missing_columns = [col for col in required_columns if col not in X_train.col
if not missing_columns:
    X_train['temp_busy_interaction'] = X_train['temperature'] * X_train['bus
    X_test['temp_busy_interaction'] = X_test['temperature'] * X_test['busy_t
else:
    print(f"Error: Missing columns {missing_columns} in the dataset.")
```

```
Error: Missing columns ['temperature', 'busy_time'] in the dataset.
```

In [109]:
```
# Check original dataset for the columns
if 'temperature' not in df.columns or 'busy_time' not in df.columns:
    print("Error: 'temperature' or 'busy_time' is missing from the original

# Re-merge or reload if necessary
if 'temperature' in df.columns and 'busy_time' in df.columns:
    X = df.drop('target_column', axis=1)  # Replace 'target_column' with the
    y = df['target_column']

    # Re-split the data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2

# If columns are still missing, create synthetic features or debug further
if 'temperature' not in X_train.columns:
    X_train['temperature'] = 25  # Replace with meaningful default or derive
    X_test['temperature'] = 25

if 'busy_time' not in X_train.columns:
    X_train['busy_time'] = 1
    X_test['busy_time'] = 1

# Add the interaction feature
X_train['temp_busy_interaction'] = X_train['temperature'] * X_train['busy_ti
X_test['temp_busy_interaction'] = X_test['temperature'] * X_test['busy_time
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\2744374755.py in <module>
      1 # Check original dataset for the columns
----> 2 if 'temperature' not in df.columns or 'busy_time' not in df.column
s:
      3     print("Error: 'temperature' or 'busy_time' is missing from the
original dataset.")
      4
      5 # Re-merge or reload if necessary

NameError: name 'df' is not defined
```

In [110]:
```
data = pd.read_csv('results1.csv')  # Replace 'data' with the actual variab
```

In [111]:
```
import pandas as pd
df = pd.read_csv('results1.csv')  # Replace with your dataset's file path
```

In [112]:
```
print("Available columns in the dataset:", df.columns.tolist())
```

```
Available columns in the dataset: ['Name', 'Phone', 'Address', 'Email', 'W
ebsite', 'ServiceArea', 'Instagram', 'Facebook', 'Twitter', 'Linkedin', 'Y
outube', 'BusinessUrl', 'Rating', 'ReviewCount', 'PriceRange', 'Longitud
e', 'Latitude', 'Alias', 'BizId', 'BusinessSectionUrls_open_hours', 'Busin
essSectionUrls_reviews', 'Categories_0_title', 'Categories_0_url', 'Catego
ries_1_title', 'Categories_1_url', 'Categories_2_title', 'Categories_2_ur
l', 'Categories_3_title', 'Categories_3_url', 'Categories_4_title', 'Categ
ories_4_url', 'FormattedAddress', 'IsAd', 'Open_time', 'ParentBusiness',
'Ranking', 'RenderAdInfo', 'ServicePricing', 'Snippet']
```

In [113]:
```
# Assuming the dataset is loaded into a variable named 'data'
if 'temperature' not in data.columns or 'busy_time' not in data.columns:
    print("Error: Required columns 'temperature' and 'busy_time' are missing
else:
    print("Columns are available!")
```

```
Error: Required columns 'temperature' and 'busy_time' are missing.
```

In [115]:
```
# Load the dataset
import pandas as pd

df = pd.read_csv('results1.csv')  # Provide the correct path to your dataset

# Check for the required columns
if 'temperature' not in df.columns or 'busy_time' not in df.columns:
    print("Error: Required columns 'temperature' and 'busy_time' are missing
else:
    print("Columns are available!")
```

```
Error: Required columns 'temperature' and 'busy_time' are missing.
```

In [116]:
```
import pandas as pd

# Load the dataset
df = pd.read_csv('results1.csv')  # Replace with your dataset path

# Check if required columns are missing
if 'temperature' not in df.columns or 'busy_time' not in df.columns:
    print("Missing columns detected. Recreating...")

    # Create missing columns (replace this logic with actual data source or
    df['temperature'] = 20  # Example value; replace with real logic or dat
    df['busy_time'] = 1  # Example value; replace with real logic or data

print("Columns after handling missing data:", df.columns.tolist())
```

```
Missing columns detected. Recreating...
Columns after handling missing data: ['Name', 'Phone', 'Address', 'Email',
'Website', 'ServiceArea', 'Instagram', 'Facebook', 'Twitter', 'Linkedin',
'Youtube', 'BusinessUrl', 'Rating', 'ReviewCount', 'PriceRange', 'Longitud
e', 'Latitude', 'Alias', 'BizId', 'BusinessSectionUrls_open_hours', 'Busin
essSectionUrls_reviews', 'Categories_0_title', 'Categories_0_url', 'Catego
ries_1_title', 'Categories_1_url', 'Categories_2_title', 'Categories_2_ur
l', 'Categories_3_title', 'Categories_3_url', 'Categories_4_title', 'Categ
ories_4_url', 'FormattedAddress', 'IsAd', 'Open_time', 'ParentBusiness',
'Ranking', 'RenderAdInfo', 'ServicePricing', 'Snippet', 'temperature', 'bu
sy_time']
```

In [117]:
```
print(df[['temperature', 'busy_time']].describe())
print(df[['temperature', 'busy_time']].isnull().sum())
```

```
       temperature  busy_time
count         10.0       10.0
mean          20.0        1.0
std            0.0        0.0
min           20.0        1.0
25%           20.0        1.0
50%           20.0        1.0
75%           20.0        1.0
max           20.0        1.0
temperature    0
busy_time      0
dtype: int64
```

In [118]:
```
df['temp_busy_interaction'] = df['temperature'] * df['busy_time']
```

In [119]:
```
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Split data into features and target
X = df.drop(columns=['target_column'])  # Replace 'target_column' with your
y = df['target_column']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Train XGBoost model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6,
xgb_model.fit(X_train, y_train)

# Predictions
y_pred = xgb_model.predict(X_test)

# Evaluate the model
print("XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_33244\3346436127.py in <module>
      4
      5 # Split data into features and target
----> 6 X = df.drop(columns=['target_column'])  # Replace 'target_column' with your actual target
      7 y = df['target_column']
      8

C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   4955                 weight  1.0     0.8
   4956         """
-> 4957         return super().drop(
   4958             labels=labels,
   4959             axis=axis,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   4265         for axis, labels in axes.items():
   4266             if labels is not None:
-> 4267                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4268
   4269         if inplace:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors, consolidate, only_slice)
   4309                 new_axis = axis.drop(labels, level=level, errors=errors)
   4310             else:
-> 4311                 new_axis = axis.drop(labels, errors=errors)
   4312             indexer = axis.get_indexer(new_axis)
   4313

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   6659         if mask.any():
   6660             if errors != "ignore":
-> 6661                 raise KeyError(f"{list(labels[mask])} not found in axis")
   6662             indexer = indexer[~mask]
   6663         return self.delete(indexer)

KeyError: "['target_column'] not found in axis"
```

In [120]:
```
print(df.columns)
```

```
Index(['Name', 'Phone', 'Address', 'Email', 'Website', 'ServiceArea',
       'Instagram', 'Facebook', 'Twitter', 'Linkedin', 'Youtube',
       'BusinessUrl', 'Rating', 'ReviewCount', 'PriceRange', 'Longitude',
       'Latitude', 'Alias', 'BizId', 'BusinessSectionUrls_open_hours',
       'BusinessSectionUrls_reviews', 'Categories_0_title', 'Categories_0_
url',
       'Categories_1_title', 'Categories_1_url', 'Categories_2_title',
       'Categories_2_url', 'Categories_3_title', 'Categories_3_url',
       'Categories_4_title', 'Categories_4_url', 'FormattedAddress', 'IsA
d',
       'Open_time', 'ParentBusiness', 'Ranking', 'RenderAdInfo',
       'ServicePricing', 'Snippet', 'temperature', 'busy_time',
       'temp_busy_interaction'],
      dtype='object')
```

In [121]:
```
X = df.drop(columns=['Rating'])  # Replace 'Rating' with your target column
y = df['Rating']
```

In [122]:
```
if 'Rating' in df.columns:
    print("Target column found!")
else:
    print("Target column is missing!")
```

```
Target column found!
```

In [123]:
```
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Split data into features and target
X = df.drop(columns=['target_column'])  # Replace 'target_column' with your
y = df['target_column']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Train XGBoost model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6, 
xgb_model.fit(X_train, y_train)

# Predictions
y_pred = xgb_model.predict(X_test)

# Evaluate the model
print("XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_33244\3346436127.py in <module>
      4
      5 # Split data into features and target
----> 6 X = df.drop(columns=['target_column'])  # Replace 'target_column' with your actual target
      7 y = df['target_column']
      8

C:\ProgramData\Anaconda3\lib\site-packages\pandas\util\_decorators.py in wrapper(*args, **kwargs)
    309                     stacklevel=stacklevel,
    310                 )
--> 311             return func(*args, **kwargs)
    312
    313         return wrapper

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   4955                 weight  1.0     0.8
   4956         """
-> 4957         return super().drop(
   4958             labels=labels,
   4959             axis=axis,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in drop(self, labels, axis, index, columns, level, inplace, errors)
   4265         for axis, labels in axes.items():
   4266             if labels is not None:
-> 4267                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   4268
   4269         if inplace:

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\generic.py in _drop_axis(self, labels, axis, level, errors, consolidate, only_slice)
   4309                 new_axis = axis.drop(labels, level=level, errors=errors)
   4310             else:
-> 4311                 new_axis = axis.drop(labels, errors=errors)
   4312             indexer = axis.get_indexer(new_axis)
   4313

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\indexes\base.py in drop(self, labels, errors)
   6659         if mask.any():
   6660             if errors != "ignore":
-> 6661                 raise KeyError(f"{list(labels[mask])} not found in axis")
   6662             indexer = indexer[~mask]
   6663         return self.delete(indexer)

KeyError: "['target_column'] not found in axis"
```

In [124]:
```
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Check dataset columns
print(df.columns)

# Define features and target
X = df.drop(columns=['Rating'])  # Replace 'Rating' with your actual target
y = df['Rating']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# Train XGBoost model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6,
xgb_model.fit(X_train, y_train)

# Predictions
y_pred = xgb_model.predict(X_test)

# Evaluate the model
print("XGBoost Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Index(['Name', 'Phone', 'Address', 'Email', 'Website', 'ServiceArea',
       'Instagram', 'Facebook', 'Twitter', 'Linkedin', 'Youtube',
       'BusinessUrl', 'Rating', 'ReviewCount', 'PriceRange', 'Longitud
e',
       'Latitude', 'Alias', 'BizId', 'BusinessSectionUrls_open_hours',
       'BusinessSectionUrls_reviews', 'Categories_0_title', 'Categories_
0_url',
       'Categories_1_title', 'Categories_1_url', 'Categories_2_title',
       'Categories_2_url', 'Categories_3_title', 'Categories_3_url',
       'Categories_4_title', 'Categories_4_url', 'FormattedAddress', 'Is
Ad',
       'Open_time', 'ParentBusiness', 'Ranking', 'RenderAdInfo',
       'ServicePricing', 'Snippet', 'temperature', 'busy_time',
       'temp_busy_interaction'],
      dtype='object')

---------------------------------------------------------------------------
---
ValueError                                Traceback (most recent call la
st)
```

In [125]:

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_33244\4045642178.py in <module>
     12 # 3. Train XGBoost Model
     13 xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6, random_state=42)
---> 14 xgb_model.fit(X_train, y_train)
     15
     16 # 4. Make Predictions

~\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py in inner_f(*args, **kwargs)
    724             for k, arg in zip(sig.parameters, args):
    725                 kwargs[k] = arg
--> 726             return func(**kwargs)
    727
    728         return inner_f
```

In [126]:
```
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# 1. Select Features and Target
X = df.drop(columns=['Rating'])  # Assuming 'Rating' is the target column
y = df['Rating']

# 2. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# 3. Train XGBoost Model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6, 
xgb_model.fit(X_train, y_train)

# 4. Make Predictions
y_pred = xgb_model.predict(X_test)

# 5. Evaluate Model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("XGBoost Model Performance:")
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)
```

```
---------------------------------------------------------------------------
---
ValueError                                Traceback (most recent call la
st)
~\AppData\Local\Temp\ipykernel_33244\4045642178.py in <module>
     12 # 3. Train XGBoost Model
     13 xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, ma
x_depth=6, random_state=42)
---> 14 xgb_model.fit(X_train, y_train)
     15
     16 # 4. Make Predictions

~\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py in inner
_f(*args, **kwargs)
    724             for k, arg in zip(sig.parameters, args):
    725                 kwargs[k] = arg
--> 726             return func(**kwargs)
    727
    728         return inner_f
```

In [127]:
```
from sklearn.preprocessing import LabelEncoder
from xgboost import XGBRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# 1. Drop Irrelevant Columns
irrelevant_cols = ['Name', 'Phone', 'Address', 'Email', 'Website', 'Instagra
                   'Facebook', 'Twitter', 'Linkedin', 'Youtube', 'BusinessU
                   'Categories_0_title', 'Categories_0_url', 'Categories_1_
                   'Categories_1_url', 'Categories_2_title', 'Categories_2_
                   'Categories_3_title', 'Categories_3_url', 'Categories_4_
                   'Categories_4_url', 'Snippet']  # Update based on your n
df = df.drop(columns=irrelevant_cols)

# 2. Encode Categorical Columns
label_encoder = LabelEncoder()
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    df[col] = label_encoder.fit_transform(df[col])

# 3. Separate Features and Target
X = df.drop(columns=['Rating'])  # Replace 'Rating' with your target column
y = df['Rating']

# 4. Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra

# 5. Train XGBoost Model
xgb_model = XGBRegressor(n_estimators=100, learning_rate=0.1, max_depth=6, 
xgb_model.fit(X_train, y_train)

# 6. Make Predictions
y_pred = xgb_model.predict(X_test)

# 7. Evaluate Model
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("XGBoost Model Performance:")
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)
```

```
XGBoost Model Performance:
Mean Absolute Error (MAE): 0.23837389945983878
Mean Squared Error (MSE): 0.08388744844817198
R² Score: -2.728331042140981
```

In [128]:
```
# Analyze Target Distribution
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(y, kde=True)
plt.title("Target Variable Distribution")
plt.show()

# Feature Importance
import numpy as np
feature_importances = xgb_model.feature_importances_
sorted_indices = np.argsort(feature_importances)[::-1]
important_features = X.columns[sorted_indices[:10]]  # Top 10 features
print("Top 10 Important Features:")
print(important_features)

# Hyperparameter Tuning (Example with GridSearchCV)
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 6, 9],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1],
    'colsample_bytree': [0.8, 1]
```

```
  File "C:\Users\shubh\AppData\Local\Temp\ipykernel_33244\4032536339.py", line 23
    'colsample_bytree': [0.8, 1]
                                 ^
SyntaxError: unexpected EOF while parsing
```

In [129]:
```
# Hyperparameter Tuning (Example with GridSearchCV)
from sklearn.model_selection import GridSearchCV
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [3, 6, 9],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.8, 1],
    'colsample_bytree': [0.8, 1]  # Fixed: Ensure this is properly closed
}  # Closing the dictionary

grid_search = GridSearchCV(XGBRegressor(random_state=42), param_grid, scori
grid_search.fit(X_train, y_train)

# Displaying best parameters
print("Best Parameters:", grid_search.best_params_)

# Re-train the model with best parameters
best_xgb_model = grid_search.best_estimator_
y_pred = best_xgb_model.predict(X_test)

# Re-evaluate Performance
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco
print("Improved XGBoost Performance:")
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Fitting 3 folds for each of 108 candidates, totalling 324 fits
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=50, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=50, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=50, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=50, subsample=1; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=50, subsample=1; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=50, subsample=1; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=0.8; total time=   0.0s
[CV] END colsample_bytree=0.8, learning_rate=0.01, max_depth=3, n_estima
tors=100, subsample=0.8; total time=   0.0s
```

In [131]:
```
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 6, 9, 12],
    'learning_rate': [0.01, 0.05, 0.1],
    'subsample': [0.6, 0.8, 1],
    'colsample_bytree': [0.6, 0.8, 1]
}
```

In [132]:
```
from sklearn.ensemble import RandomForestRegressor

# Initialize and train the Random Forest model
rf_model = RandomForestRegressor(n_estimators=200, max_depth=10, random_sta
rf_model.fit(X_train, y_train)

# Predictions
y_pred_rf = rf_model.predict(X_test)

# Evaluate the Random Forest model
print("Random Forest Performance:")
print("Mean Absolute Error (MAE):", mean_absolute_error(y_test, y_pred_rf))
print("Mean Squared Error (MSE):", mean_squared_error(y_test, y_pred_rf))
print("R² Score:", r2_score(y_test, y_pred_rf))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_33244\3505081756.py in <module>
      3 # Initialize and train the Random Forest model
      4 rf_model = RandomForestRegressor(n_estimators=200, max_depth=10, random_state=42)
----> 5 rf_model.fit(X_train, y_train)
      6
      7 # Predictions

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in fit(self, X, y, sample_weight)
    325         if issparse(y):
    326             raise ValueError("sparse multilabel-indicator for y is not supported.")
--> 327         X, y = self._validate_data(
    328             X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
    329         )

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separately, **check_params)
    579                 y = check_array(y, **check_y_params)
    580             else:
--> 581                 X, y = check_X_y(X, y, **check_params)
    582             out = X, y
    583

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    962         raise ValueError("y cannot be None")
    963
--> 964     X = check_array(
    965         X,
    966         accept_sparse=accept_sparse,

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    798
    799         if force_all_finite:
--> 800             _assert_all_finite(array, allow_nan=force_all_finite == "allow-nan")
    801
    802     if ensure_min_samples > 0:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in _assert_all_finite(X, allow_nan, msg_dtype)
    112         ):
    113             type_err = "infinity" if allow_nan else "NaN, infinity"
--> 114             raise ValueError(
    115                 msg_err.format(
    116                     type_err, msg_dtype if msg_dtype is not None else X.dtype
```

```
ValueError: Input contains NaN, infinity or a value too large for dtype('f
loat32').
```

In [ ]:
```
!pip install catboost
```

In [140]:
```
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegresso
from catboost import CatBoostRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

# Initialize models
models = {
    'Random Forest': RandomForestRegressor(n_estimators=200, max_depth=10, 
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=200, learni
    'CatBoost': CatBoostRegressor(n_estimators=200, learning_rate=0.1, deptl
}

# Dictionary to store results
results = {}

for name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Evaluate the model
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Store the results
    results[name] = {'MAE': mae, 'MSE': mse, 'R2': r2}

    print(f"{name} Performance:")
    print(f"Mean Absolute Error (MAE): {mae}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R² Score: {r2}\n")

# Display the best model
best_model = max(results, key=lambda x: results[x]['R2'])
print(f"Best Model: {best_model} with R² Score: {results[best_model]['R2']}
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\483527583.py in <module>
     15 for name, model in models.items():
     16     # Train the model
---> 17     model.fit(X_train, y_train)
     18
     19     # Make predictions

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in
fit(self, X, y, sample_weight)
    325         if issparse(y):
    326             raise ValueError("sparse multilabel-indicator for y is
not supported.")
--> 327         X, y = self._validate_data(
    328             X, y, multi_output=True, accept_sparse="csc", dtype=DT
YPE
    329         )

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_da
ta(self, X, y, reset, validate_separately, **check_params)
    579                 y = check_array(y, **check_y_params)
    580             else:
--> 581                 X, y = check_X_y(X, y, **check_params)
    582             out = X, y
    583

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, fo
rce_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ens
ure_min_features, y_numeric, estimator)
    962         raise ValueError("y cannot be None")
    963
--> 964     X = check_array(
    965         X,
    966         accept_sparse=accept_sparse,

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,
force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_feat
ures, estimator)
    798
    799         if force_all_finite:
--> 800             _assert_all_finite(array, allow_nan=force_all_finite =
= "allow-nan")
    801
    802     if ensure_min_samples > 0:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
_assert_all_finite(X, allow_nan, msg_dtype)
    112         ):
    113             type_err = "infinity" if allow_nan else "NaN, infinit
y"
--> 114             raise ValueError(
    115                 msg_err.format(
    116                     type_err, msg_dtype if msg_dtype is not None e
lse X.dtype

ValueError: Input contains NaN, infinity or a value too large for dtype('f
```

```
loat32').
```

In [138]:
```
pip install catboost
```

```
Defaulting to user installation because normal site-packages is not writea
bleNote: you may need to restart the kernel to use updated packages.

Collecting catboost
  Downloading catboost-1.2.7-cp39-cp39-win_amd64.whl (101.8 MB)
     ---------------------------------- 101.8/101.8 MB 861.2 kB/s eta 0:
00:00
Requirement already satisfied: plotly in c:\programdata\anaconda3\lib\site
-packages (from catboost) (5.9.0)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib
\site-packages (from catboost) (3.5.2)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from catboost) (1.9.1)
Requirement already satisfied: numpy<2.0,>=1.16.0 in c:\programdata\anacon
da3\lib\site-packages (from catboost) (1.21.5)
Requirement already satisfied: pandas>=0.24 in c:\programdata\anaconda3\li
b\site-packages (from catboost) (1.4.4)
Collecting graphviz
  Downloading graphviz-0.20.3-py3-none-any.whl (47 kB)
     ---------------------------------------- 47.1/47.1 kB 2.3 MB/s eta 0:
00:00
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-pa
ckages (from catboost) (1.16.0)
Requirement already satisfied: pytz>=2020.1 in c:\programdata\anaconda3\li
b\site-packages (from pandas>=0.24->catboost) (2022.1)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\programdata\an
aconda3\lib\site-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pyparsing>=2.2.1 in c:\programdata\anaconda
3\lib\site-packages (from matplotlib->catboost) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anacond
a3\lib\site-packages (from matplotlib->catboost) (4.25.0)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->catboost) (21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anacond
a3\lib\site-packages (from matplotlib->catboost) (1.4.2)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\li
b\site-packages (from matplotlib->catboost) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\l
ib\site-packages (from matplotlib->catboost) (9.2.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\shubh\appdata\r
oaming\python\python39\site-packages (from plotly->catboost) (8.3.0)
Installing collected packages: graphviz, catboost
Successfully installed catboost-1.2.7 graphviz-0.20.3
```

