```
!pip install pandas numpy matplotlib seaborn scikit-learn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (1.5.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.23.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.12.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.3.post1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.46.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->panda
```

```
# List the files in the extracted directory
extracted_files = os.listdir(extracted_dir)
extracted_files
```

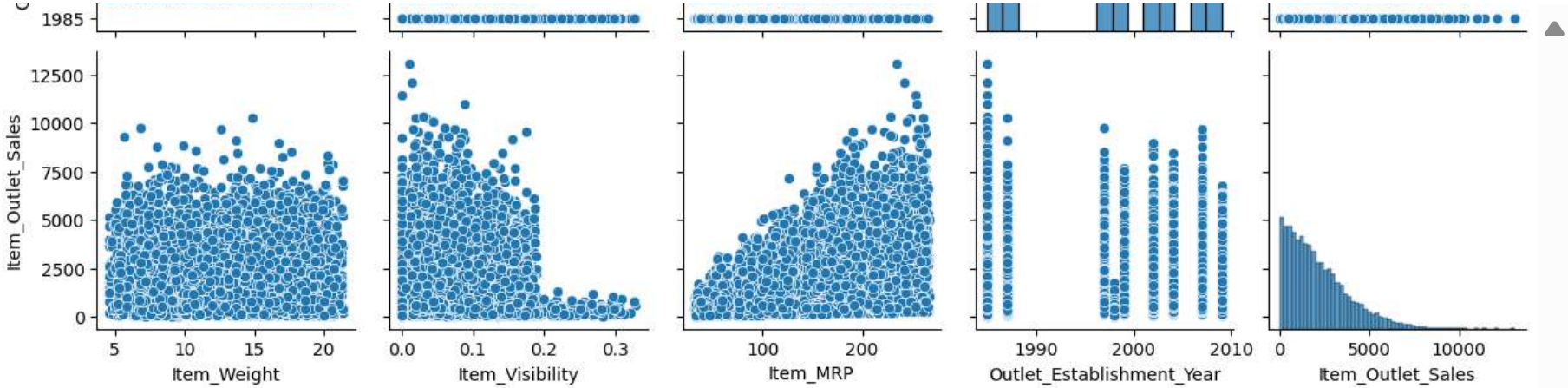
```
['Train.csv', 'Test.csv']
```

```
import zipfile
import os

# Define the path to the zip file
zip_file_path = '/content/9961_14084_bundle_archive.zip'

# Define the directory to extract to
extracted_dir = '/content/dataset'

# Create the directory if it doesn't exist
os.makedirs(extracted_dir, exist_ok=True)

# Extract the contents of the zip file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_dir)

# List the files in the extracted directory
extracted_files = os.listdir(extracted_dir)
extracted_files
```

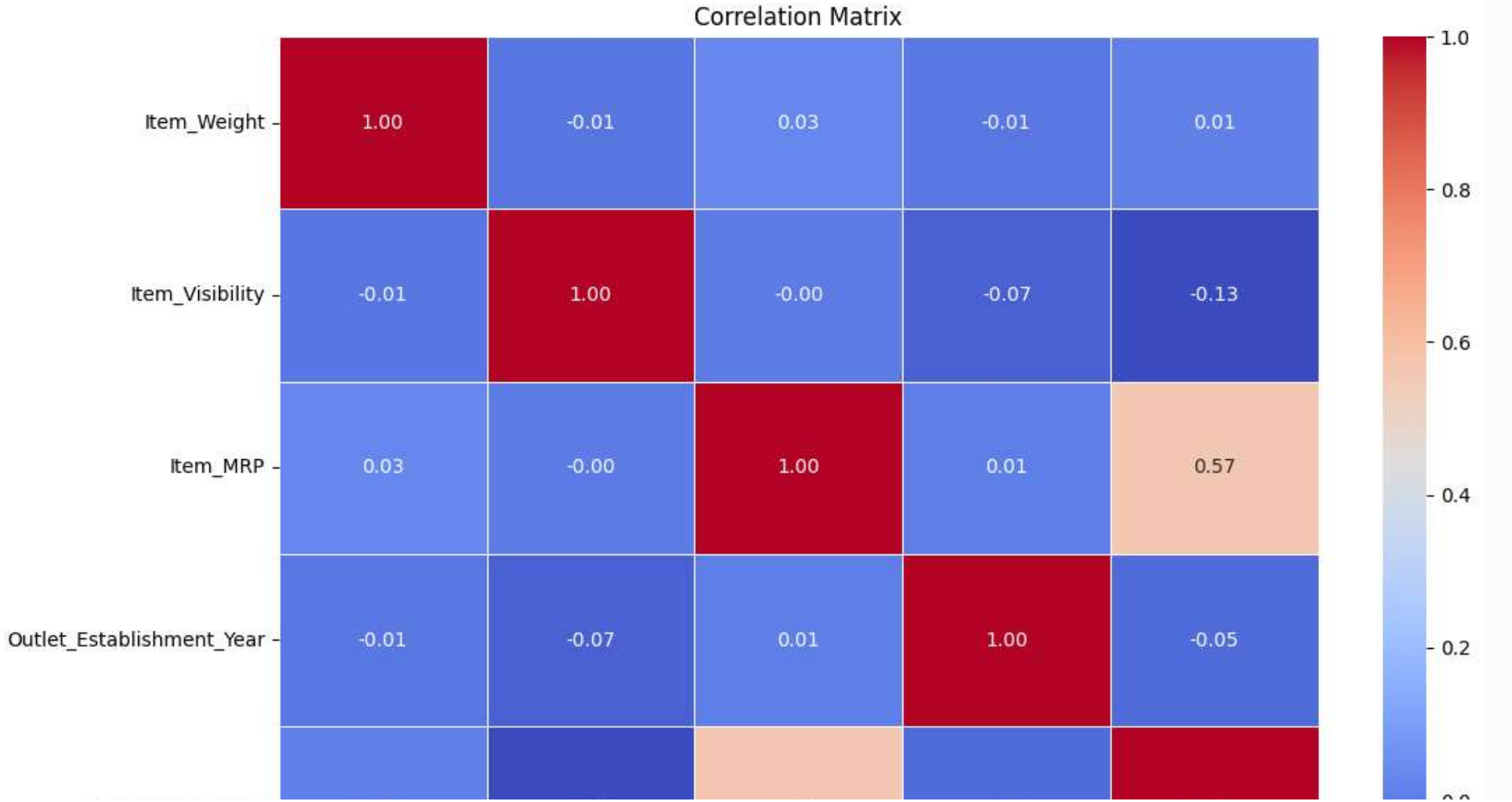
```
['Train.csv', 'Test.csv']
```

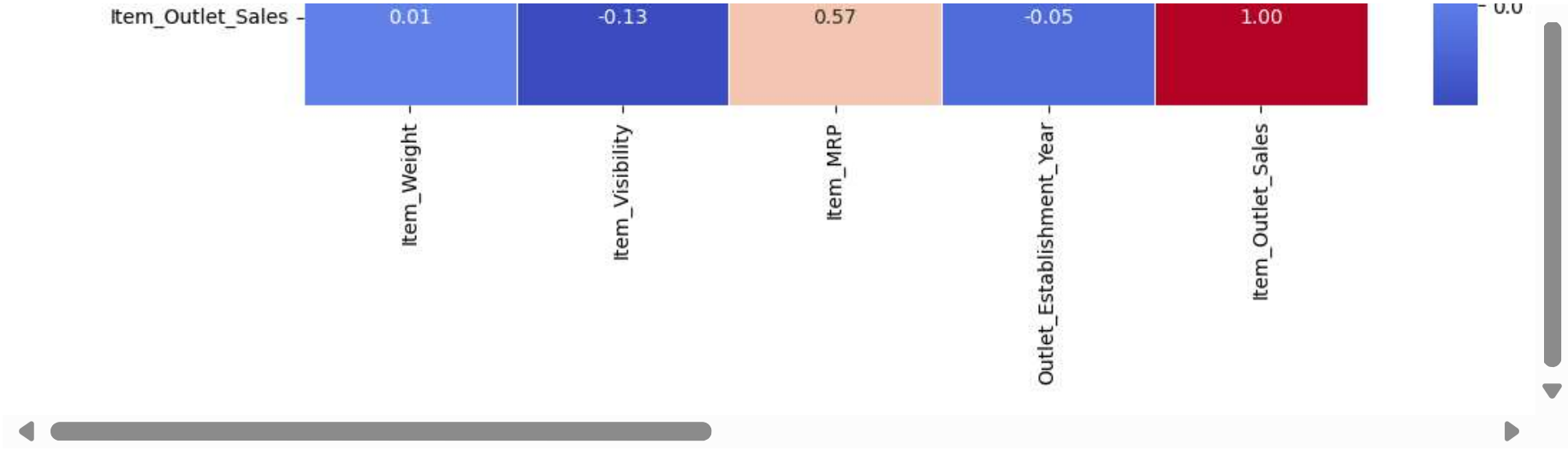
```
import zipfile
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Define the path to the zip file
zip_file_path = '/content/9961_14084_bundle_archive.zip'

# Define the directory to extract to
extracted_dir = '/content/dataset'

# Create the directory if it doesn't exist
os.makedirs(extracted_dir, exist_ok=True)

# Extract the contents of the zip file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_dir)

# List the files in the extracted directory
extracted_files = os.listdir(extracted_dir)
extracted_files

# Assuming the dataset is in CSV format
# Check the extracted files and identify the CSV file
csv_file = [file for file in extracted_files if file.lower().endswith('.csv')][0]
csv_file_path = os.path.join(extracted_dir, csv_file)

# Load the dataset into a Pandas DataFrame
df = pd.read_csv(csv_file_path)

# Display basic information about the dataset
print("Dataset Information:")
print(df.info())

# Summary statistics of numerical features
print("\nSummary Statistics:")
print(df.describe())
```

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

# Explore categorical variables
categorical_columns = df.select_dtypes(include=['object']).columns
if len(categorical_columns) > 0:
    print("\nCategorical Variables:")
    for column in categorical_columns:
        print(f"\n{column}:\n{df[column].value_counts()}")

# Visualize the distribution of numerical features
plt.figure(figsize=(12, 6))
df.select_dtypes(include=['int64', 'float64']).hist(bins=20, color='blue', edgecolor='black', grid=False)
plt.suptitle('Distribution of Numerical Features', x=0.5, y=1.02, ha='center', fontsize='x-large')
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

# Visualize relationships between variables using a pair plot
sns.pairplot(df.select_dtypes(include=['int64', 'float64']))
plt.suptitle('Pair Plot of Numerical Features', x=0.5, y=1.02, ha='center', fontsize='x-large')
plt.show()

# Visualize correlation matrix
correlation_matrix = df.corr()
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```




```
<ipython-input-12-00b78e2d179c>:65: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated
  correlation_matrix = df.corr()
```




Item_Outlet_Sales
0.01
-0.13
0.57
-0.05
1.00
Item_Weight
Item_Visibility
Item_MRP
Outlet_Establishment_Year
Item_Outlet_Sales
0.0

```
# Assuming 'df' is the DataFrame containing your dataset

# Step 2.1: Feature Engineering (if needed)
# Example: Extracting information from date columns, creating new features, etc.

# Step 2.2: Encode Categorical Variables
# Example: Using one-hot encoding for categorical columns
df = pd.get_dummies(df, columns=['Item_Weight', 'Item_MRP'], drop_first=True)

# Step 2.3: Handle Missing Values
# Example: Impute missing values for numerical columns with mean and for categorical columns with mode
from sklearn.impute import SimpleImputer

# Numerical columns
numerical_cols = df.select_dtypes(include=['int64', 'float64']).columns
numerical_imputer = SimpleImputer(strategy='mean')
df[numerical_cols] = numerical_imputer.fit_transform(df[numerical_cols])

# Categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
categorical_imputer = SimpleImputer(strategy='most_frequent')
df[categorical_cols] = categorical_imputer.fit_transform(df[categorical_cols])

# Step 2.4: Split the Data
from sklearn.model_selection import train_test_split

# Specify features and target variable
X = df.drop('Item_Outlet_Sales', axis=1)
y = df['Item_Outlet_Sales']

# 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, random_state=42)

# Display the first few rows of the preprocessed DataFrame
print("Preprocessed DataFrame:")
print(X_train.head())
```

```
Preprocessed DataFrame:
     Item_Identifier Item_Fat_Content  Item_Visibility              Item_Type  \
549            FDW44          Regular         0.035206  Fruits and Vegetables
7757           NCF54          Low Fat         0.047473              Household
764            FDY03          Regular         0.076122                   Meat
6867           FDQ20          Low Fat         0.029845  Fruits and Vegetables
2716           FDP34          Low Fat         0.137228            Snack Foods

     Outlet_Identifier  Outlet_Establishment_Year Outlet_Size  \
549             OUT049                     1999.0      Medium
7757            OUT045                     2002.0      Medium
764             OUT046                     1997.0       Small
6867            OUT045                     2002.0      Medium
2716            OUT046                     1997.0       Small

     Outlet_Location_Type        Outlet_Type  Item_Weight_4.59  ...  \
549                Tier 1  Supermarket Type1                 0  ...
7757               Tier 2  Supermarket Type1                 0  ...
764                Tier 1  Supermarket Type1                 0  ...
6867               Tier 2  Supermarket Type1                 0  ...
2716               Tier 1  Supermarket Type1                 0  ...

      Item_MRP_265.5568  Item_MRP_265.6884  Item_MRP_265.7884  \
549                   0                  0                  0
7757                  0                  0                  0
764                   0                  0                  0
6867                  0                  0                  0
2716                  0                  0                  0

      Item_MRP_265.8884  Item_MRP_266.0226  Item_MRP_266.1884  \
549                   0                  0                  0
7757                  0                  0                  0
764                   0                  0                  0
6867                  0                  0                  0
2716                  0                  0                  0

      Item_MRP_266.2884  Item_MRP_266.5884  Item_MRP_266.6884  \
549                   0                  0                  0
7757                  0                  0                  0
764                   0                  0                  0
6867                  0                  0                  0
2716                  0                  0                  0
```

```
      Item_MRP_266.8884
549                   0
7757                  0
764                   0
6867                  0
2716                  0

[5 rows x 6360 columns]
```

```
import zipfile
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
zip_file_path = '/content/9961_14084_bundle_archive.zip'
extracted_dir = '/content/dataset'
os.makedirs(extracted_dir, exist_ok=True)

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_dir)

# Identify the CSV file in the extracted directory
csv_file = [file for file in os.listdir(extracted_dir) if file.lower().endswith('.csv')][0]
csv_file_path = os.path.join(extracted_dir, csv_file)

# Load the dataset into a Pandas DataFrame
df = pd.read_csv(csv_file_path)

# Data Processing
# Assuming 'Item_Weight', 'Item_MRP', and 'Item_Outlet_Sales' are relevant columns
selected_columns = ['Item_Weight', 'Item_MRP', 'Item_Outlet_Sales']
df_model = df[selected_columns].dropna()

# Split the data into features (X) and target variable (y)
X = df_model[['Item_Weight', 'Item_MRP']]
y = df_model['Item_Outlet_Sales']

# 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, random_state=42)

# Build a Predictive Model
model = LinearRegression()
model.fit(X_train, y_train)

# Make predictions on the testing set
```

```
y_pred = model.predict(X_test)

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

print("Mean Squared Error:", mse)
print("R-squared (R2) Score:", r2)

# Display the coefficients of the model
print("Model Coefficients:")
print("Intercept:", model.intercept_)
print("Coefficients:", model.coef_)
```

```
Mean Squared Error: 1453287.2973690468
R-squared (R2) Score: 0.40218538459496067
Model Coefficients:
Intercept: 9.025218670723916
Coefficients: [-2.37575074 15.13098914]
```

```
import zipfile
import os
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
zip_file_path = '/content/9961_14084_bundle_archive.zip'
extracted_dir = '/content/dataset'
os.makedirs(extracted_dir, exist_ok=True)

with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extracted_dir)

# Identify the CSV file in the extracted directory
csv_file = [file for file in os.listdir(extracted_dir) if file.lower().endswith('.csv')][0]
csv_file_path = os.path.join(extracted_dir, csv_file)

# Load the dataset into a Pandas DataFrame
df = pd.read_csv(csv_file_path)

# Data Processing
# Assuming 'Item_Weight', 'Item_MRP', and 'Item_Outlet_Sales' are relevant columns
selected_columns = ['Item_Weight', 'Item_MRP', 'Item_Outlet_Sales']
df_model = df[selected_columns].dropna()

# Split the data into features (X) and target variable (y)
X = df_model[['Item_Weight', 'Item_MRP']]
y = df_model['Item_Outlet_Sales']

# 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, random_state=42)

# Build a Predictive Model (Linear Regression)
model = LinearRegression()

# Fine-Tuning and Optimization (GridSearchCV)
# LinearRegression does not have a 'normalize' parameter, so we skip this step
```

```
# Continue with training and evaluation
model.fit(X_train, y_train)

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

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

print("Model - Mean Squared Error:", mse)
print("Model - R-squared (R2) Score:", r2)
```

```
Model - Mean Squared Error: 1453287.2973690468
Model - R-squared (R2) Score: 0.40218538459496067
```