In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

In [2]:
```
# Load dataset
df = pd.read_csv('Restaurant_revenue.csv')
```

In [3]:
```
# Display the first few rows of the dataset
print(df.head())
```

```
   Number_of_Customers  Menu_Price  Marketing_Spend Cuisine_Type  \
0                   61   43.117635        12.663793     Japanese
1                   24   40.020077         4.577892      Italian
2                   81   41.981485         4.652911     Japanese
3                   70   43.005307         4.416053      Italian
4                   30   17.456199         3.475052      Italian

   Average_Customer_Spending  Promotions  Reviews  Monthly_Revenue
0                  36.236133           0       45       350.912040
1                  17.952562           0       36       221.319091
2                  22.600420           1       91       326.529763
3                  18.984098           1       59       348.190573
4                  12.766143           1       30       185.009121
```

In [4]:
```
print(df.isnull())
```

```
     Number_of_Customers  Menu_Price  Marketing_Spend  Cuisine_Type  \
0                  False       False            False         False
1                  False       False            False         False
2                  False       False            False         False
3                  False       False            False         False
4                  False       False            False         False
..                   ...         ...              ...           ...
995                False       False            False         False
996                False       False            False         False
997                False       False            False         False
998                False       False            False         False
999                False       False            False         False

     Average_Customer_Spending  Promotions  Reviews  Monthly_Revenue
0                        False       False    False            False
1                        False       False    False            False
2                        False       False    False            False
3                        False       False    False            False
4                        False       False    False            False
..                         ...         ...      ...              ...
995                      False       False    False            False
996                      False       False    False            False
997                      False       False    False            False
998                      False       False    False            False
999                      False       False    False            False

[1000 rows x 8 columns]
```

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
Number_of_Customers          0
Menu_Price                   0
Marketing_Spend              0
Cuisine_Type                 0
Average_Customer_Spending    0
Promotions                   0
Reviews                      0
Monthly_Revenue              0
dtype: int64
```

In [7]:
```
# Encode categorical variables
label_encoders = {}
for column in df.select_dtypes(include=['object']).columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    label_encoders[column] = le
```

In [8]:
```
# Preprocess the dataset
# Handle missing values if any
df.fillna(method='ffill', inplace=True)

X = df.drop('Monthly_Revenue', axis=1)
y = df['Monthly_Revenue']
```

In [9]:
```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
```

In [10]:
```
# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [11]:
```
# Train and evaluate Decision Tree model
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
y_pred_dt = dt_model.predict(X_test)
```

In [12]:
```
# Calculate evaluation metrics for Decision Tree
mae_dt = mean_absolute_error(y_test, y_pred_dt)
rmse_dt = np.sqrt(mean_squared_error(y_test, y_pred_dt))
print(f"Decision Tree MAE: {mae_dt}")
print(f"Decision Tree RMSE: {rmse_dt}")
```

```
Decision Tree MAE: 71.43678489215324
Decision Tree RMSE: 87.14453462032287
```

In [13]:
```
# Train and evaluate Nearest Neighbor model
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model.fit(X_train, y_train)
y_pred_knn = knn_model.predict(X_test)
```

In [14]:
```
# Calculate evaluation metrics for Nearest Neighbor
mae_knn = mean_absolute_error(y_test, y_pred_knn)
rmse_knn = np.sqrt(mean_squared_error(y_test, y_pred_knn))
print(f"Nearest Neighbor MAE: {mae_knn}")
print(f"Nearest Neighbor RMSE: {rmse_knn}")
```

```
Nearest Neighbor MAE: 55.27671957613795
Nearest Neighbor RMSE: 69.03173966742676
```

In [15]:
```
# Comparative analysis
print(f"Comparative Analysis:")
print(f"Decision Tree - MAE: {mae_dt}, RMSE: {rmse_dt}")
print(f"Nearest Neighbor - MAE: {mae_knn}, RMSE: {rmse_knn}")
```