In [142]:
```
import numpy as np

# Replace infinity values with NaN
X_train.replace([np.inf, -np.inf], np.nan, inplace=True)
X_test.replace([np.inf, -np.inf], np.nan, inplace=True)

# Drop or fill missing values
X_train.fillna(X_train.mean(), inplace=True)
X_test.fillna(X_test.mean(), inplace=True)
```

In [143]:
```
print("NaN values in X_train:", X_train.isnull().sum().sum())
print("NaN values in X_test:", X_test.isnull().sum().sum())
```

```
NaN values in X_train: 40
NaN values in X_test: 10
```

In [144]:
```
print(X_train.dtypes)
```

```
ServiceArea                      float64
ReviewCount                        int64
PriceRange                         int32
Longitude                        float64
Latitude                         float64
Alias                              int32
BizId                              int32
BusinessSectionUrls_open_hours     int32
BusinessSectionUrls_reviews        int32
FormattedAddress                 float64
IsAd                                bool
Open_time                        float64
ParentBusiness                   float64
Ranking                            int64
RenderAdInfo                        bool
ServicePricing                   float64
temperature                        int64
busy_time                          int64
temp_busy_interaction              int64
dtype: object
```

In [145]:
```
# Evaluate each model
for name, model in models.items():
    print(f"\n{name} Performance:")
    y_pred = model.predict(X_test)

    # Performance metrics
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    print(f"Mean Absolute Error (MAE): {mae}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R² Score: {r2}")
```

```
Random Forest Performance:
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\1445970307.py in <module>
      2 for name, model in models.items():
      3     print(f"\n{name} Performance:")
----> 4     y_pred = model.predict(X_test)
      5
      6     # Performance metrics

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in
predict(self, X)
    969         check_is_fitted(self)
    970         # Check data
--> 971         X = self._validate_X_predict(X)
    972
    973         # Assign chunk of trees to jobs

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in
_validate_X_predict(self, X)
    577         Validate X whenever one tries to predict, apply, predict_p
roba."""
    578         check_is_fitted(self)
--> 579         X = self._validate_data(X, dtype=DTYPE, accept_sparse="cs
r", reset=False)
    580         if issparse(X) and (X.indices.dtype != np.intc or X.indpt
r.dtype != np.intc):
    581             raise ValueError("No support for np.int64 index based
sparse matrices")

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_da
ta(self, X, y, reset, validate_separately, **check_params)
    564             raise ValueError("Validation should be done on X, y or
both.")
    565         elif not no_val_X and no_val_y:
--> 566             X = check_array(X, **check_params)
    567             out = X
    568         elif no_val_X and not no_val_y:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,
force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_feat
ures, estimator)
    798
    799         if force_all_finite:
--> 800             _assert_all_finite(array, allow_nan=force_all_finite =
= "allow-nan")
    801
    802     if ensure_min_samples > 0:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
_assert_all_finite(X, allow_nan, msg_dtype)
    112         ):
    113             type_err = "infinity" if allow_nan else "NaN, infinit
y"
--> 114             raise ValueError(
    115                 msg_err.format(
    116                     type_err, msg_dtype if msg_dtype is not None e
lse X.dtype
```

```
ValueError: Input contains NaN, infinity or a value too large for dtype('f
loat32').
```

In [146]:
```
import numpy as np

# Check for NaN or Infinity in X_test
if np.any(np.isnan(X_test)) or np.any(np.isinf(X_test)):
    print("NaN or Infinity found in X_test. Cleaning the data...")
    X_test = np.nan_to_num(X_test, nan=0.0, posinf=np.finfo(np.float32).max
```

```
NaN or Infinity found in X_test. Cleaning the data...
```

In [147]:
```
for name, model in models.items():
    print(f"\n{name} Performance:")

    # Make predictions on the cleaned X_test
    y_pred = model.predict(X_test)

    # Calculate performance metrics
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Print metrics
    print(f"Mean Absolute Error (MAE): {mae}")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R² Score: {r2}")
```

```
Random Forest Performance:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarnin
g: X does not have valid feature names, but RandomForestRegressor was fitt
ed with feature names
  warnings.warn(
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\3667026806.py in <module>
      3
      4     # Make predictions on the cleaned X_test
----> 5     y_pred = model.predict(X_test)
      6
      7     # Calculate performance metrics

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in
predict(self, X)
    969         check_is_fitted(self)
    970         # Check data
--> 971         X = self._validate_X_predict(X)
    972
    973         # Assign chunk of trees to jobs

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in
_validate_X_predict(self, X)
    577         Validate X whenever one tries to predict, apply, predict_p
roba."""
    578         check_is_fitted(self)
--> 579         X = self._validate_data(X, dtype=DTYPE, accept_sparse="cs
r", reset=False)
    580         if issparse(X) and (X.indices.dtype != np.intc or X.indpt
r.dtype != np.intc):
    581             raise ValueError("No support for np.int64 index based
sparse matrices")

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_da
ta(self, X, y, reset, validate_separately, **check_params)
    564             raise ValueError("Validation should be done on X, y or
both.")
    565         elif not no_val_X and no_val_y:
--> 566             X = check_array(X, **check_params)
    567             out = X
    568         elif no_val_X and not no_val_y:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy,
force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_feat
ures, estimator)
    798
    799         if force_all_finite:
--> 800             _assert_all_finite(array, allow_nan=force_all_finite =
= "allow-nan")
    801
    802     if ensure_min_samples > 0:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in
_assert_all_finite(X, allow_nan, msg_dtype)
    112         ):
    113             type_err = "infinity" if allow_nan else "NaN, infinit
y"
--> 114             raise ValueError(
    115                 msg_err.format(
    116                     type_err, msg_dtype if msg_dtype is not None e
lse X.dtype
```

```
ValueError: Input contains NaN, infinity or a value too large for dtype('f
loat32').
```

