: INTERNSHIP PROJECT:

B. RAVI KUMAR
VU21CSEN0500333
Ph:9391350642

2.Problem statement: -

Sales Prediction Project.

The BigMart sales dataset is a treasure trove of learning opportunities. It consists of 2013 sales data for 1559 products across ten outlets in different cities. Your goal in this Ml. project is to build a regression model that can predict the sales of each of these 1559 products for the following year in each of the 10 different BigMart outlets. The dataset also includes specific attributes for each product and store, providing valuable insights into the factors influencing sales. This project is a fantastic way to understand how machine learning can help businesses like BigMart increase their sales.

CODE: -

```
# 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, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import joblib
# Load the datasets
```

```python
train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')

# Data Cleaning
# Fill missing values
train_data['Item_Weight'].fillna(train_data['Item_Weight'].mean(), inplace=True)
train_data['Outlet_Size'].fillna(train_data['Outlet_Size'].mode()[0], inplace=True)
test_data['Item_Weight'].fillna(test_data['Item_Weight'].mean(), inplace=True)
test_data['Outlet_Size'].fillna(test_data['Outlet_Size'].mode()[0], inplace=True)

# Clean Item_Fat_Content
train_data['Item_Fat_Content'] = train_data['Item_Fat_Content'].replace(
    {'LF': 'Low Fat', 'low fat': 'Low Fat', 'reg': 'Regular'})
test_data['Item_Fat_Content'] = test_data['Item_Fat_Content'].replace(
    {'LF': 'Low Fat', 'low fat': 'Low Fat', 'reg': 'Regular'})

# Feature Engineering
train_data['Outlet_Age'] = 2024 - train_data['Outlet_Establishment_Year']
test_data['Outlet_Age'] = 2024 - test_data['Outlet_Establishment_Year']



# Exploratory Data Analysis (EDA)
plt.figure(figsize=(10, 6))
sns.histplot(train_data['Item_Outlet_Sales'], bins=30, kde=True)
plt.title('Distribution of Item Outlet Sales')
plt.xlabel('Item Outlet Sales')
```

```
plt.ylabel('Frequency')
plt.show()


plt.figure(figsize=(10, 6))
sns.histplot(train_data['Item_MRP'], bins=30, kde=True)
plt.title('Distribution of Item MRP')
plt.xlabel('Item MRP')
plt.ylabel('Frequency')
plt.show()


plt.figure(figsize=(12, 6))
sns.countplot(data=train_data, y='Item_Type',
order=train_data['Item_Type'].value_counts().index)
plt.title('Count of Items by Item Type')
plt.xlabel('Count')
plt.ylabel('Item Type')
plt.show()



plt.figure(figsize=(10, 6))
sns.scatterplot(data=train_data, x='Item_Visibility', y='Item_Outlet_Sales', hue='Item_Type',
alpha=0.7)
plt.title('Item Visibility vs. Item Outlet Sales')
plt.xlabel('Item Visibility')
plt.ylabel('Item Outlet Sales')
plt.show()
```

```
plt.figure(figsize=(10, 6))

sns.boxplot(data=train_data, x='Outlet_Type', y='Item_Outlet_Sales')

plt.title('Sales Distribution by Outlet Type')

plt.xlabel('Outlet Type')

plt.ylabel('Item Outlet Sales')

plt.show()

# Select features and target variable

X = train_data.drop(columns=['Item_Identifier', 'Outlet_Identifier', 'Item_Outlet_Sales'])

y = train_data['Item_Outlet_Sales']

X_test = test_data.drop(columns=['Item_Identifier', 'Outlet_Identifier'])

# Define column transformer

numeric_features = ['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Outlet_Age']

numeric_transformer = Pipeline(steps=[

    ('imputer', SimpleImputer(strategy='mean')),

  ('scaler', StandardScaler())

])


categorical_features = ['Item_Fat_Content', 'Item_Type', 'Outlet_Size', 'Outlet_Location_Type',
'Outlet_Type']

categorical_transformer = Pipeline(steps=[

    ('imputer', SimpleImputer(strategy='most_frequent')),

    ('onehot', OneHotEncoder(handle_unknown='ignore'))

])


preprocessor = ColumnTransformer(

    transformers=[

        ('num', numeric_transformer, numeric_features),
```

```
        ('cat', categorical_transformer, categorical_features)
    ])

# Define the model
model = Pipeline(steps=[('preprocessor', preprocessor),
                        ('regressor', RandomForestRegressor(n_estimators=100, random_state=42))])

# Split the data
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Make predictions
y_pred = model.predict(X_val)

# Evaluate the model
mae = mean_absolute_error(y_val, y_pred)
mse = mean_squared_error(y_val, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_val, y_pred)

print(f'Mean Absolute Error: {mae}')
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
print(f'R^2 Score: {r2}')
```

```
# Make predictions on the test set
test_predictions = model.predict(X_test)

# Prepare the submission file
submission = pd.DataFrame({'Item_Identifier': test_data['Item_Identifier'],
                           'Outlet_Identifier': test_data['Outlet_Identifier'],
                           'Item_Outlet_Sales': test_predictions})

submission.to_csv('submission.csv', index=False)

# Save the model
joblib.dump(model, 'sales_prediction_model.pkl')
print('Model training complete and submission file created.')
```

OUTPUT: -

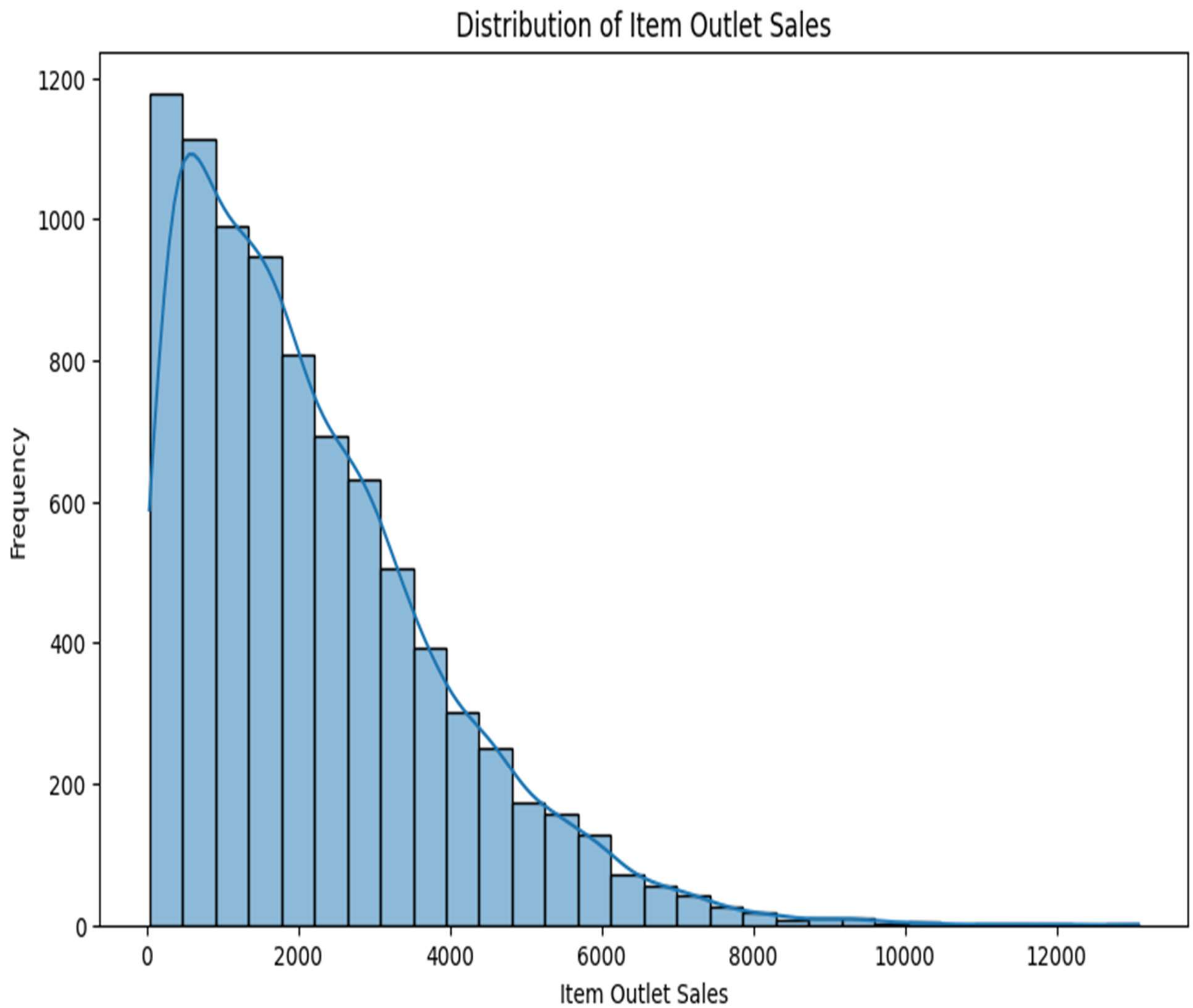
Mean Absolute Error: 762.6070544609971

Mean Squared Error: 1193925.618953065

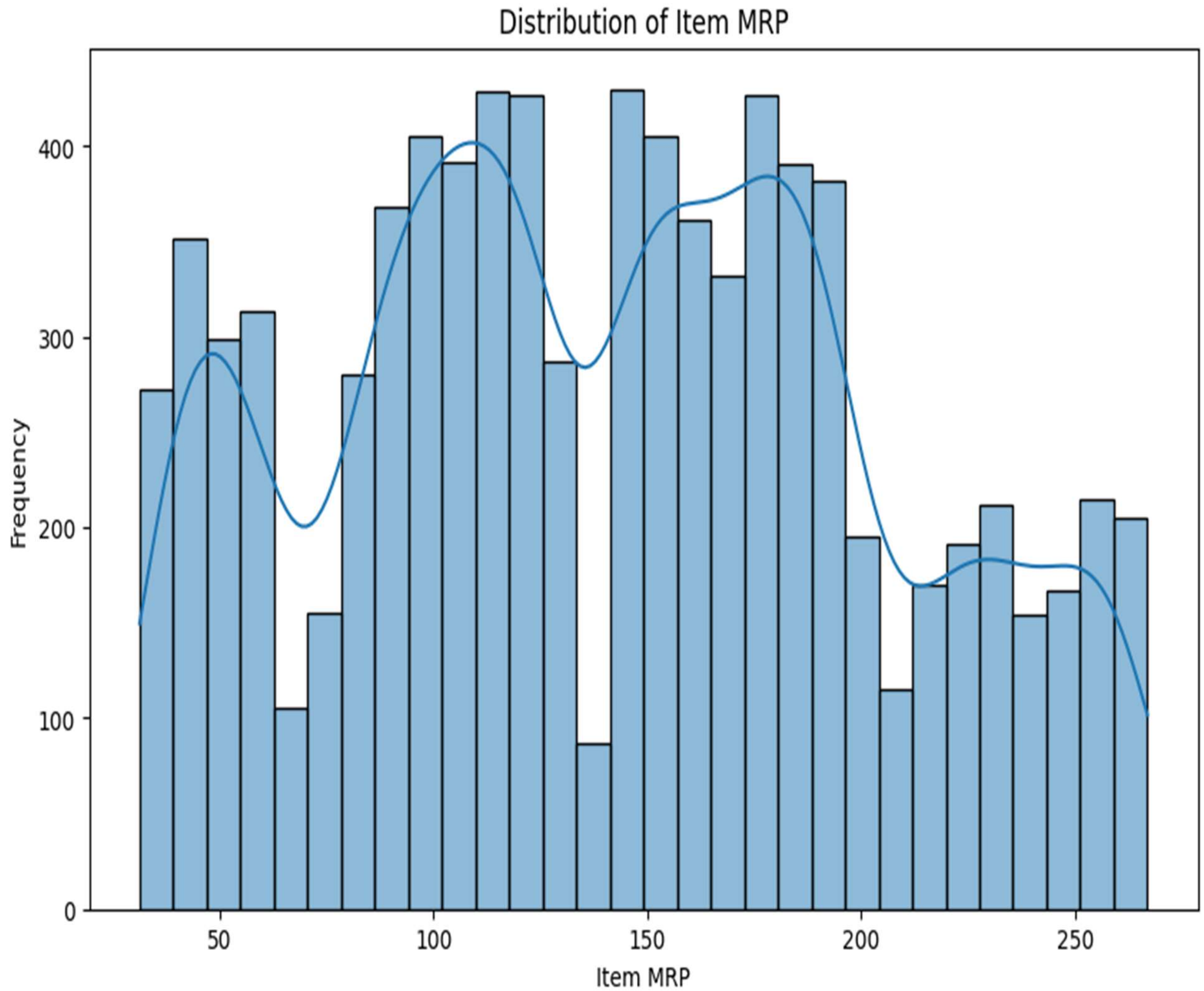
Root Mean Squared Error: 1092.6690344990404

R^2 score: 0.560728930161015

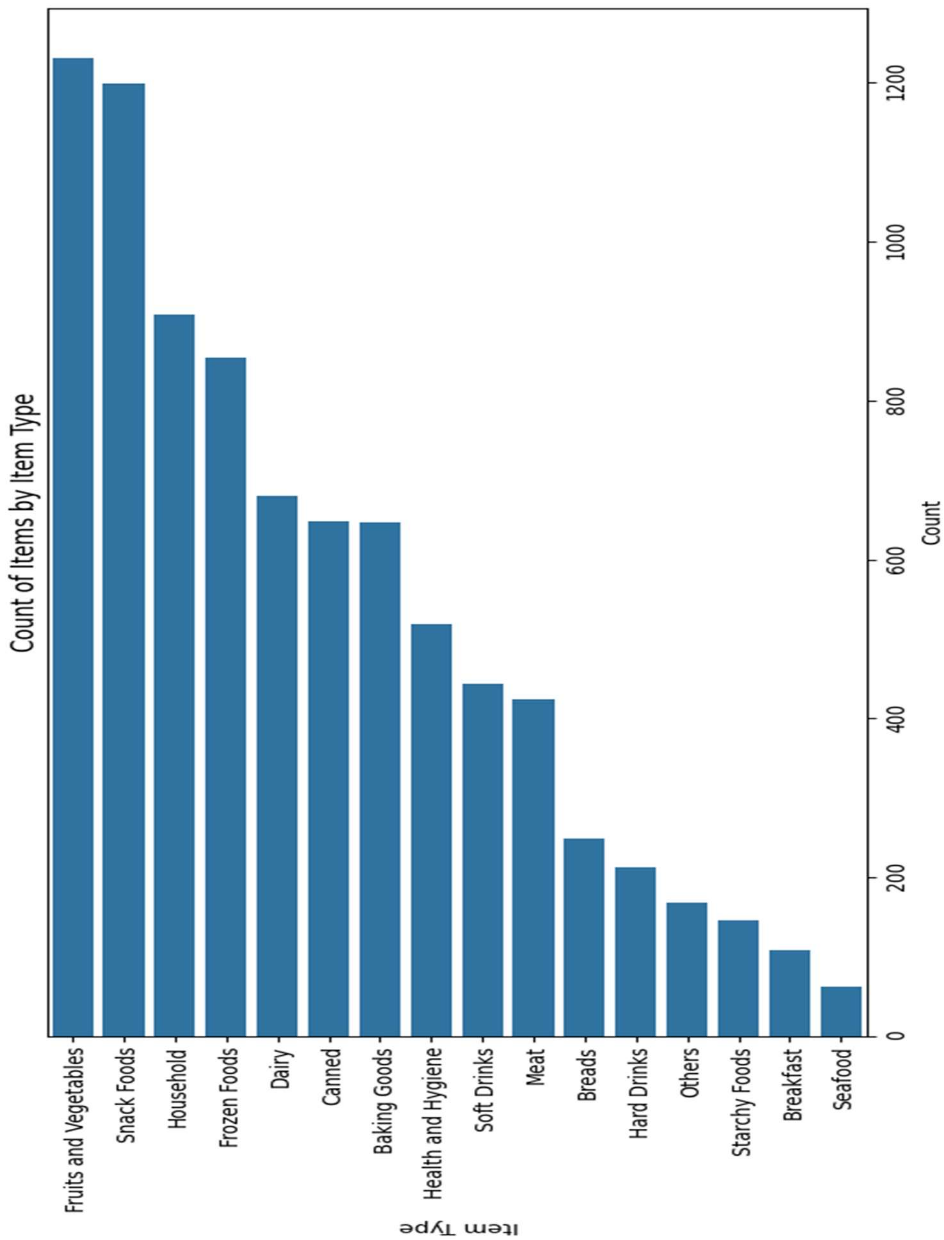
1.Distribution of item Outlet Sales



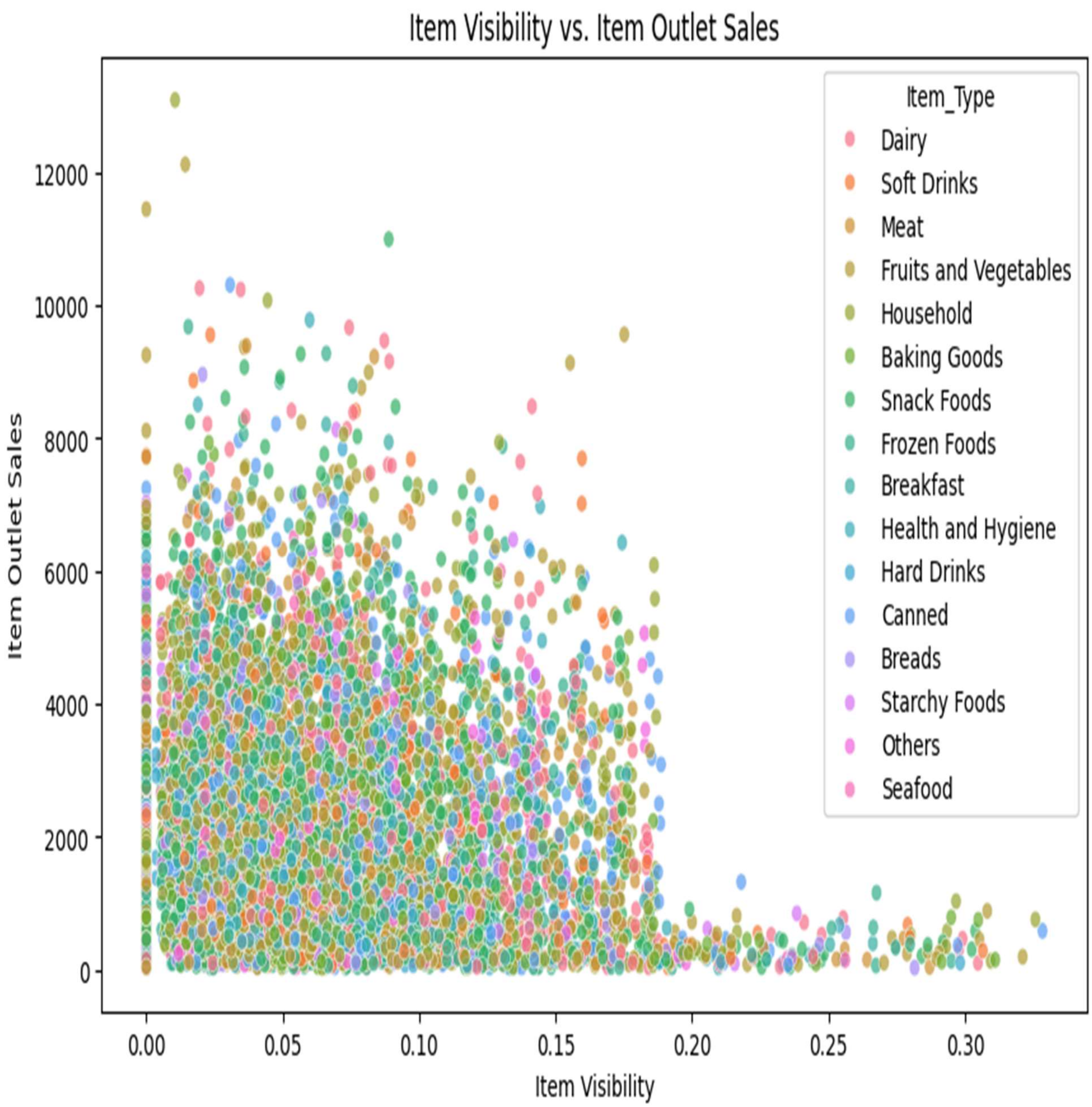
2.Distribution of item MRP.



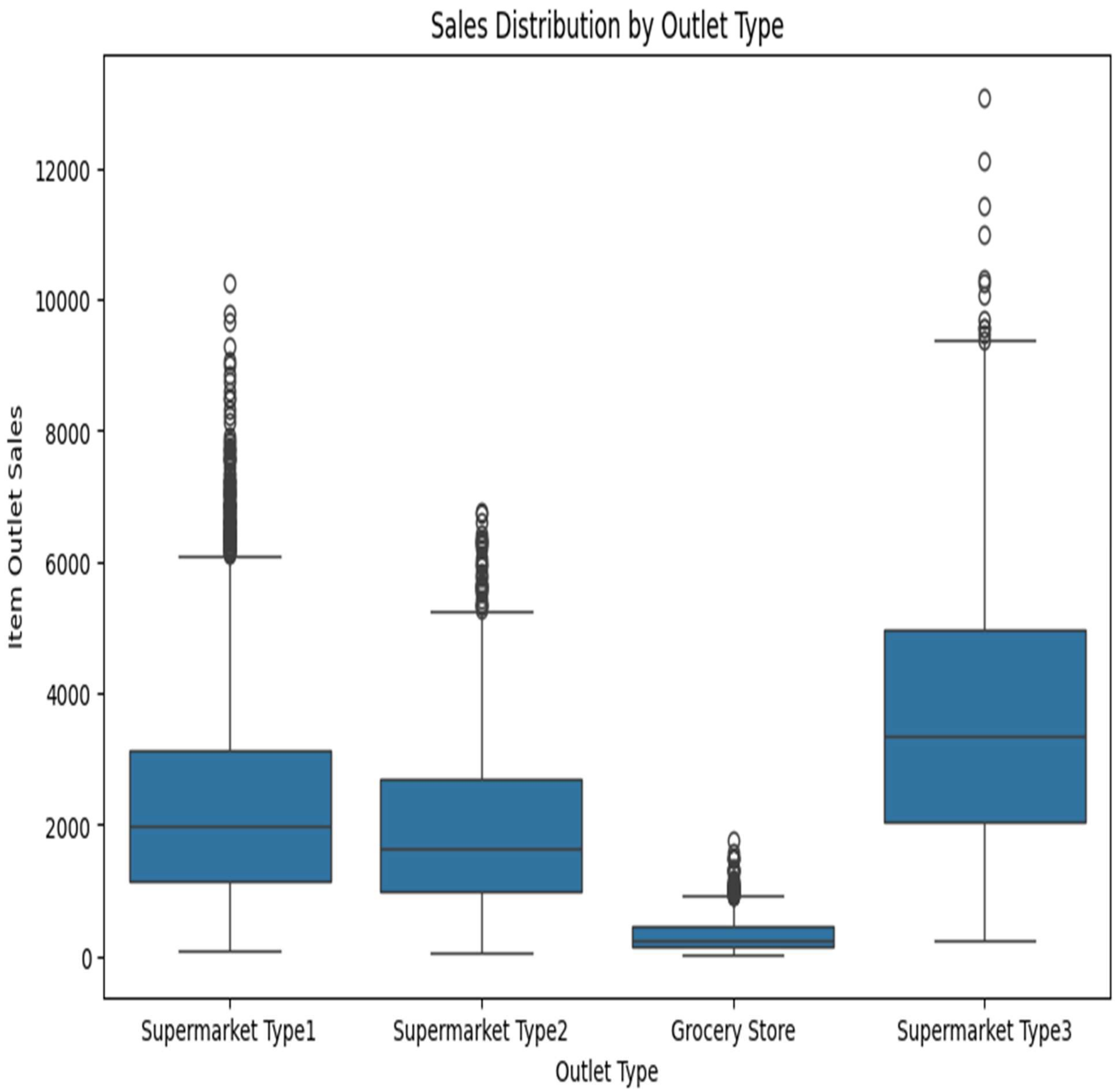
3.Count Of items by item Type



4.Item visibility vs item Outlet Sales.



5.Sales Distribution by Outlet Type.



Thank You ☺