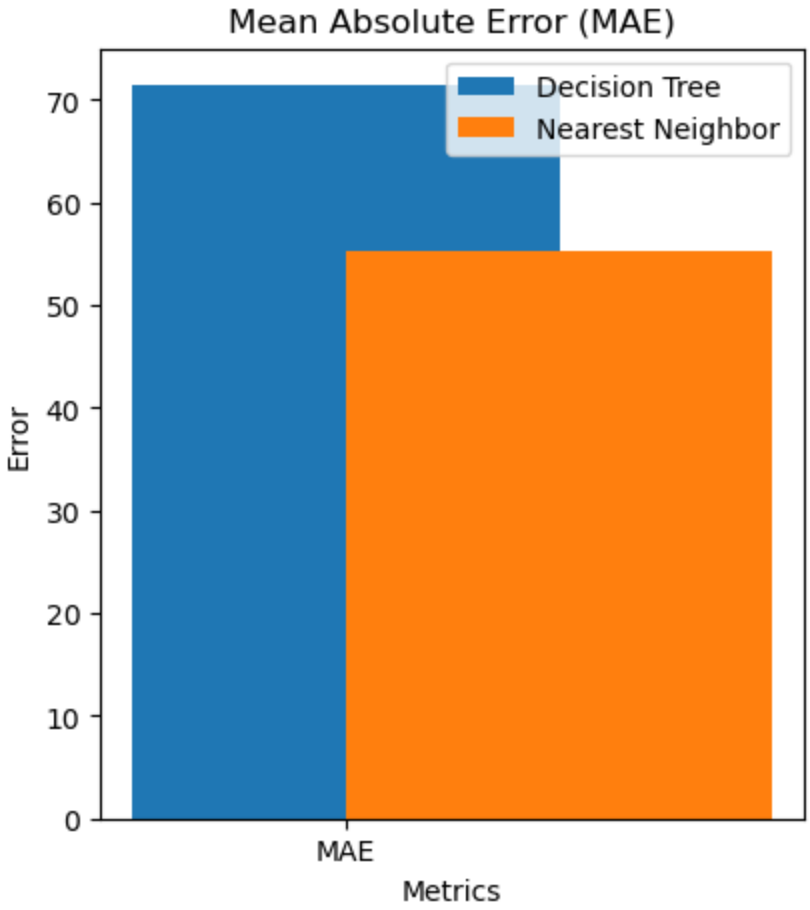
```
Comparative Analysis:
Decision Tree - MAE: 71.43678489215324, RMSE: 87.14453462032287
Nearest Neighbor - MAE: 55.27671957613795, RMSE: 69.03173966742676
```

In [16]:
```
# Performance metrics for Decision Tree
mae_dt = 71.43678489215324
rmse_dt = 87.14453462032287
```

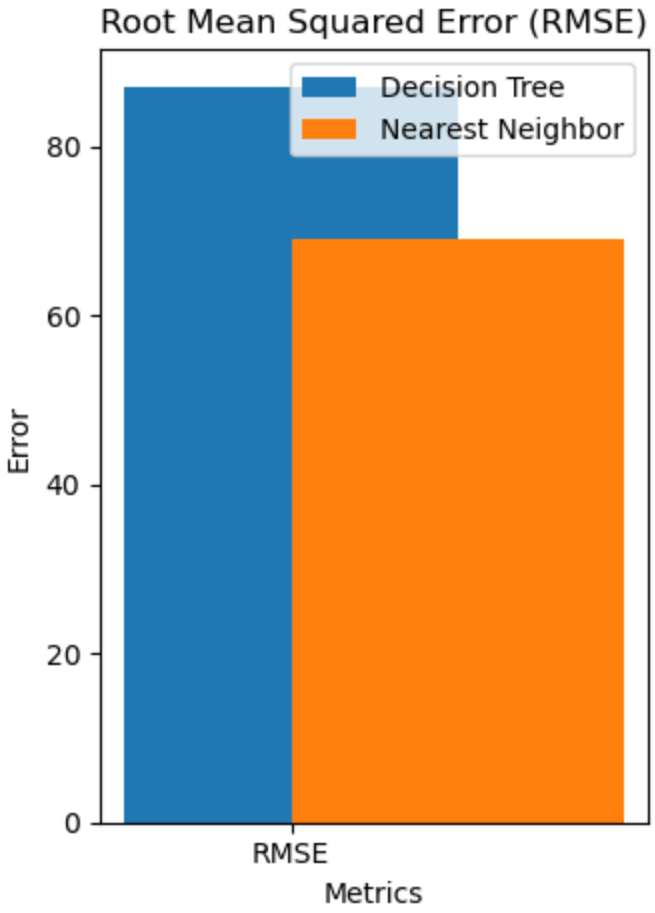
In [17]:
```
# Performance metrics for Nearest Neighbor
mae_knn = 55.27671957613795
rmse_knn = 69.03173966742676
```

In [18]:
```
import matplotlib.pyplot as plt

# Bar chart for MAE
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
metrics = ['MAE']
dt_values = [mae_dt]
knn_values = [mae_knn]
x = range(len(metrics))
plt.bar(x, dt_values, width=0.4, label='Decision Tree', align='center')
plt.bar(x, knn_values, width=0.4, label='Nearest Neighbor', align='edge')
plt.xlabel('Metrics')
plt.ylabel('Error')
plt.title('Mean Absolute Error (MAE)')
plt.xticks(x, metrics)
plt.legend()
```