In [148]:
```
# Check for NaN or infinity values in X_test
print("NaN values in X_test:", np.isnan(X_test).sum().sum())
print("Infinity values in X_test:", np.isinf(X_test).sum().sum())

# Display summary statistics
print("X_test summary statistics:")
print(pd.DataFrame(X_test).describe())
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\2406769108.py in <module>
      1 # Check for NaN or infinity values in X_test
----> 2 print("NaN values in X_test:", np.isnan(X_test).sum().sum())
      3 print("Infinity values in X_test:", np.isinf(X_test).sum().sum())
      4
      5 # Display summary statistics

TypeError: ufunc 'isnan' not supported for the input types, and the inputs
could not be safely coerced to any supported types according to the castin
g rule ''safe''
```

In [149]:
```
print(type(X_test))
```

```
<class 'numpy.ndarray'>
```

In [150]:
```
print("X_test dtype:", X_test.dtype)
```

```
X_test dtype: object
```

In [151]:
```
X_test = X_test.astype(float)
```

In [152]:
```
# Replace NaN with 0 and Inf/-Inf with maximum/minimum float values
X_test = np.nan_to_num(X_test, nan=0.0, posinf=np.finfo(np.float32).max, ne

# Verify the cleanup
print("NaN values in X_test after cleaning:", np.isnan(X_test).sum())
print("Infinity values in X_test after cleaning:", np.isinf(X_test).sum())
```

```
NaN values in X_test after cleaning: 0
Infinity values in X_test after cleaning: 0
```

In [153]:
```
for name, model in models.items():
    print(f"\n{name} Performance:")
    y_pred = model.predict(X_test)

    # Calculate performance metrics
    print("MAE:", mean_absolute_error(y_test, y_pred))
    print("MSE:", mean_squared_error(y_test, y_pred))
    print("R²:", r2_score(y_test, y_pred))
```

```
Random Forest Performance:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarnin
g: X does not have valid feature names, but RandomForestRegressor was fitt
ed with feature names
  warnings.warn(

---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\2146738923.py in <module>
      1 for name, model in models.items():
      2     print(f"\n{name} Performance:")
----> 3     y_pred = model.predict(X_test)
      4
      5     # Calculate performance metrics

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in
predict(self, X)
    975
    976         # avoid storing the output of every estimator by summing t
hem here
--> 977         if self.n_outputs_ > 1:
    978             y_hat = np.zeros((X.shape[0], self.n_outputs_), dtype=
np.float64)
    979         else:

AttributeError: 'RandomForestRegressor' object has no attribute 'n_outputs
_'
```

In [154]:
```
from sklearn.utils.validation import check_is_fitted

for name, model in models.items():
    try:
        check_is_fitted(model)
        print(f"{name} is successfully trained.")
    except:
        print(f"{name} is not properly trained.")
```

```
Random Forest is successfully trained.
Gradient Boosting is not properly trained.
CatBoost is not properly trained.
```

In [155]:
```
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("NaN in X_train:", np.isnan(X_train).sum())
print("NaN in y_train:", np.isnan(y_train).sum())
```

```
X_train shape: (8, 19)
y_train shape: (8,)
NaN in X_train: ServiceArea                      8
ReviewCount                      0
PriceRange                       0
Longitude                        0
Latitude                         0
Alias                            0
BizId                            0
BusinessSectionUrls_open_hours   0
BusinessSectionUrls_reviews      0
FormattedAddress                 8
IsAd                             0
Open_time                        8
ParentBusiness                   8
Ranking                          0
RenderAdInfo                     0
ServicePricing                   8
temperature                      0
busy_time                        0
temp_busy_interaction            0
dtype: int64
NaN in y_train: 0
```

In [156]:
```
for name, model in models.items():
    model.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_33244\1070650683.py in <module>
      1 for name, model in models.items():
----> 2     model.fit(X_train, y_train)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\ensemble\_forest.py in fit(self, X, y, sample_weight)
    325         if issparse(y):
    326             raise ValueError("sparse multilabel-indicator for y is not supported.")
--> 327         X, y = self._validate_data(
    328             X, y, multi_output=True, accept_sparse="csc", dtype=DTYPE
    329         )

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py in _validate_data(self, X, y, reset, validate_separately, **check_params)
    579                 y = check_array(y, **check_y_params)
    580             else:
--> 581                 X, y = check_X_y(X, y, **check_params)
    582             out = X, y
    583

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_X_y(X, y, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, multi_output, ensure_min_samples, ensure_min_features, y_numeric, estimator)
    962         raise ValueError("y cannot be None")
    963
--> 964     X = check_array(
    965         X,
    966         accept_sparse=accept_sparse,

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, ensure_min_features, estimator)
    798
    799         if force_all_finite:
--> 800             _assert_all_finite(array, allow_nan=force_all_finite == "allow-nan")
    801
    802     if ensure_min_samples > 0:

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py in _assert_all_finite(X, allow_nan, msg_dtype)
    112         ):
    113             type_err = "infinity" if allow_nan else "NaN, infinity"
--> 114             raise ValueError(
    115                 msg_err.format(
    116                     type_err, msg_dtype if msg_dtype is not None else X.dtype

ValueError: Input contains NaN, infinity or a value too large for dtype('float32').
```

In [157]:
```
# Replace NaN or inf with median in X_train and X_test
from sklearn.impute import SimpleImputer
import numpy as np

imputer = SimpleImputer(missing_values=np.nan, strategy='median')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Ensure there are no infinite values
X_train[np.isinf(X_train)] = np.nan
X_test[np.isinf(X_test)] = np.nan

# Reapply imputation to handle replaced NaN
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Handle y_train and y_test
y_train = np.nan_to_num(y_train, nan=np.median(y_train))
y_test = np.nan_to_num(y_test, nan=np.media
```

```
  File "C:\Users\shubh\AppData\Local\Temp\ipykernel_33244\760876028.py", l
ine 19
    y_test = np.nan_to_num(y_test, nan=np.media
                                               ^
SyntaxError: unexpected EOF while parsing
```

In [158]:
```
# Replace NaN or inf with median in X_train and X_test
from sklearn.impute import SimpleImputer
import numpy as np

imputer = SimpleImputer(missing_values=np.nan, strategy='median')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Ensure there are no infinite values
X_train[np.isinf(X_train)] = np.nan
X_test[np.isinf(X_test)] = np.nan

# Reapply imputation to handle replaced NaN
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Handle y_train and y_test
y_train = np.nan_to_num(y_train, nan=np.median(y_train))
y_test = np.nan_to_num(y_test, nan=np.median(y_test))
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\base.py:450: UserWarnin
g: X does not have valid feature names, but SimpleImputer was fitted with
feature names
  warnings.warn(
```

In [159]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [160]:
```
print("X_train shape:", X_train.shape)
print("X_test shape:", X_test.shape)
print("NaN in X_train:", np.isnan(X_train).sum())
print("NaN in X_test:", np.isnan(X_test).sum())
print("Max value in X_train:", np.max(X_train))
print("Max value in X_test:", np.max(X_test))
```

```
X_train shape: (8, 14)
X_test shape: (2, 14)
NaN in X_train: 0
NaN in X_test: 0
Max value in X_train: 2.6457513110645903
Max value in X_test: 1.299867367239363
```

In [161]:
```
for name, model in models.items():
    model.fit(X_train, y_train)
    print(f"{name} successfully trained.")
```

```
Random Forest successfully trained.
Gradient Boosting successfully trained.
CatBoost successfully trained.
```

In [162]:
```
for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"\n{name} Performance:")
    print("MAE:", mean_absolute_error(y_test, y_pred))
    print("MSE:", mean_squared_error(y_test, y_pred))
    print("R²:", r2_score(y_test, y_pred))
```

```
Random Forest Performance:
MAE: 0.19724999999999637
MSE: 0.04412762499999882
R²: -0.9612277777777276

Gradient Boosting Performance:
MAE: 0.2710947020328498
MSE: 0.07531644146794239
R²: -2.3473973985752217

CatBoost Performance:
MAE: 0.3727569282814571
MSE: 0.1511500144230224
R²: -5.717778418801004
```

In [163]:
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco
import numpy as np

# Define the Random Forest Regressor
rf = RandomForestRegressor(random_state=42)

# Define the parameter grid for RandomizedSearchCV
param_grid = {
    'n_estimators': [100, 200, 300, 400, 500],
    'max_depth': [None, 10, 20, 30, 40],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Perform RandomizedSearchCV
random_search = RandomizedSearchCV(
    estimator=rf,
    param_distributions=param_grid,
    n_iter=50,
    scoring='neg_mean_squared_error',
    cv=3,
    verbose=2,
    random_state=42,
    n_jobs=-1
)

# Fit the RandomizedSearchCV to the data
random_search.fit(X_train, y_train)

# Best parameters and model
tuned_rf = random_search.best_estimator_
best_params = random_search.best_params_
print("Best Parameters:", best_params)

# Evaluate the tuned model
y_pred = tuned_rf.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nTuned Random Forest Performance:")
print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("R² Score:", r2)
```

```
Fitting 3 folds for each of 50 candidates, totalling 150 fits
Best Parameters: {'n_estimators': 300, 'min_samples_split': 5, 'min_sample
s_leaf': 1, 'max_features': 'auto', 'max_depth': 20}

Tuned Random Forest Performance:
Mean Absolute Error (MAE): 0.18016547619047296
Mean Squared Error (MSE): 0.04541356143140422
R² Score: -1.0183805080624126
```

In [164]:
```
import joblib

# Save the trained Random Forest model to a file
model_filename = 'tuned_random_forest_model.pkl'
joblib.dump(best_model, model_filename)
print(f"Model saved as {model_filename}")
```

```
Model saved as tuned_random_forest_model.pkl
```

In [165]:
```
# Load the saved model
loaded_model = joblib.load(model_filename)

# Make predictions with the loaded model
y_pred = loaded_model.predict(X_test)
print(f"Predictions: {y_pred}")
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\755928901.py in <module>
      3
      4 # Make predictions with the loaded model
----> 5 y_pred = loaded_model.predict(X_test)
      6 print(f"Predictions: {y_pred}")

~\AppData\Roaming\Python\Python39\site-packages\xgboost\sklearn.py in pred
ict(self, X, output_margin, validate_features, base_margin, iteration_rang
e)
   1184             if self._can_use_inplace_predict():
   1185                 try:
-> 1186                     predts = self.get_booster().inplace_predict(
   1187                         data=X,
   1188                         iteration_range=iteration_range,

~\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py in inplace
_predict(self, data, iteration_range, predict_type, missing, validate_feat
ures, base_margin, strict_shape)
   2522                 )
   2523             if len(data.shape) != 1 and self.num_features() != dat
a.shape[1]:
-> 2524                 raise ValueError(
   2525                     f"Feature shape mismatch, expected: {self.num_
features()}, "
   2526                     f"got {data.shape[1]}"

ValueError: Feature shape mismatch, expected: 10, got 14
```

In [166]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
import joblib

# Example: Preprocessing and model pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), list(X_train.columns)),  # Scale numeric j
    ]
)

pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('model', best_model)  # Use the tuned Random Forest model
])

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

# Save the pipeline
pipeline_filename = 'random_forest_pipeline.pkl'
joblib.dump(pipeline, pipeline_filename)
print(f"Pipeline saved as {pipeline_filename}")
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\182222383.py in <module>
      7 preprocessor = ColumnTransformer(
      8     transformers=[
----> 9         ('num', StandardScaler(), list(X_train.columns)),  # Scale
numeric features
     10     ]
     11 )

AttributeError: 'numpy.ndarray' object has no attribute 'columns'
```

In [171]:
```
import pandas as pd
import joblib

# Load the pipeline
pipeline = joblib.load('tuned_random_forest_model.pkl')

# If X_test is a NumPy array, convert it back to DataFrame
if isinstance(X_test, np.ndarray):
    column_names = ['feature1', 'feature2', ..., 'featureN']  # Add actual j
    X_test = pd.DataFrame(X_test, columns=column_names)

# Align columns with training data
X_test_aligned = X_test[X_train.columns]

# Make predictions
y_pred = pipeline.predict(X_test_aligned)
print(f"Predictions: {y_pred}")
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\1686163636.py in <module>
      8 if isinstance(X_test, np.ndarray):
      9     column_names = ['feature1', 'feature2', ..., 'featureN']  # Ad
d actual feature names
---> 10     X_test = pd.DataFrame(X_test, columns=column_names)
     11
     12 # Align columns with training data

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\frame.py in __init_
_(self, data, index, columns, dtype, copy)
    692                 )
    693             else:
--> 694                 mgr = ndarray_to_mgr(
    695                     data,
    696                     index,

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\construct
ion.py in ndarray_to_mgr(values, index, columns, dtype, copy, typ)
    349     )
    350
--> 351     _check_values_indices_shape_match(values, index, columns)
    352
    353     if typ == "array":

C:\ProgramData\Anaconda3\lib\site-packages\pandas\core\internals\construct
ion.py in _check_values_indices_shape_match(values, index, columns)
    420         passed = values.shape
    421         implied = (len(index), len(columns))
--> 422         raise ValueError(f"Shape of passed values is {passed}, ind
ices imply {implied}")
    423
    424

ValueError: Shape of passed values is (2, 14), indices imply (2, 4)
```

In [172]:
```
import pandas as pd

# Assuming X_test is a NumPy array
print(f"Shape of X_test: {X_test.shape}")

# Correct number of column names for the data
column_names = ['feature1', 'feature2', ..., 'feature14']  # Add all 14 nam

# Convert to DataFrame
try:
    X_test_df = pd.DataFrame(X_test, columns=column_names)
    print("X_test successfully converted to DataFrame.")
except ValueError as e:
    print(f"Error: {e}")
```

```
Shape of X_test: (2, 14)
Error: Shape of passed values is (2, 14), indices imply (2, 4)
```

In [170]:
```
# Align X_test with X_train columns
X_test_aligned = X_test_df[X_train.columns]
```

```
---------------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\2731562584.py in <module>
      1 # Align X_test with X_train columns
----> 2 X_test_aligned = X_test_df[X_train.columns]

NameError: name 'X_test_df' is not defined
```

In [173]:
```
# Make predictions
y_pred = loaded_model.predict(X_test)

# Evaluate performance
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_sco

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Model Performance on Test Data:")
print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"R² Score: {r2}")
```

```
---------------------------------------------------------------------------
-
ValueError                                Traceback (most recent call las
t)
~\AppData\Local\Temp\ipykernel_33244\4266660602.py in <module>
      1 # Make predictions
----> 2 y_pred = loaded_model.predict(X_test)
      3
      4 # Evaluate performance
      5 from sklearn.metrics import mean_absolute_error, mean_squared_erro
r, r2_score

~\AppData\Roaming\Python\Python39\site-packages\xgboost\sklearn.py in pred
ict(self, X, output_margin, validate_features, base_margin, iteration_rang
e)
   1184             if self._can_use_inplace_predict():
   1185                 try:
-> 1186                     predts = self.get_booster().inplace_predict(
   1187                         data=X,
   1188                         iteration_range=iteration_range,

~\AppData\Roaming\Python\Python39\site-packages\xgboost\core.py in inplace
_predict(self, data, iteration_range, predict_type, missing, validate_feat
ures, base_margin, strict_shape)
   2522                 )
   2523             if len(data.shape) != 1 and self.num_features() != dat
a.shape[1]:
-> 2524                 raise ValueError(
   2525                     f"Feature shape mismatch, expected: {self.num_
features()}, "
   2526                     f"got {data.shape[1]}"

ValueError: Feature shape mismatch, expected: 10, got 14
```

In [174]:
```
# Check column alignment
print("Train Features:", X_train.columns)
print("Test Features:", X_test.columns)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_33244\2632926398.py in <module>
      1 # Check column alignment
----> 2 print("Train Features:", X_train.columns)
      3 print("Test Features:", X_test.columns)

AttributeError: 'numpy.ndarray' object has no attribute 'columns'
```

In [ ]:
```
```