```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score


# Load the dataset
df = pd.read_csv('100_Sales.csv')  # Change the filename according to the file you have
df.head()
```

	Region	Country	Item_Type	Sales_Channel	Order_Priority	Ship_Date	Unit_Cost	Total_Revenue	Total_Profit	Unnamed: 9	Unnamed: 10
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	27/06/2010	159.42	2533654.00	951410.50	NaN	NaN
1	Central America and the Caribbean	Grenada	Cereal	Online	C	15/09/2012	117.11	576782.80	248406.36	NaN	NaN
2	Europe	Russia	Office Supplies	Offline	L	05/08/2014	524.96	1158502.59	224598.75	NaN	NaN
3	Sub_Saharan Africa	Sao Tome and Principe	Fruits	Online	C	07/05/2014	6.92	75591.66	19525.82	NaN	NaN
	Sub_Saharan		Office								

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# Check for missing values
print("\nMissing Values Before Handling:\n", df.isnull().sum())
```

```
# Drop columns with all missing values
df.dropna(axis=1, how='all', inplace=True)

# Handle missing values for numerical columns, let's fill with the mean value
df['Unit_Cost'].fillna(df['Unit_Cost'].mean(), inplace=True)
df['Total_Profit'].fillna(df['Total_Profit'].mean(), inplace=True)

# After handling missing values, verify again
print("\nMissing Values After Handling:\n", df.isnull().sum())
```

```
Missing Values Before Handling:
 Region              0
Country             0
Item_Type           0
Sales_Channel       0
Order_Priority      0
Ship_Date           0
Unit_Cost           0
Total_Revenue       0
Total_Profit        0
Unnamed: 9        100
Unnamed: 10       100
dtype: int64

Missing Values After Handling:
 Region            0
Country           0
Item_Type         0
Sales_Channel     0
Order_Priority    0
Ship_Date         0
Unit_Cost         0
Total_Revenue     0
Total_Profit      0
dtype: int64
<ipython-input-41-205c2df17bf5>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Unit_Cost'].fillna(df['Unit_Cost'].mean(), inplace=True)
```

```
<ipython-input-41-205c2df17bf5>:9: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df['Total_Profit'].fillna(df['Total_Profit'].mean(), inplace=True)
```

```
# Drop 'Unnamed' columns if they exist
df = df.loc[:, ~df.columns.str.contains('^Unnamed')]

# Display the dataset again after dropping unwanted columns
df.head()
```

	Region	Country	Item_Type	Sales_Channel	Order_Priority	Ship_Date	Unit_Cost	Total_Revenue	Total_Profit
0	Australia and Oceania	Tuvalu	Baby Food	Offline	H	27/06/2010	159.42	2533654.00	951410.50
1	Central America and the Caribbean	Grenada	Cereal	Online	C	15/09/2012	117.11	576782.80	248406.36
2	Europe	Russia	Office Supplies	Offline	L	05/08/2014	524.96	1158502.59	224598.75
3	Sub_Saharan Africa	Sao Tome and Principe	Fruits	Online	C	07/05/2014	6.92	75591.66	19525.82
4	Sub_Saharan Africa	Rwanda	Office Supplies	Offline	L	02/06/2013	524.96	3296425.02	639077.50

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
# One-hot encode categorical columns
df_encoded = pd.get_dummies(df, drop_first=True)

# Display the encoded dataset
df_encoded.head()
```

	Unit_Cost	Total_Revenue	Total_Profit	Region_Australia and Oceania	Region_Central America and the Caribbean	Region_Europe	Region_Middle East and North Africa	Region_North America	Region_Sub_Sahar Afri
0	159.42	2533654.00	951410.50	True	False	False	False	False	Fa
1	117.11	576782.80	248406.36	False	True	False	False	False	Fa
2	524.96	1158502.59	224598.75	False	False	True	False	False	Fa
3	6.92	75591.66	19525.82	False	False	False	False	False	Ti
4	524.96	3296425.02	639077.50	False	False	False	False	False	Ti

5 rows × 197 columns

```
# Define target variable (update the column name based on your dataset)
y = df_encoded['Total_Revenue']  # Target variable: Change as required
X = df_encoded.drop(columns=['Total_Revenue'])  # Features: Drop target column


# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Verify the split
print("X_train shape:", X_train.shape)
print("y_train shape:", y_train.shape)
print("X_test shape:", X_test.shape)
print("y_test shape:", y_test.shape)
```

```
X_train shape: (80, 196)
y_train shape: (80,)
X_test shape: (20, 196)
y_test shape: (20,)
```

```
# Initialize the Decision Tree Regressor model
dt_model = DecisionTreeRegressor(random_state=42)

# Train the model
```

```
dt_model.fit(X_train, y_train)

# Make predictions
y_pred_dt = dt_model.predict(X_test)

# Evaluate the model
print("Decision Tree - Mean Absolute Error:", mean_absolute_error(y_test, y_pred_dt))
print("Decision Tree - Mean Squared Error:", mean_squared_error(y_test, y_pred_dt))
print("Decision Tree - R2 Score:", r2_score(y_test, y_pred_dt))
```

```
Decision Tree - Mean Absolute Error: 450258.1825000001
Decision Tree - Mean Squared Error: 457717223225.0508
Decision Tree - R2 Score: 0.7932414302893978
```

```
# Initialize the Random Forest Regressor model
rf_model = RandomForestRegressor(random_state=42)

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

# Make predictions
y_pred_rf = rf_model.predict(X_test)

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

```
Random Forest - Mean Absolute Error: 373385.7374149999
Random Forest - Mean Squared Error: 350122119657.1751
Random Forest - R2 Score: 0.8418439485971261
```

```
# Compare models based on R2 Score or other metrics
print(f"Decision Tree R2 Score: {r2_score(y_test, y_pred_dt)}")
print(f"Random Forest R2 Score: {r2_score(y_test, y_pred_rf)}")
```

```
Decision Tree R2 Score: 0.7932414302893978
Random Forest R2 Score: 0.8418439485971261
```

Start coding or generate with AI.