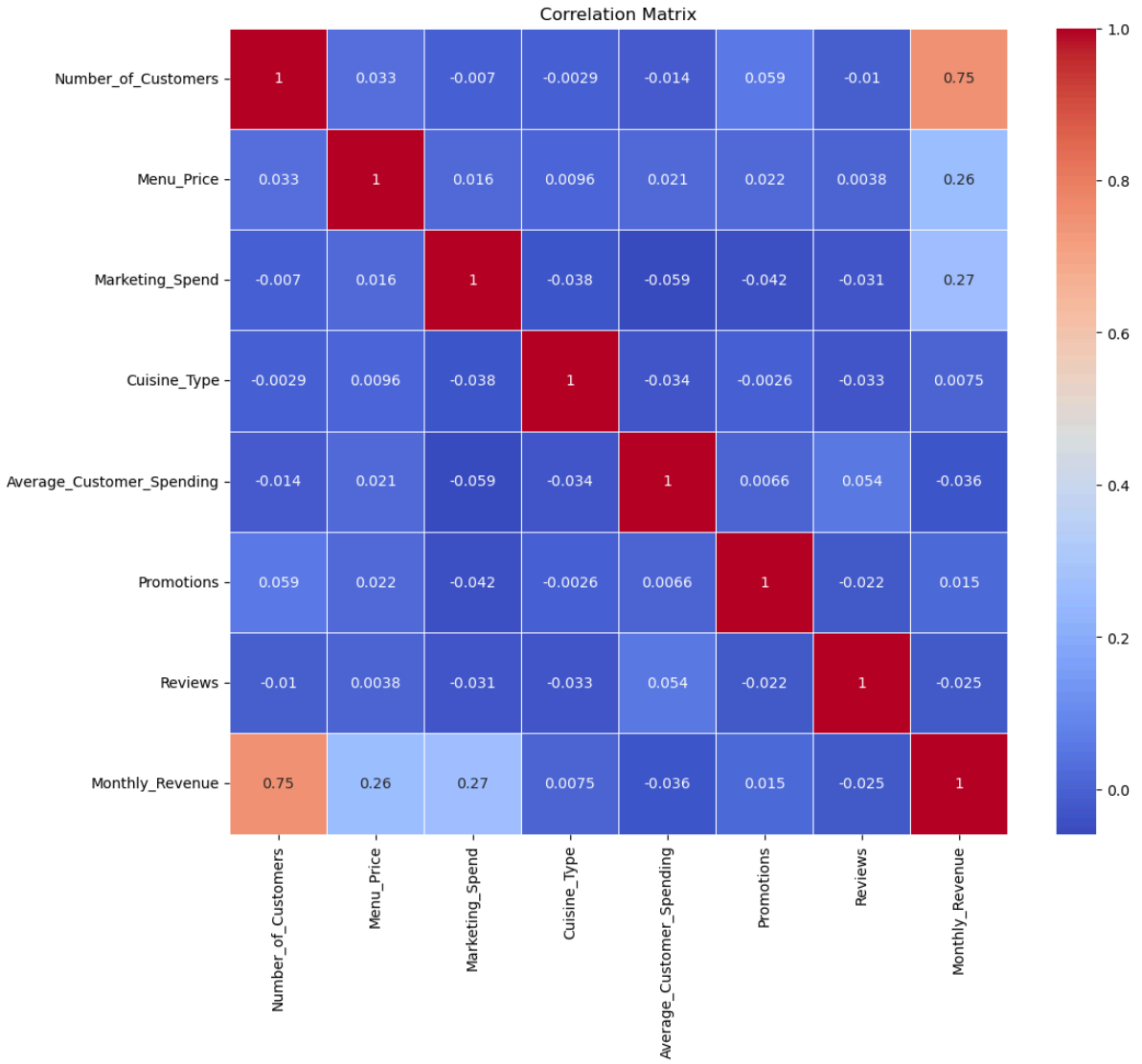
Out[18]:
```
<matplotlib.legend.Legend at 0x1b0eeb8e310>
```



In [19]:
```
# Bar chart for RMSE
plt.subplot(1, 2, 2)
metrics = ['RMSE']
dt_values = [rmse_dt]
knn_values = [rmse_knn]

x = range(len(metrics))
plt.bar(x, dt_values, width=0.4, label='Decision Tree', align='center')
plt.bar(x, knn_values, width=0.4, label='Nearest Neighbor', align='edge')
plt.xlabel('Metrics')
plt.ylabel('Error')
plt.title('Root Mean Squared Error (RMSE)')
plt.xticks(x, metrics)
plt.legend()

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



In [20]:
```
import seaborn as sns

# Draw correlation matrix
plt.figure(figsize=(12, 10))
correlation_matrix = df.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



In [21]:
```
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.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classifi
```

In [ ]: