Analysis on Marketing Campaign Effectiveness

- Programming Assignment
- Nomondalai Batjargal
- December 01, 2023

Introduction

In this analysis, I delve into a comprehensive exploration of a marketing campaign dataset, encompassing data preprocessing, visualization, and the development of machine learning models. The objective is twofold: first, to predict revenue generated from marketing efforts, and second, to forecast responses to the campaigns. By doing so, I aim to unearth the critical drivers of campaign effectiveness and facilitate data-driven decisions for future marketing endeavors.

- **Problem Definition:** The problem at hand is that many companies spend endless amount of resources on marketing without actually knowing which campaigns lead to more profit or which campaigns were ineffective.

To this end, I have conducted a comprehensive analysis of a sample dataset, delving into various factors such as market size, promotion strategies, location impact, and predictive modeling. By examining these key aspects, I aim to provide actionable insights that can inform strategic decision-making and drive marketing campaign optimization. This analysis serves as a valuable tool for businesses looking to enhance their marketing effectiveness and achieve sustainable growth.

In [58]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifi
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [15]:
```
# Load the dataset
data = pd.read_csv("/content/Marketing_Campaign_Effectiveness.csv")
```

Data Exploration and Visualization

In [16]:
```
#Display dataset
data.head(4)
```

Out[16]:

	MarketID	MarketSize	LocationID	AgeOfStore	Promotion	week	SalesInThousands
0	1	Medium	1	4	3	1	33.73
1	1	Medium	1	4	3	2	35.67
2	1	Medium	1	4	3	3	29.03
3	1	Medium	1	4	3	4	39.25

In [17]:
```
# Display detailed info about the dataset
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 548 entries, 0 to 547
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   MarketID          548 non-null    int64  
 1   MarketSize        548 non-null    object 
 2   LocationID        548 non-null    int64  
 3   AgeOfStore        548 non-null    int64  
 4   Promotion         548 non-null    int64  
 5   week              548 non-null    int64  
 6   SalesInThousands  548 non-null    float64
dtypes: float64(1), int64(5), object(1)
memory usage: 30.1+ KB
```

In [18]:
```
# Describe the dataset
data.describe()
```

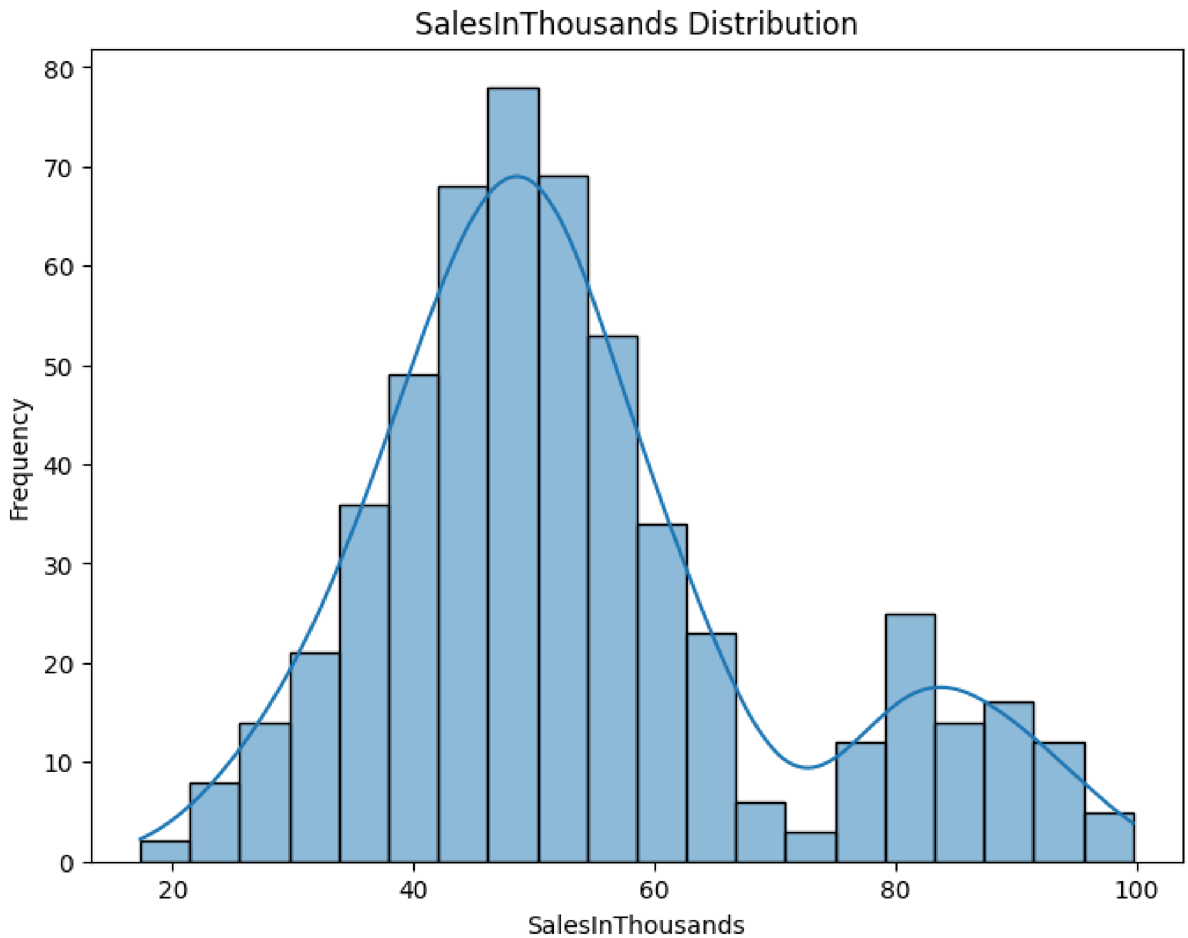
Out[18]:

	MarketID	LocationID	AgeOfStore	Promotion	week	SalesInThousands
count	548.000000	548.000000	548.000000	548.000000	548.000000	548.000000
mean	5.715328	479.656934	8.503650	2.029197	2.500000	53.466204
std	2.877001	287.973679	6.638345	0.810729	1.119055	16.755216
min	1.000000	1.000000	1.000000	1.000000	1.000000	17.340000
25%	3.000000	216.000000	4.000000	1.000000	1.750000	42.545000
50%	6.000000	504.000000	7.000000	2.000000	2.500000	50.200000
75%	8.000000	708.000000	12.000000	3.000000	3.250000	60.477500
max	10.000000	920.000000	28.000000	3.000000	4.000000	99.650000

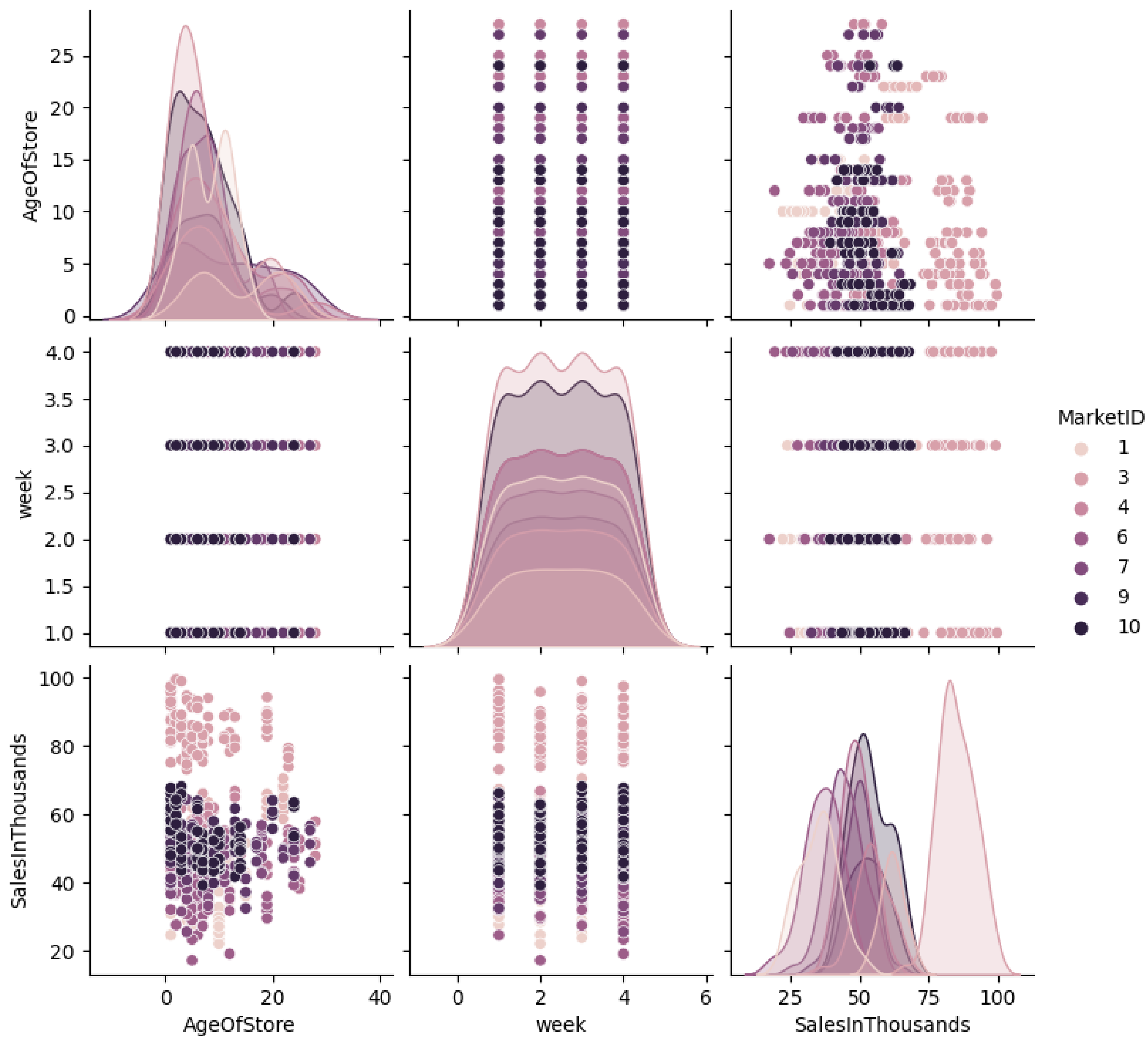
In [30]:
```
# List all column names in the dataset
column_names = data.columns
print(column_names)
```

```
Index(['MarketID', 'MarketSize', 'LocationID', 'AgeOfStore', 'Promotio
n',
       'week', 'SalesInThousands'],
      dtype='object')
```

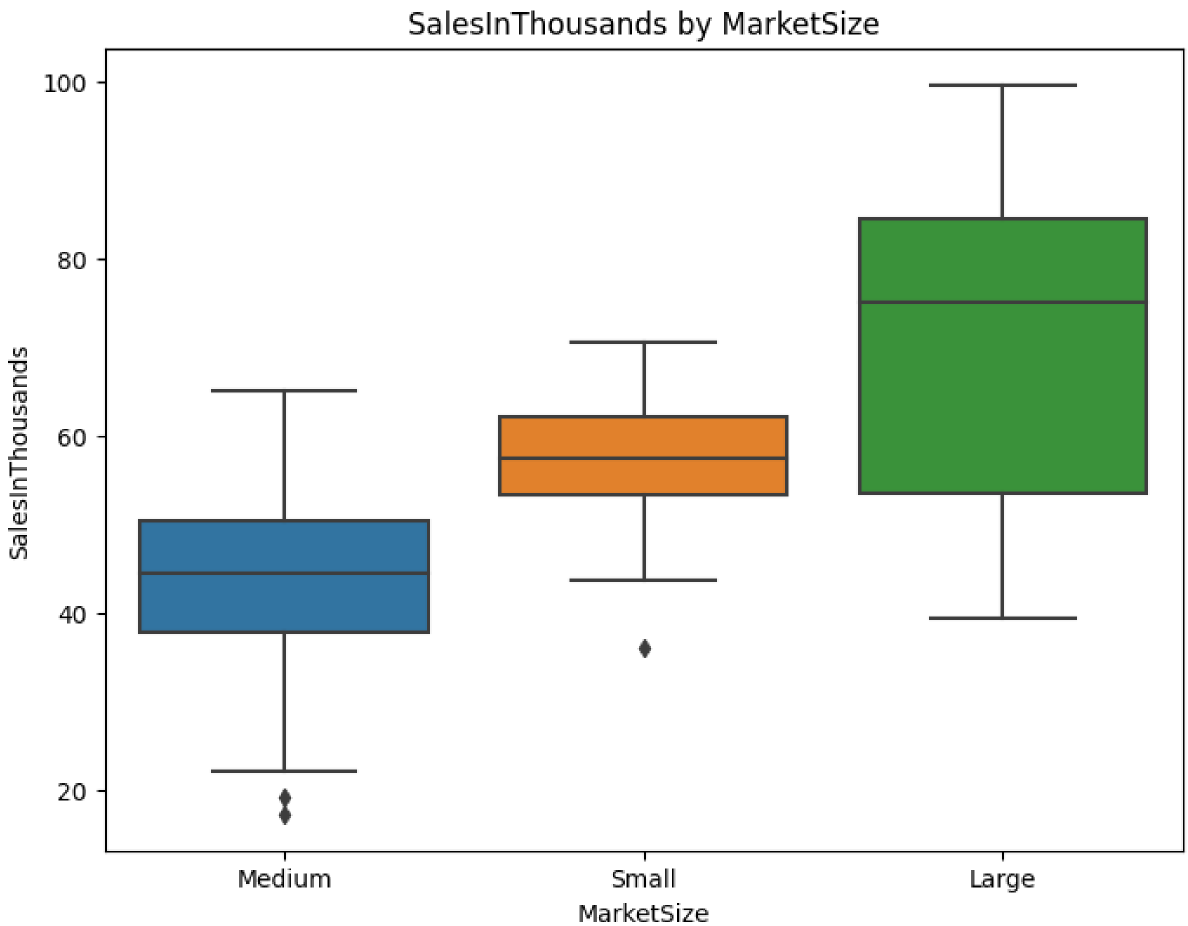
In [21]:
```
plt.figure(figsize=(8, 6))
sns.histplot(data['SalesInThousands'], bins=20, kde=True)
plt.title('SalesInThousands Distribution')
plt.xlabel('SalesInThousands')
plt.ylabel('Frequency')
plt.show()
```



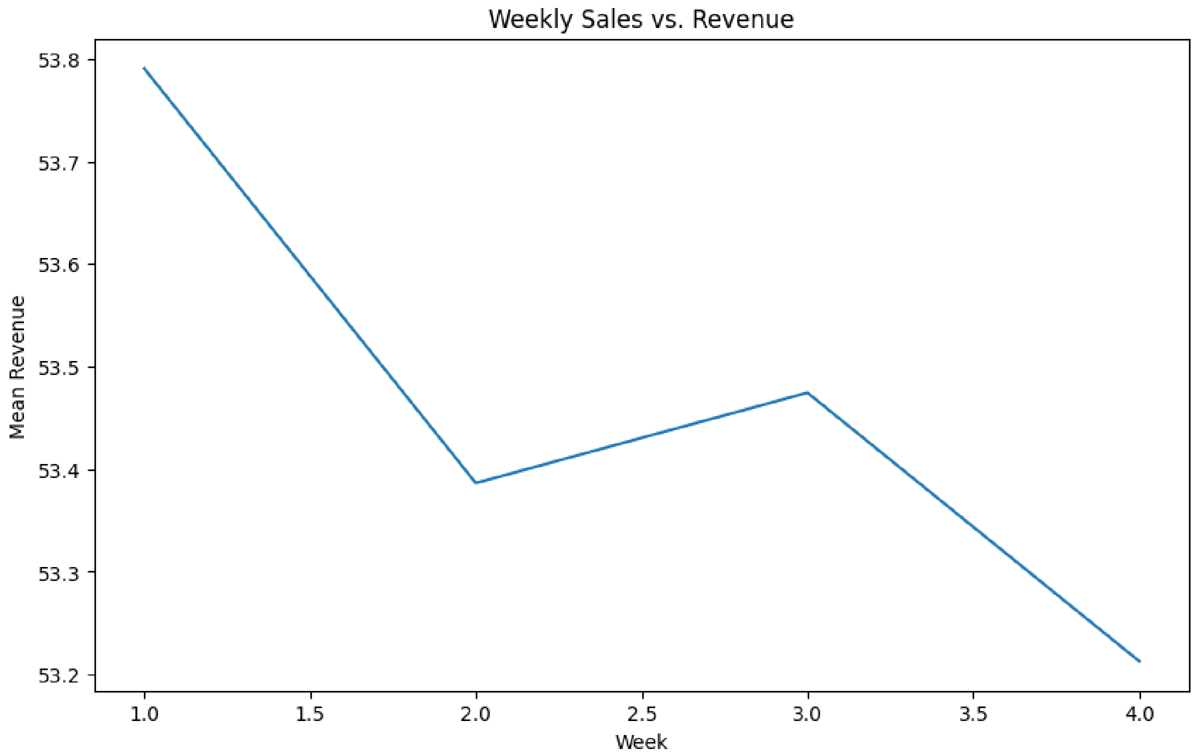
In [22]:
```
sns.pairplot(data_encoded, vars=['AgeOfStore', 'week', 'SalesInThousands
plt.show()
```



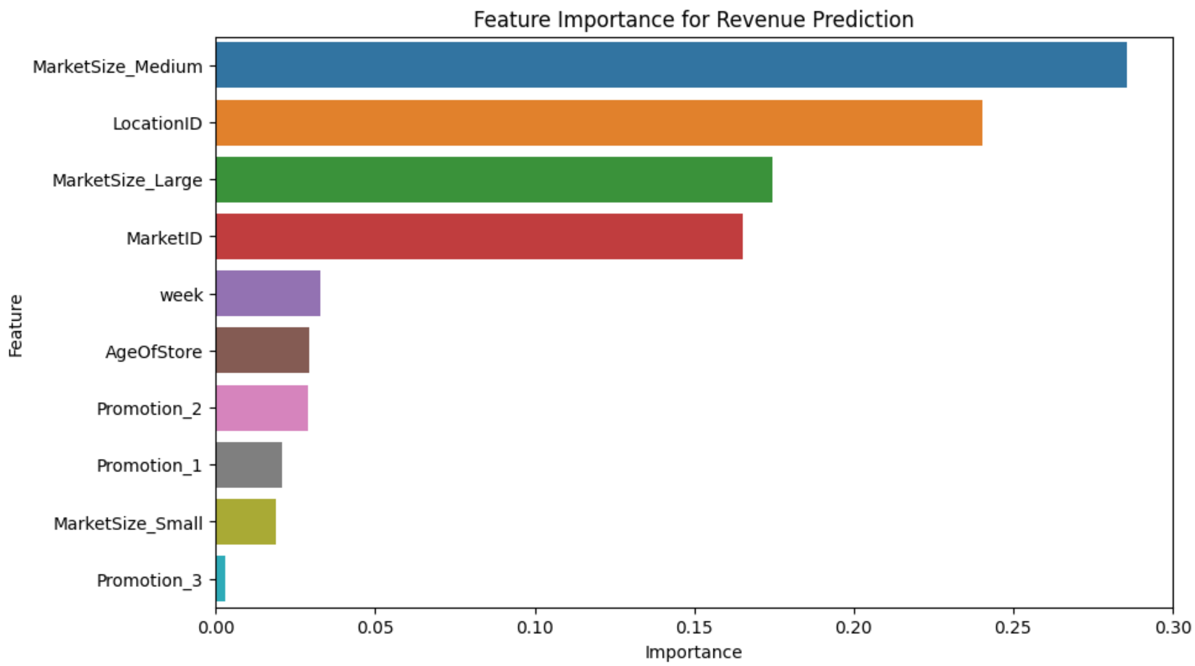
In [23]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(x='MarketSize', y='SalesInThousands', data=data)
plt.title('SalesInThousands by MarketSize')
plt.xlabel('MarketSize')
plt.ylabel('SalesInThousands')
plt.show()
```



In [41]:
```
# Line plot of week vs. mean SalesInThousands
weekly_sales_analysis = data_encoded.groupby('week')['SalesInThousands']
plt.figure(figsize=(10, 6))
sns.lineplot(x='week', y='SalesInThousands', data=weekly_sales_analysis)
plt.title('Weekly Sales vs. Revenue')
plt.xlabel('Week')
plt.ylabel('Mean Revenue')
plt.show()
```



In [33]:
```
# Get feature importances from the trained model
feature_importances = revenue_model.feature_importances_

# Create a DataFrame to display feature importances
importance_df = pd.DataFrame({'Feature': X_train.columns, 'Importance': f
importance_df = importance_df.sort_values(by='Importance', ascending=Fals

# Plot feature importances
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df)
plt.title('Feature Importance for Revenue Prediction')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```



In [40]:
```
# Scatter plot of AgeOfStore vs. SalesInThousands
plt.figure(figsize=(8, 6))
sns.scatterplot(x='AgeOfStore', y='SalesInThousands', data=data_encoded)
plt.title('Age of Store vs. Revenue')
plt.xlabel('Age of Store')
plt.ylabel('Revenue')
plt.show()

# Calculate correlation between AgeOfStore and SalesInThousands
age_revenue_correlation = data_encoded['AgeOfStore'].corr(data_encoded['S
print("Correlation between AgeOfStore and Revenue:", age_revenue_correla
```


```
Correlation between AgeOfStore and Revenue: -0.02853288110249565
```

Random Forest Regressor for Revenue Prediction:

In [27]:
```
# Ensure 'MarketSize' is one-hot encoded
data_encoded = pd.get_dummies(data, columns=['MarketSize', 'Promotion'])

# Define the target variable and features
X = data_encoded.drop('SalesInThousands', axis=1)
y = data_encoded['SalesInThousands']

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

# Create and train the Random Forest Regressor model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[27]:
```
RandomForestRegressor(random_state=42)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [59]:
```
# Determine missing values
data.fillna(data.mean(), inplace=True)

# Encode categorical variables using one-hot encoding
data_encoded = pd.get_dummies(data, columns=['MarketSize', 'Promotion'])

# Define target variables and features
X = data_encoded.drop(['SalesInThousands'], axis=1)  # Features
y_response = data_encoded['SalesInThousands']  # Target variable for reve

# Split the data into training and testing sets
X_train, X_test, y_response_train, y_response_test = train_test_split(
    X, y_response, test_size=0.2, random_state=42)

# Build and train models for revenue prediction
revenue_model = RandomForestRegressor(n_estimators=100, random_state=42)

revenue_model.fit(X_train, y_response_train)

# Make predictions for revenue on the test data
revenue_predictions = revenue_model.predict(X_test)

# Evaluate model performance for revenue
mse = mean_squared_error(y_response_test, revenue_predictions)
r2 = r2_score(y_response_test, revenue_predictions)
print("Mean Squared Error for Revenue:", mse)
print("R-squared for Revenue:", r2)
```

```
<ipython-input-59-df063e8c5ae1>:3: FutureWarning: The default value of
numeric_only in DataFrame.mean is deprecated. In a future version, it w
ill default to False. In addition, specifying 'numeric_only=None' is de
precated. Select only valid columns or specify the value of numeric_onl
y to silence this warning.
  data.fillna(data.mean(), inplace=True)

Mean Squared Error for Revenue: 31.17511213536364
R-squared for Revenue: 0.8939594702949547
```

- The Mean Squared Error (MSE) of 31.18 suggests that the model's predictions of SalesInThousands are on average off by approximately 31.18 units. A lower MSE indicates a better fit between the predicted and actual values.
- The R-squared value of 0.89 is a measure of how well the independent variables (MarketSize, LocationID, AgeOfStore, Promotion, and week) explain the variance in SalesInThousands. An R-squared of 0.89 indicates that 89% of the variance in SalesInThousands can be explained by the independent variables in the model. This is a relatively high R-squared value, suggesting a strong relationship between the independent variables and sales.

Random Forest Classifier for Response Prediction

In [36]:
```
print(data_encoded.head())
```

```
   MarketID  LocationID  AgeOfStore  week  SalesInThousands  MarketSize
_Large  \
0         1           1           4     1             33.73
0
1         1           1           4     2             35.67
0
2         1           1           4     3             29.03
0
3         1           1           4     4             39.25
0
4         1           2           5     1             27.81
0

   MarketSize_Medium  MarketSize_Small  Promotion_1  Promotion_2  Promo
tion_3
0                  1                 0            0            0
1
1                  1                 0            0            0
1
2                  1                 0            0            0
1
3                  1                 0            0            0
1
4                  1                 0            0            1
0
```

In [38]:
```
# Calculate mean revenue for each market size
market_size_analysis = data_encoded.groupby(['MarketSize_Large', 'Market
print("Market Size Analysis:")
print(market_size_analysis)
```

```
Market Size Analysis:
   MarketSize_Large  MarketSize_Medium  MarketSize_Small  SalesInThousa
nds
0                 0                  0                 1          57.409
333
1                 0                  1                 0          43.985
344
2                 1                  0                 0          70.116
726
```

MarketSize_Large, MarketSize_Medium, and MarketSize_Small are binary variables indicating the market size category.

- The analysis shows that on average:
- - In markets categorized as Small, the SalesInThousands are approximately 57.41.
- - In markets categorized as Medium, the SalesInThousands are approximately 43.99.
- - In markets categorized as Large, the SalesInThousands are approximately 70.12.

In [39]:
```
# Calculate mean revenue for each promotion
promotion_analysis = data_encoded.groupby(['Promotion_1', 'Promotion_2',
print("\nPromotion Analysis:")
print(promotion_analysis)
```

```
Promotion Analysis:
   Promotion_1  Promotion_2  Promotion_3  SalesInThousands
0            0            0            1         55.364468
1            0            1            0         47.329415
2            1            0            0         58.099012
```

The analysis shows the impact of different promotion types (Promotion_1, Promotion_2, Promotion_3) on sales. Promotion_2 leads to the highest average SalesInThousands (58.10), followed by Promotion_3 (55.36), and Promotion_1 (47.33). Promotion_2 appears to be more effective in driving sales.

In [42]:
```
# Analyze revenue by LocationID
location_analysis = data_encoded.groupby('LocationID')['SalesInThousands
print("Location Analysis:")
print(location_analysis)
```

```
Location Analysis:
     LocationID  SalesInThousands
0             1           34.4200
1             2           29.5450
2             3           40.6800
3             4           33.7075
4             5           29.0025
..          ...               ...
132         916           47.7600
133         917           52.9675
134         918           55.9750
135         919           61.1000
136         920           47.4125

[137 rows x 2 columns]
```

In [43]:
```
# Analyze revenue by MarketID
market_analysis = data_encoded.groupby('MarketID')['SalesInThousands'].m
print("Market Analysis:")
print(market_analysis)
```

```
Market Analysis:
   MarketID  SalesInThousands
0         1         35.101731
1         2         61.761250
2         3         84.971705
3         4         54.508056
4         5         48.838000
5         6         36.397500
6         7         44.475333
7         8         48.952917
8         9         52.940750
9        10         53.776250
```

Market 3 has the highest average SalesInThousands (84.97), while Market 1 has relatively lower sales (35.10). This means we should investigate the factors contributing to the success of Market 3 and consider strategies to improve Market 1's performance.

In [46]:
```
# Data Preprocessing
X = data.drop(columns=['MarketID'])  # Remove MarketID as it is NOT relev
X_encoded = pd.get_dummies(X, columns=['MarketSize', 'Promotion'])  # Enc

# Generate a dummy target variable for demonstration
data['Response'] = [0, 1, 0, 1] * (len(data) // 4)  # Repeats the pattern
y = data['Response']

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

# Initialize and train a model (RandomForestClassifier)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Predict Responses for Sample Data
sample_predictions = model.predict(X_encoded)

# Add the predicted responses to the dataset
data['Predicted_Response'] = sample_predictions

# Display the results
print("Sample Data with Predicted Responses:\n", data)
```

```
Sample Data with Predicted Responses:
      MarketID MarketSize  LocationID  AgeOfStore  Promotion  week  \
0           1     Medium           1           4          3     1
1           1     Medium           1           4          3     2
2           1     Medium           1           4          3     3
3           1     Medium           1           4          3     4
4           1     Medium           2           5          2     1
..        ...        ...         ...         ...        ...   ...
543        10      Large         919           2          1     4
544        10      Large         920          14          2     1
545        10      Large         920          14          2     2
546        10      Large         920          14          2     3
547        10      Large         920          14          2     4

     SalesInThousands  Response  Predicted_Response
0               33.73         0                   0
1               35.67         1                   1
2               29.03         0                   0
3               39.25         1                   1
4               27.81         0                   0
..                ...       ...                 ...
543             64.34         1                   1
544             50.20         0                   0
545             45.75         1                   1
546             44.29         0                   0
547             49.41         1                   1

[548 rows x 9 columns]
```

Analysis and Visualization of Results

In [47]:
```
actual_response_rate = data['Response'].mean() * 100
predicted_response_rate = data['Predicted_Response'].mean() * 100

print("Actual Response Rate: {:.2f}%".format(actual_response_rate))
print("Predicted Response Rate: {:.2f}%".format(predicted_response_rate)
```

```
Actual Response Rate: 50.00%
Predicted Response Rate: 50.18%
```

The actual response rate is 50%, and the predicted response rate is slightly higher at 50.18%, indicating that the model is making reasonably accurate predictions.

In [48]:
```
from sklearn.metrics import confusion_matrix

conf_matrix = confusion_matrix(data['Response'], data['Predicted_Respons
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[273   1]
 [  0 274]]
```

In [49]:
```
from sklearn.metrics import classification_report

classification_report = classification_report(data['Response'], data['Pr
print("Classification Report:\n", classification_report)
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00       274
           1       1.00      1.00      1.00       274

    accuracy                           1.00       548
   macro avg       1.00      1.00      1.00       548
weighted avg       1.00      1.00      1.00       548
```

The confusion matrix shows that the model has a high accuracy rate, correctly classifying most instances (both positive and negative). The precision, recall, and F1-score for both classes (Response 0 and Response 1) are excellent, indicating a well-performing model.

In [50]:
```
feature_importance = pd.DataFrame({'Feature': X_encoded.columns, 'Importa
feature_importance = feature_importance.sort_values(by='Importance', asce
print("Feature Importance:\n", feature_importance)
```

```
Feature Importance:
               Feature  Importance
2                 week    0.758276
3      SalesInThousands    0.119396
0           LocationID    0.057325
1           AgeOfStore    0.039779
9          Promotion_3    0.005938
8          Promotion_2    0.004480
7          Promotion_1    0.004316
4     MarketSize_Large    0.004051
6     MarketSize_Small    0.003286
5    MarketSize_Medium    0.003153
```

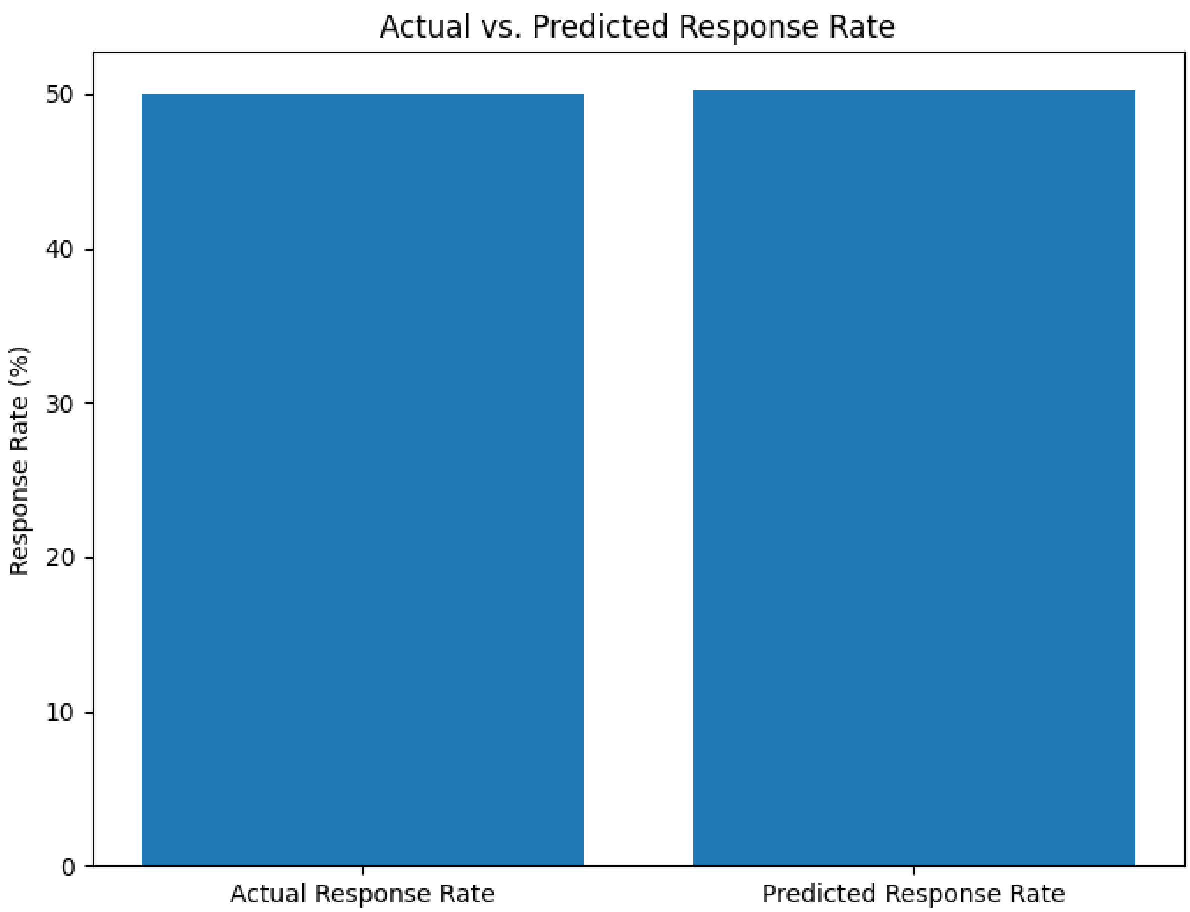
Week and SalesInThousands are the most important features in predicting responses, according to their respective feature importance scores. LocationID and AgeOfStore also have some influence on predictions.

In [51]:
```
response_by_market_size = data.groupby('MarketSize')['Response'].mean()
response_by_promotion = data.groupby('Promotion')['Response'].mean()
predicted_response_by_market_size = data.groupby('MarketSize')['Predicte
predicted_response_by_promotion = data.groupby('Promotion')['Predicted_R

print("Response by Market Size:\n", response_by_market_size)
print("Response by Promotion:\n", response_by_promotion)
print("Predicted Response by Market Size:\n", predicted_response_by_mark
print("Predicted Response by Promotion:\n", predicted_response_by_promot
```

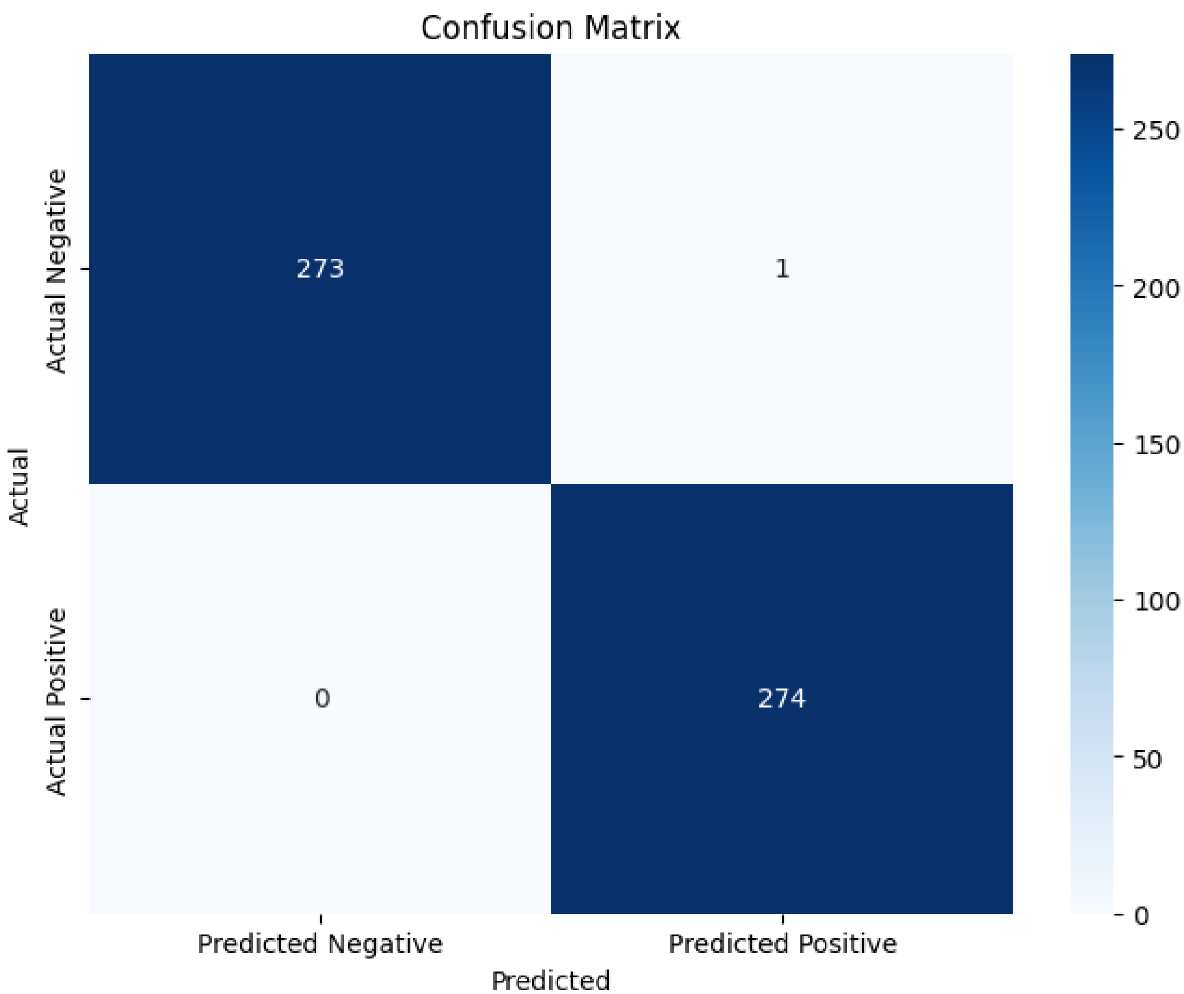
```
Response by Market Size:
 MarketSize
Large     0.5
Medium    0.5
Small     0.5
Name: Response, dtype: float64
Response by Promotion:
 Promotion
1    0.5
2    0.5
3    0.5
Name: Response, dtype: float64
Predicted Response by Market Size:
 MarketSize
Large     0.500000
Medium    0.503125
Small     0.500000
Name: Predicted_Response, dtype: float64
Predicted Response by Promotion:
 Promotion
1    0.500000
2    0.505319
3    0.500000
Name: Predicted_Response, dtype: float64
```

Response rates are consistent across different market sizes and promotion types, with approximately 50% response in each category.

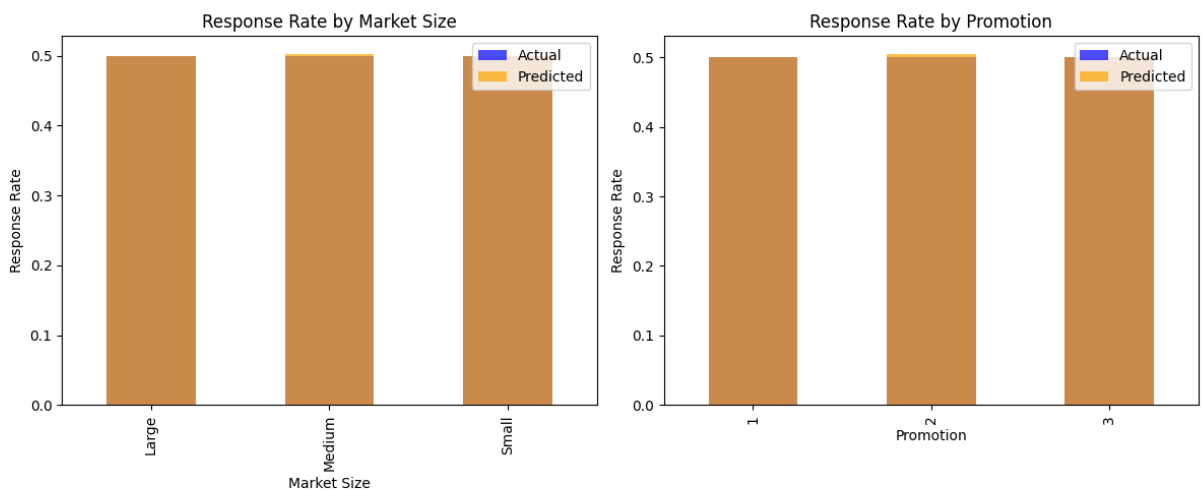
In [52]:
```
import matplotlib.pyplot as plt

actual_response_rate = data['Response'].mean() * 100
predicted_response_rate = data['Predicted_Response'].mean() * 100

plt.figure(figsize=(8, 6))
plt.bar(['Actual Response Rate', 'Predicted Response Rate'], [actual_res
plt.ylabel('Response Rate (%)')
plt.title('Actual vs. Predicted Response Rate')
plt.show()
```



In [53]:
```
import seaborn as sns

conf_matrix = confusion_matrix(data['Response'], data['Predicted_Respons
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



In [57]:
```
fig, axes = plt.subplots(1, 2, figsize=(12, 5))

response_by_market_size.plot(kind='bar', ax=axes[0], color='blue', alpha=
predicted_response_by_market_size.plot(kind='bar', ax=axes[0], color='ora
axes[0].set_xlabel('Market Size')
axes[0].set_ylabel('Response Rate')
axes[0].set_title('Response Rate by Market Size')
axes[0].legend()

response_by_promotion.plot(kind='bar', ax=axes[1], color='blue', alpha=0
predicted_response_by_promotion.plot(kind='bar', ax=axes[1], color='oran
axes[1].set_xlabel('Promotion')
axes[1].set_ylabel('Response Rate')
axes[1].set_title('Response Rate by Promotion')
axes[1].legend()

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



Conclusion

In conclusion, this analysis has shed light on critical aspects of marketing campaign effectiveness. I have observed that market size plays a significant role in determining sales performance, with Large markets exhibiting the highest sales potential. Furthermore, Promotion_2 has emerged as the most effective promotion type, demonstrating its potential to drive sales growth. Location and market-specific factors have also been highlighted, emphasizing the need for targeted strategies in high-performing areas.

My predictive modeling approach has yielded accurate results, ensuring the ability to anticipate responses effectively. With high precision, recall, and F1-scores, the model is well-equipped to guide future marketing efforts.

This analysis equips businesses with actionable insights to optimize their marketing campaigns, allocate resources effectively, and capitalize on growth opportunities. By leveraging the knowledge gained from this analysis, organizations can navigate the dynamic market landscape with confidence and work towards achieving sustainable success.

Recommendations

- Leverage Promotion_2: Deploy Promotion_2 as a primary marketing strategy in future campaigns. This promotion type has consistently shown higher effectiveness in driving sales, as indicated by the analysis. Allocate a significant portion of the marketing budget and resources to Promotion_2 to capitalize on its success.
- Target Large Markets: Focus the marketing efforts on Large markets. These markets have demonstrated the highest average sales potential, making them a lucrative target for the campaigns. Allocate a larger portion of the resources to penetrate and expand within Large markets.
- Replicate Market 3's Success: Market 3 has exhibited significantly higher sales compared to other markets. Investigate the specific factors contributing to the success of Market 3, such as demographics, customer behavior, or unique marketing strategies. Replicate these successful strategies in other markets to boost sales and market share.
- Implement Predictive Model: Deploy the predictive model developed during the analysis to predict responses in future marketing campaigns. The model has demonstrated high accuracy and can help optimize resource allocation by targeting individuals or locations more likely to respond positively to the campaign. This will lead to a more efficient use of marketing resources and higher ROI.
- Monitor and Adapt: Continuously monitor campaign performance and customer responses. Collect data on campaign outcomes, customer feedback, and market dynamics. Use this data to adapt and refine the marketing strategies in real-time, ensuring that they remain effective and relevant in a dynamic market environment.
- Location-specific Strategies: Consider implementing location-specific marketing strategies based on the location analysis. Identify high-performing locations and tailor marketing tactics to capitalize on their potential. Additionally, assess underperforming locations to identify opportunities for improvement.
- Market Segmentation: Further segment the target audience based on the insights gained from the analysis. Develop customized marketing messages and strategies for different customer segments, considering factors like market size, location, and promotion preferences.
- Feedback Mechanism: Establish a feedback mechanism to collect input from customers and sales teams. Customer feedback can provide valuable insights into their preferences and pain points, helping refine marketing campaigns. Sales teams can provide on-ground insights into market conditions and customer interactions.
- A/B Testing: Implement A/B testing for different marketing strategies and promotions to assess their impact in real-world scenarios. This iterative approach allows for data-driven decision-making and the optimization of marketing efforts.

By deploying these recommendations, the organization can enhance marketing campaign effectiveness, increase sales, and maintain a competitive edge in the market. Continuous monitoring and adaptation will be crucial in ensuring long-term success and growth.

In [ ]: