PRODUCT DEMAND ANALYSIS

PROBLEM STATEMENT:

Create a machine learning model that forecasts product demand based on historical sales and external factors, helping businesses optimize inventory management and production planning to meet customer needs efficiently.

PHASES OF DEVELOPMENT:

PHASE 1: PROBLEM DEFINITION AND DESIGN THINKING

PROBLEM DEFINITION:

The problem is to create a machine learning model that forecasts product demand based on historical sales data and external factors. The goal is to help businesses optimize inventory management and production planning to efficiently meet customer needs. This project involves data collection, data preprocessing, feature engineering, model selection, training and evaluation.

DESIGN THINKING:

1. **Data Collection**: Collect historical sales data and external factors that influence demand, such as marketing campaigns, holidays, economic indicators, etc.
2. **Data Preprocessing**: Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.
3. **Feature Engineering**: Create additional features that capture seasonal patterns, trends, and external influences on product demand.
4. **Model Selection**: Choose suitable regression algorithms (e.g., Linear Regression, Random Forest, XG Boost) for demand forecasting.
5. **Model Training**: Train the selected model using the preprocessed data.
6. **Evaluation**: Evaluate the model's performance using appropriate regression metrics (e.g., Mean Absolute Error, Root Mean Squared Error).

PHASE 2: INNOVATION

By using incorporating time series forecasting techniques like ARIMA or Prophet for capture temporal patterns in demand data.

DATA SET LINK: https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning

PHASE 3: DEVELPOMENT PART 1

Begin building the product demand prediction model by loading and preprocessing the dataset.

PHASE 4: DEVELOPMENT PART 2

Continue building the product demand prediction model by feature engineering, model training, and evaluation.

PHASE 5: PROJECT DOCUMENTATION & SUBMISSION

Document the product demand prediction project and prepare it for submission.

DESCRIBTION OF THE DATASET:

The dataset contain the index of store id, product id, total price, base price and unsold units for 4 columns and contain the row 150151 products.

DATA PREPROCESSING STEPS:

1. Import all the necessary libraries
2. Import the dataset and read it as CSV file.
3. Check the null values
4. Fill the NA values

ANALYSIS TECHNIQUES APPLIED:

For analysis time series forecasting techniques like ARIMA model is used to capture temporal patterns in demand data. In the ARIMA model PACF and TSEA plots are used.

After analysis with ARIMA model perform the encoding categorical data, feature engineering (selection), splitting the data for model training, model training and evaluation.

In the model training there are several model like 'Linear Regression, Random forest, Support vector machine and Gradient Boosting' are used in predicting the model.

In model the 'Mean Squared Error, R-squared' are used to evaluating the model.

PROGRAM:

```
# Import necessary libraries

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler


# Data Collection

# Assuming your dataset is named 'product_demand_data.csv' and located in the same directory as your Python script

data = pd.read_csv('/content/PoductDemand.csv')


# Data Preprocessing

# Handling Missing Values (if any)

data.fillna(0, inplace=True)


data.isnull().sum()


# Data Transformation

# No categorical variables to encode in this case

# Split Data

X = data[features]  # Features

y = data[target]  # Target variable

# Split the data into training and testing sets (70-30 split)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Data Standardization (optional, but often necessary for many machine learning algorithms)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

#  Encode categorical data

import numpy as np
import pandas as pd

# One-hot encode the categorical data
encoded_df = pd.get_dummies(data)
# Print the encoded DataFrame
print(encoded_df)

# Feature Selection
features = ['ID', 'Store ID', 'Total Price', 'Base Price']  # Features
target = 'Units Sold'  # Target variable

# Histograms and Box Plots

import matplotlib.pyplot as plt

# Histograms
data[features].hist(bins=20, figsize=(12, 10))
plt.suptitle("Histograms of Features")
```

```
plt.show()

# Box Plots
data[features].plot(kind='box', vert=False, figsize=(12, 6))
plt.title("Box Plots of Features")
plt.show()

# Correlation Matrix
import seaborn as sns

correlation_matrix = data[features].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()

# Pair Plot

sns.pairplot(data[features])
plt.suptitle("Pair Plot of Features")
plt.show()

# Target Variable Distribution

plt.figure(figsize=(8, 6))
sns.histplot(data[target], bins=20, kde=True)
plt.title("Distribution of Target Variable")
plt.xlabel(target)
plt.ylabel("Frequency")
```

```
plt.show()

# Feature vs. Target Plots

for feature in features:
    plt.figure(figsize=(8, 6))
    sns.scatterplot(x=data[feature], y=data[target])
    plt.title(f"{feature} vs. {target}")
    plt.xlabel(feature)
    plt.ylabel(target)
    plt.show()

# Box Plot of Target Variable Grouped by Categorical Feature

categorical_feature = 'Store ID'  # Example categorical feature
plt.figure(figsize=(10, 6))
sns.boxplot(x=categorical_feature, y=target, data=data)
plt.title(f"Box Plot of {target} Grouped by {categorical_feature}")
plt.xlabel(categorical_feature)
plt.ylabel(target)
plt.xticks(rotation=45)
plt.show()
```

 # MODEL SELECTION

```
# Import necessary libraries for different algorithms
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
```

```python
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, r2_score


# Initialize models
linear_reg = LinearRegression()
random_forest = RandomForestRegressor(random_state=42)
svm = SVR()
gradient_boosting = GradientBoostingRegressor(random_state=42)


# Train and predict using each algorithm
models = [linear_reg, random_forest, svm, gradient_boosting]
model_names = ['Linear Regression', 'Random Forest', 'Support Vector Machine', 'Gradient Boosting']
for model, name in zip(models, model_names):
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    mse = mean_squared_error(y_test, predictions)
    r2 = r2_score(y_test, predictions)
    print(f"Model: {name}")
    print(f"Mean Squared Error: {mse:.2f}")
    print(f"R-squared: {r2:.2f}")
    print("-" * 30)
```

OUTPUT:

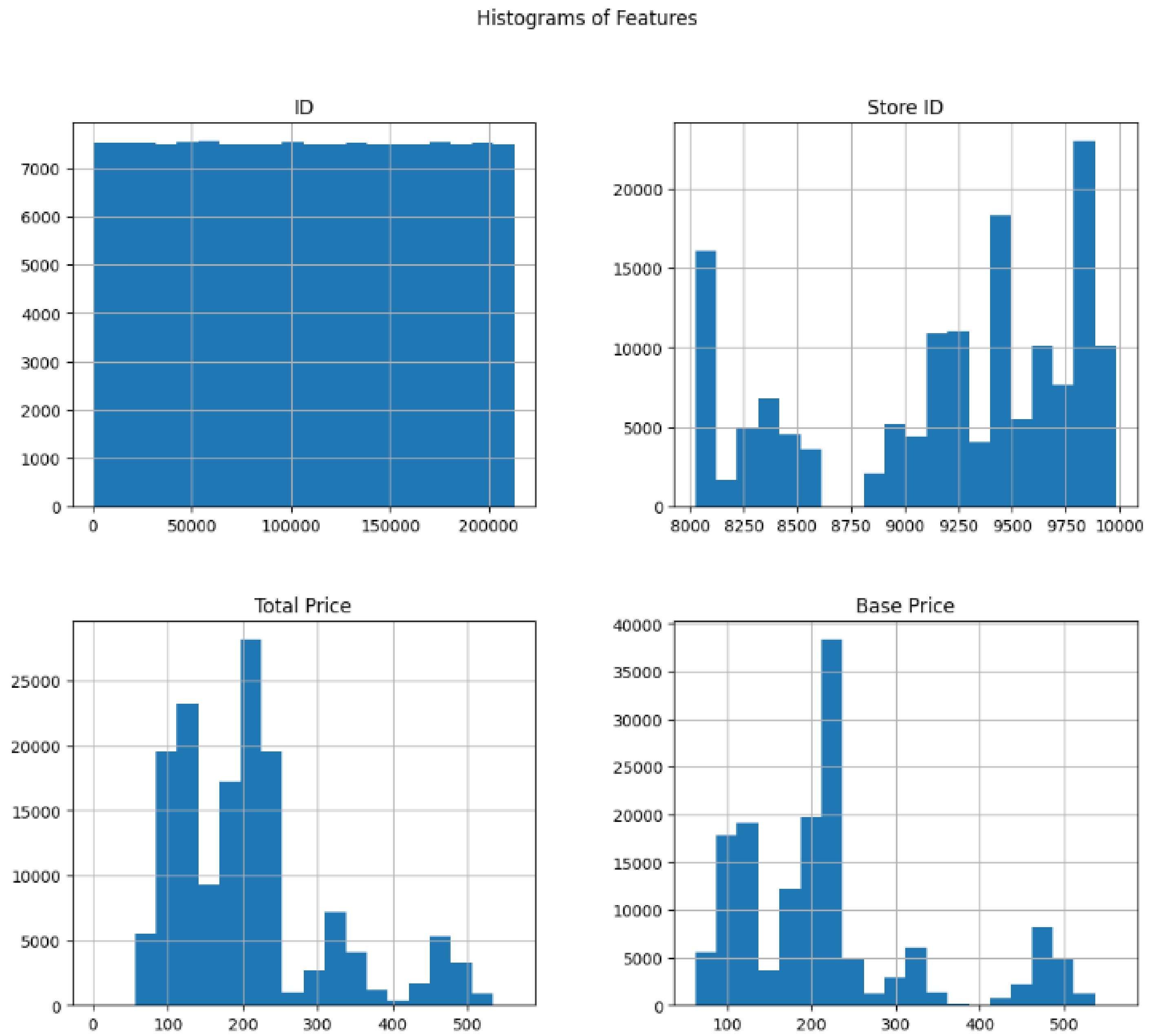
Data Preprocessing

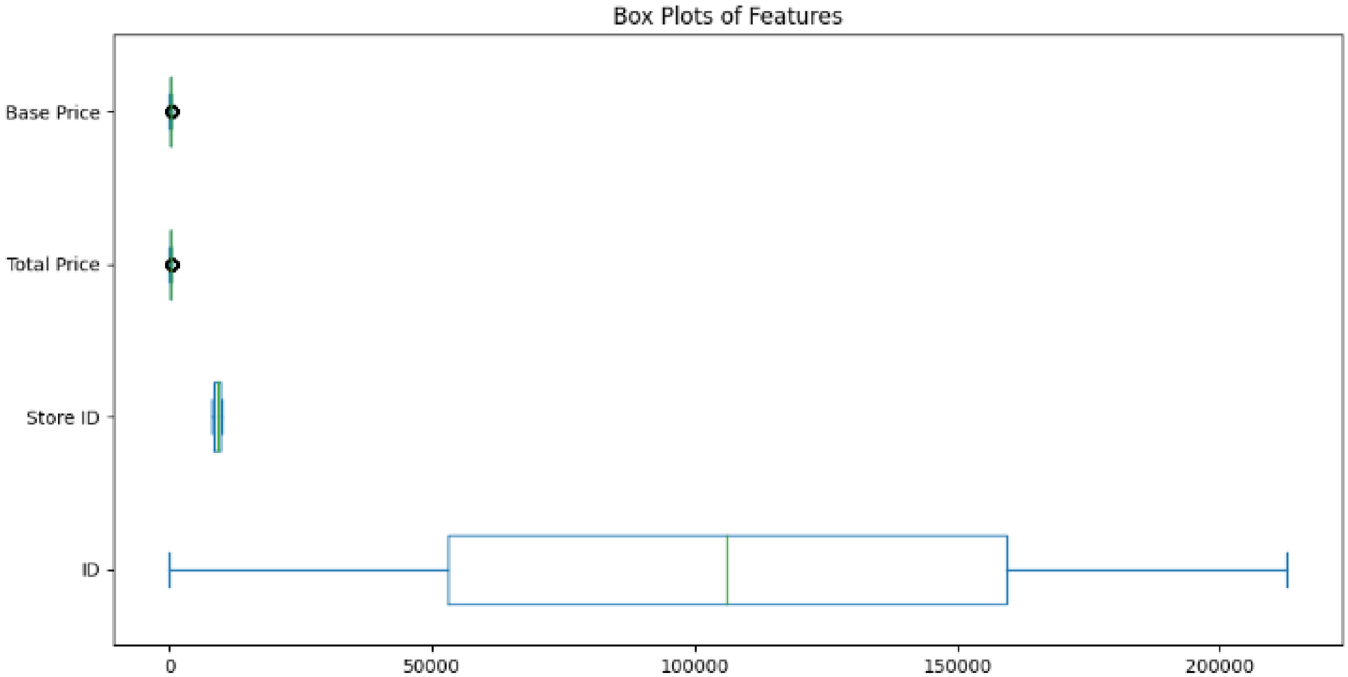
```
ID             0
Store ID       0
Total Price    0
Base Price     0
Units Sold     0
dtype: int64
```

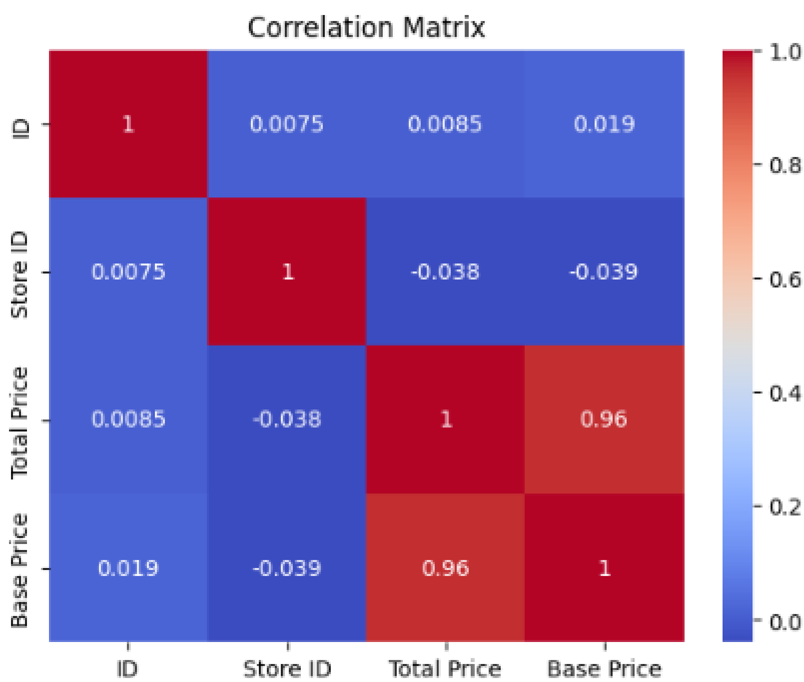
SPLIT DATA

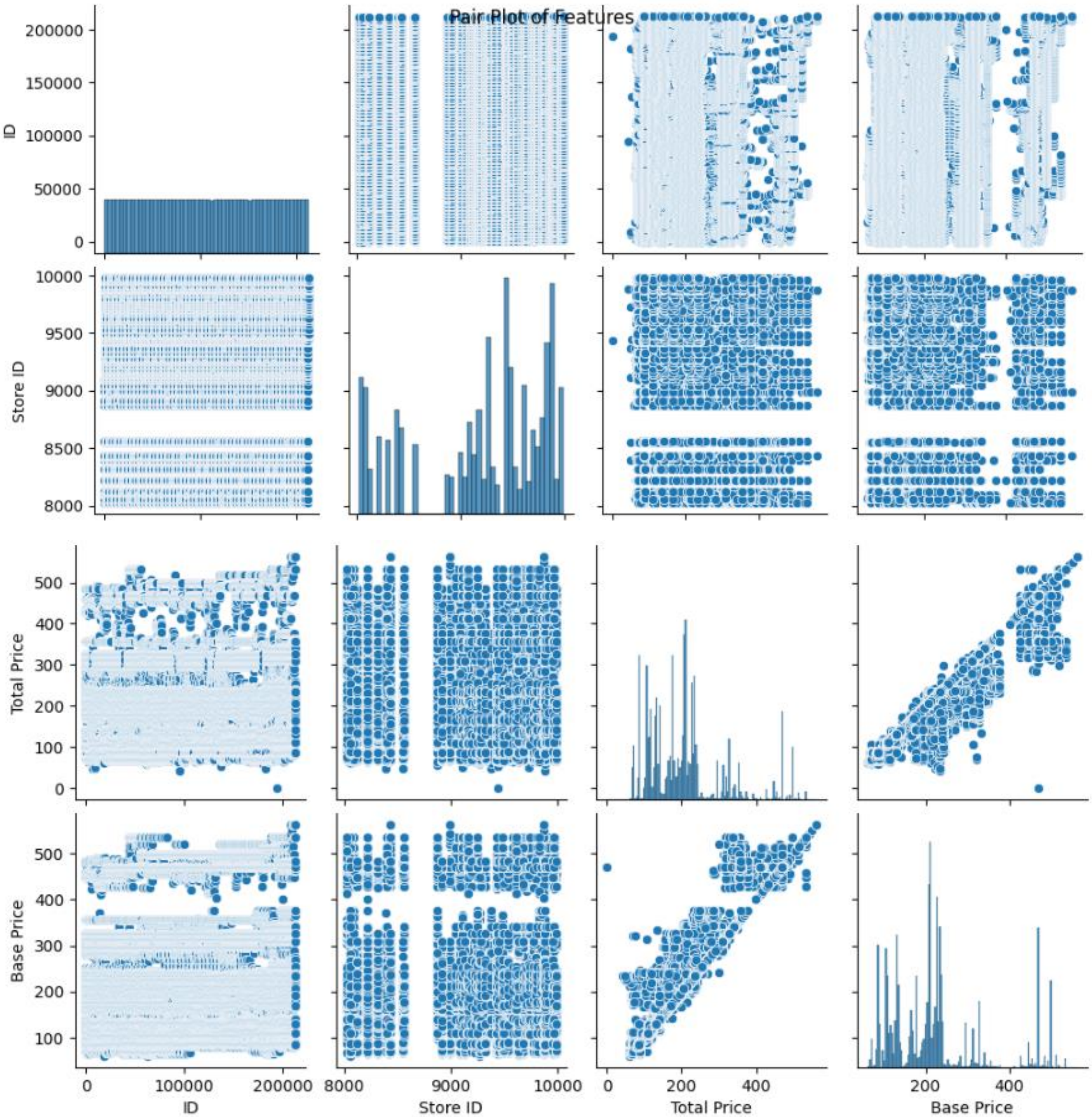
Encoding categorical data

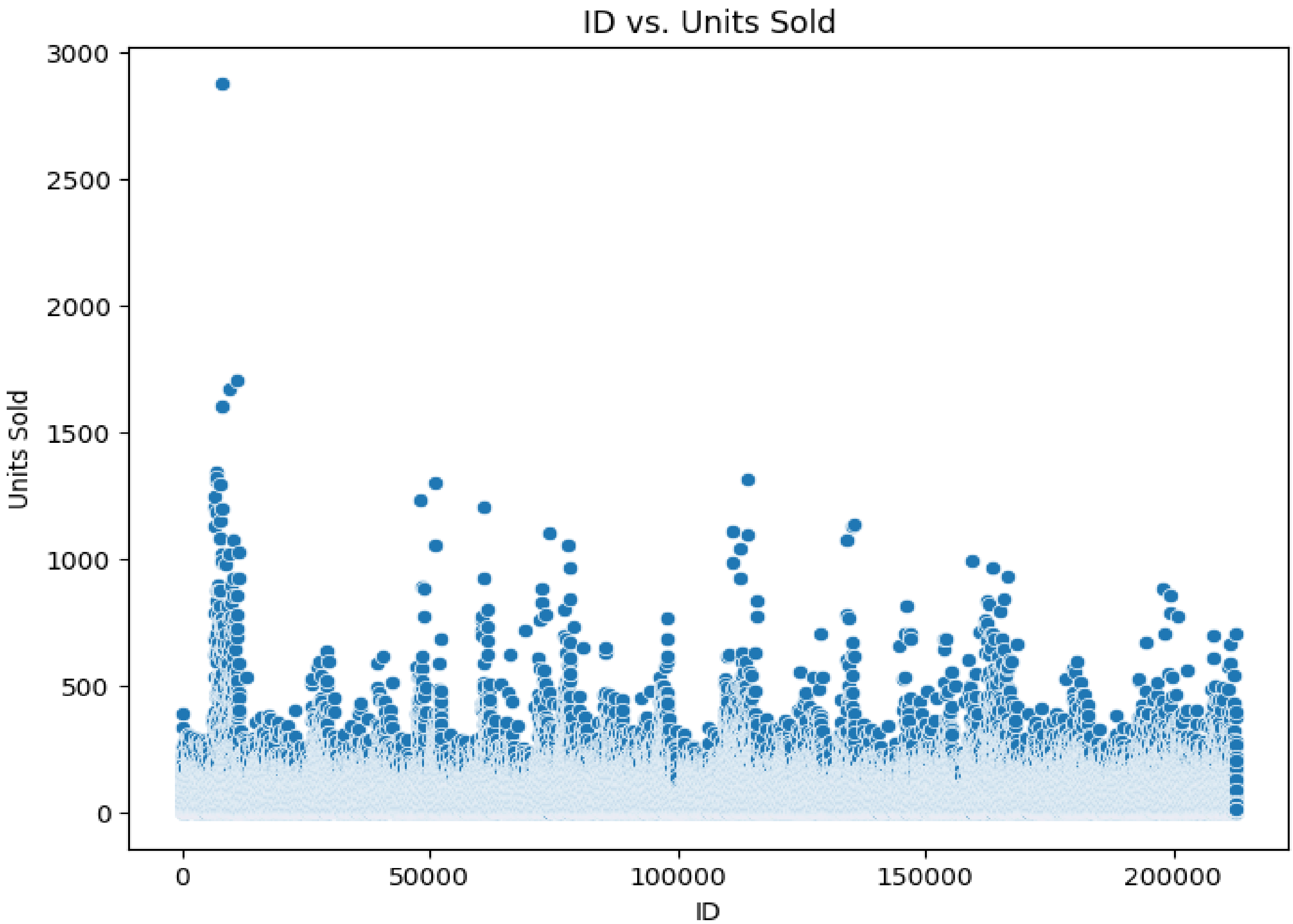
```
            ID  Store ID  Total Price  Base Price  Units Sold
0            1      8091      99.0375    111.8625          20
1            2      8091      99.0375     99.0375          28
2            3      8091     133.9500    133.9500          19
3            4      8091     133.9500    133.9500          44
4            5      8091     141.0750    141.0750          52
...        ...       ...          ...         ...         ...
150145  212638      9984     235.8375    235.8375          38
150146  212639      9984     235.8375    235.8375          30
150147  212642      9984     357.6750    483.7875          31
150148  212643      9984     141.7875    191.6625          12
150149  212644      9984     234.4125    234.4125          15

[150150 rows x 5 columns]
```

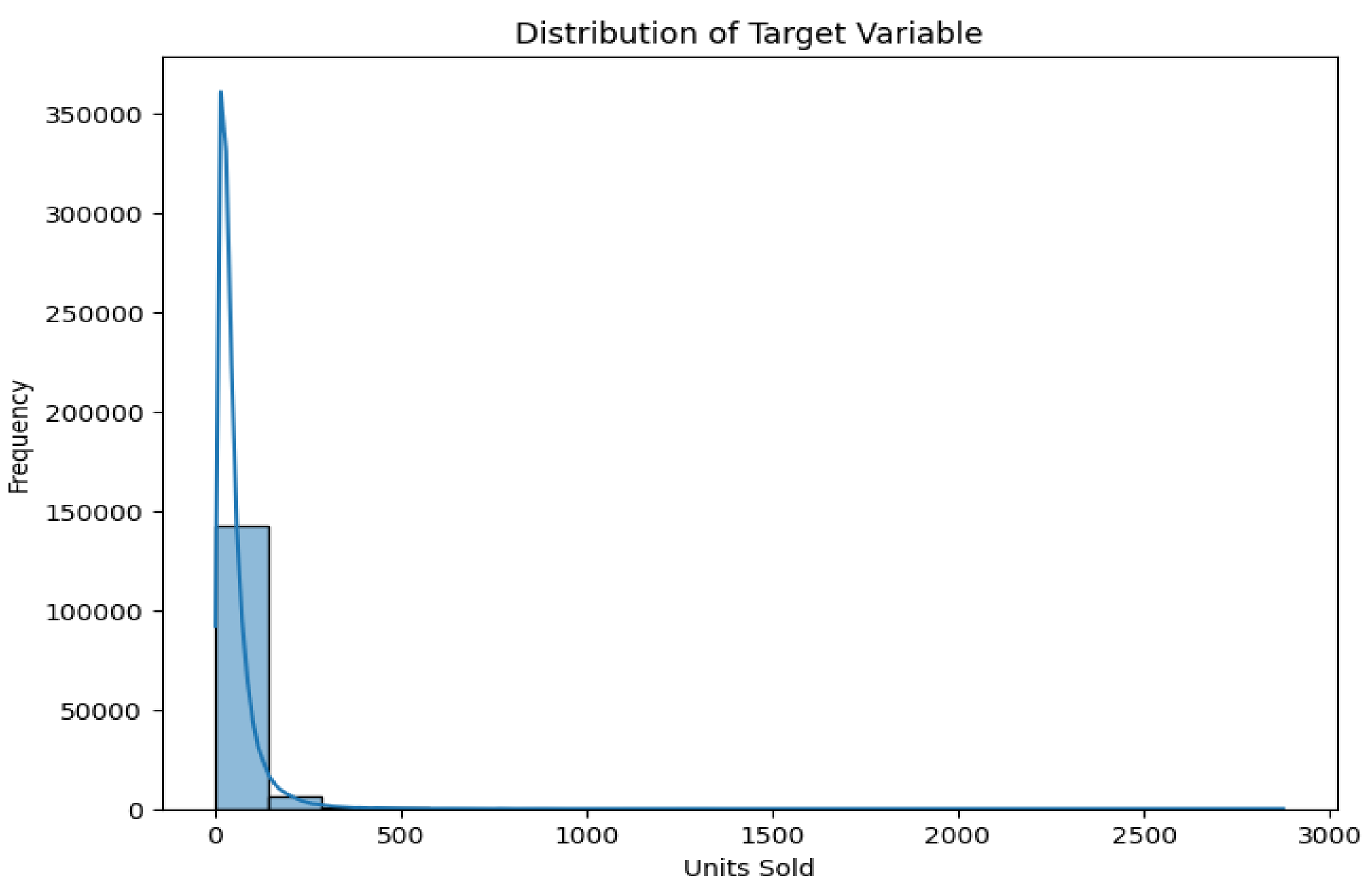

Histograms of Features
ID
0
1000
2000
3000
4000
5000
6000
7000
50000
100000
150000
200000
Store ID
5000
10000
15000
20000
8000
8250
8500
8750
9000
9250
9500
9750
10000
Total Price
25000
100
200
300
400
500
Base Price
30000
35000
40000

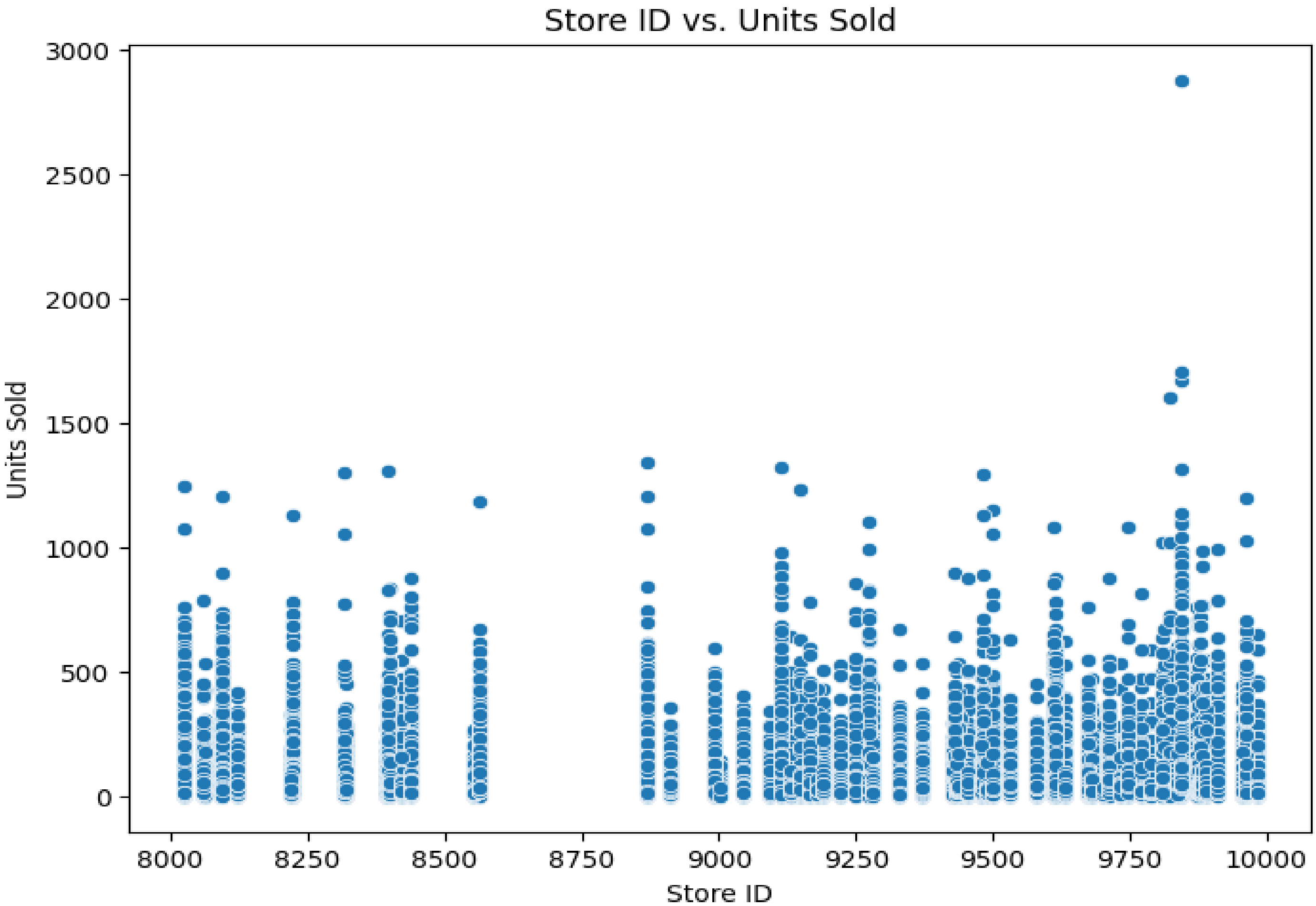

Box Plots of Features
Base Price
Total Price
Store ID
ID
0
50000
100000
150000
200000

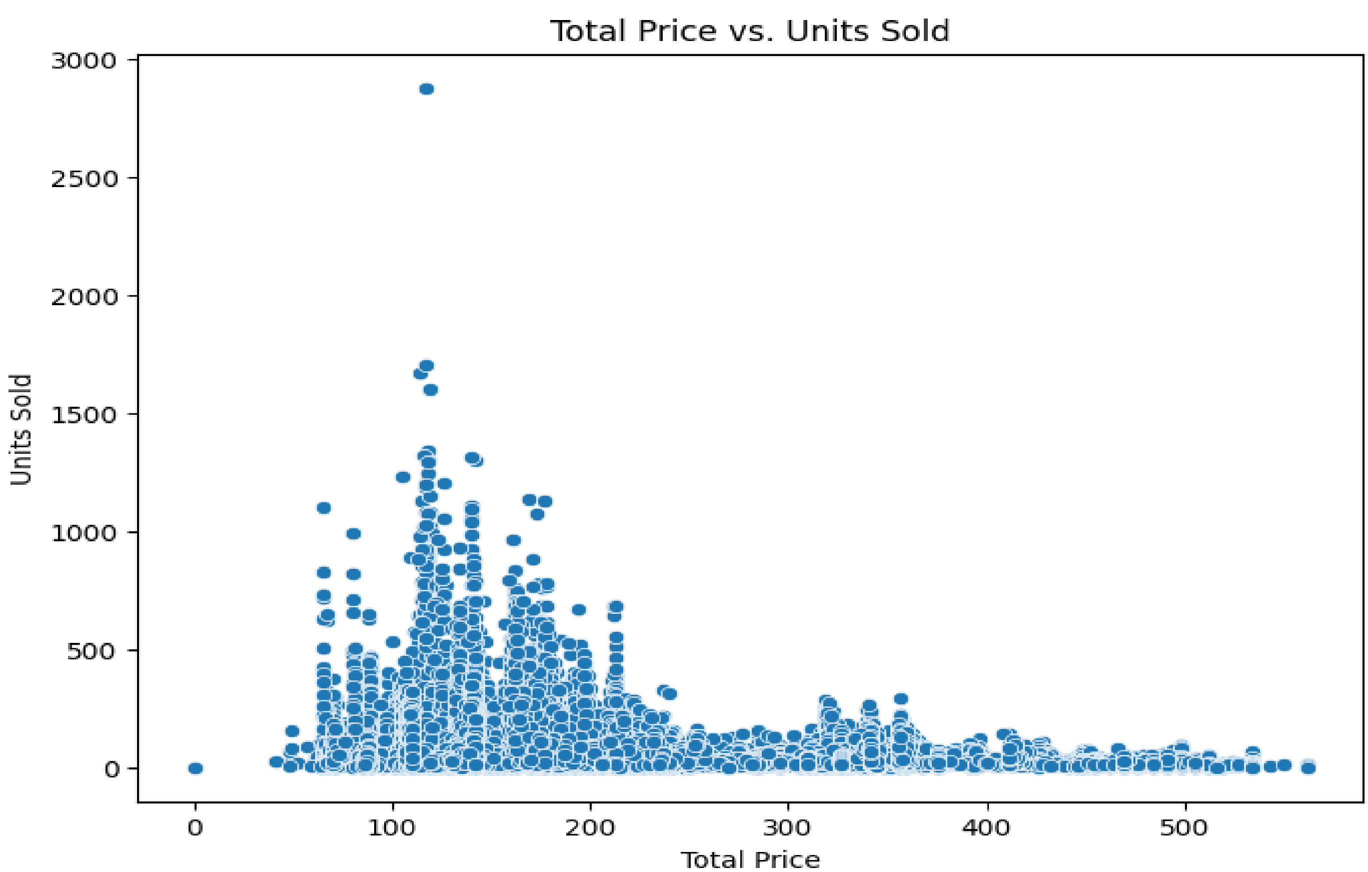

Correlation Matrix
ID
Store ID
Total Price
Base Price
1
0.0075
0.0085
0.019
0.0075
1
-0.038
-0.039
0.0085
-0.038
1
0.96
0.019
-0.039
0.96
1
1.0
0.8
0.6
0.4
0.2
0.0

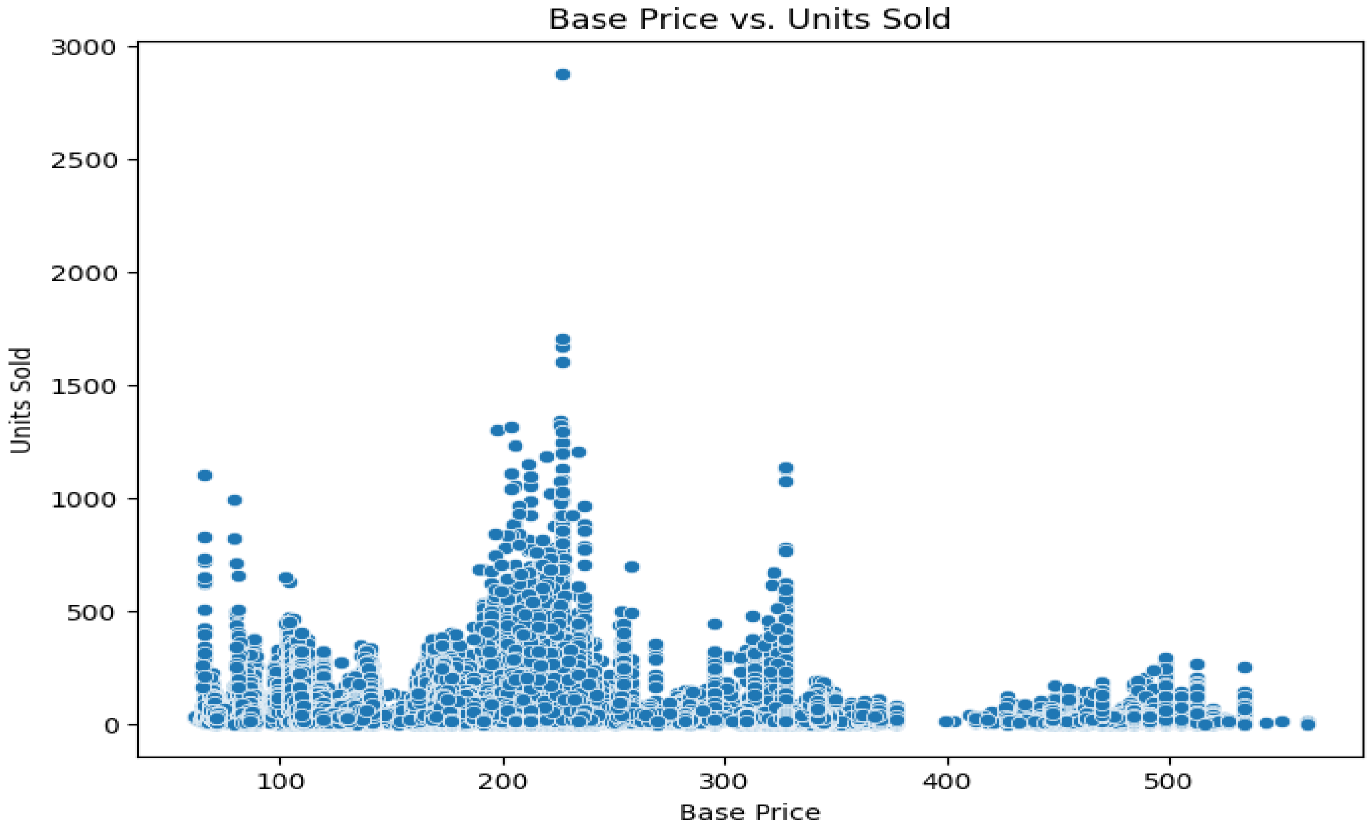

Pair Plot of Features
ID
Store ID
Total Price
Base Price
0
50000
100000
150000
200000
8000
8500
9000
9500
10000
100
200
300
400
500

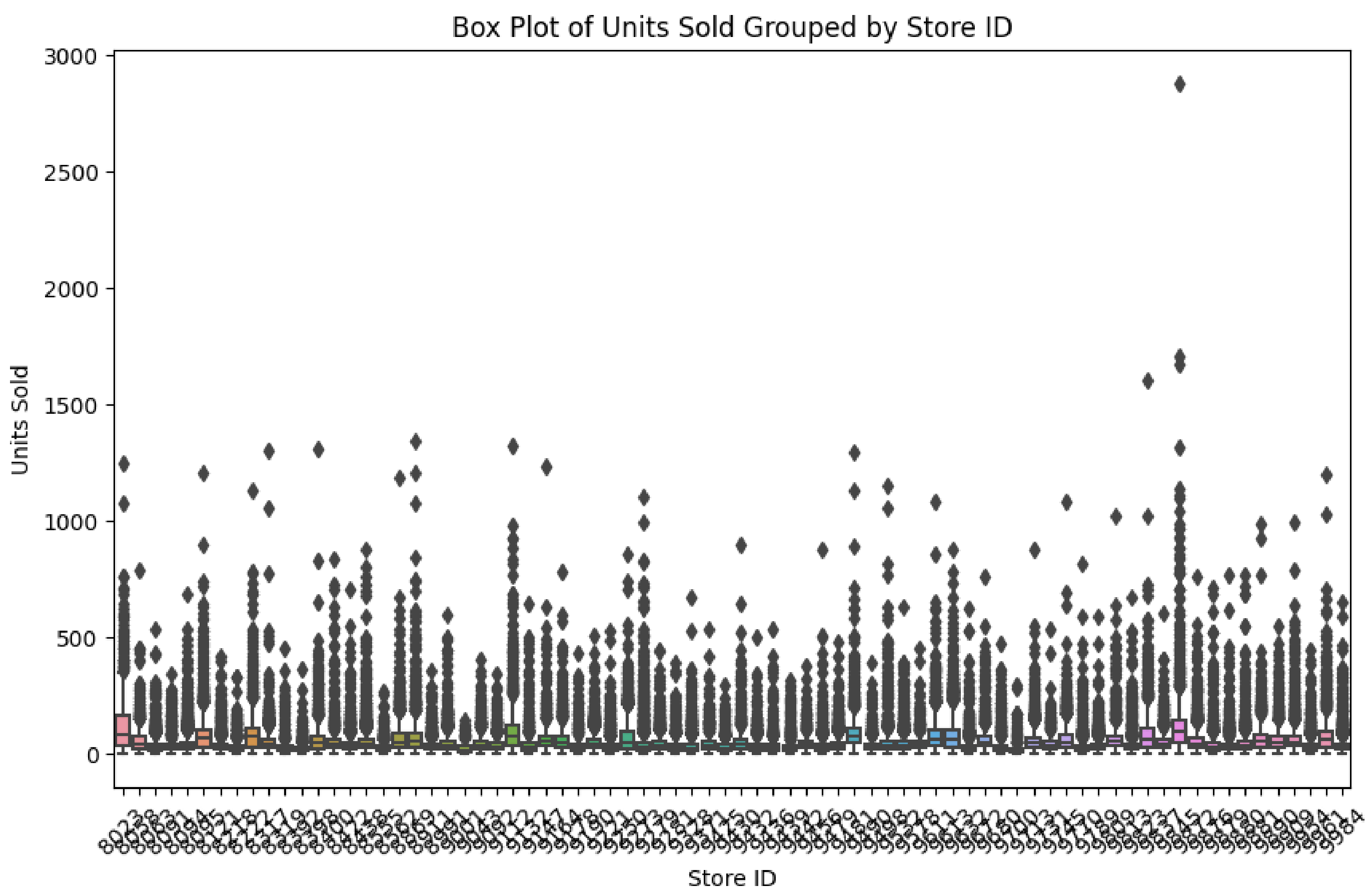

ID vs. Units Sold
Units Sold
3000
2500
2000
1500
1000
500
0
0
50000
100000
150000
200000
ID


Distribution of Target Variable
Frequency
350000
300000
250000
200000
150000
100000
50000
0
0
500
1000
1500
2000
2500
3000
Units Sold


Store ID vs. Units Sold
Units Sold
Store ID
3000
2500
2000
1500
1000
500
0
8000
8250
8500
8750
9000
9250
9500
9750
10000


Total Price vs. Units Sold
Units Sold
Total Price
3000
2500
2000
1500
1000
500
0
0
100
200
300
400
500


Base Price vs. Units Sold
Units Sold
3000
2500
2000
1500
1000
500
0
100
200
300
400
500
Base Price


Box Plot of Units Sold Grouped by Store ID
Units Sold
3000
2500
2000
1500
1000
500
0
Store ID

```
Model: Linear Regression
Mean Squared Error: 2844.00
R-squared: 0.15
------------------------------
Model: Random Forest
Mean Squared Error: 1156.38
R-squared: 0.66
------------------------------
Model: Support Vector Machine
Mean Squared Error: 2956.17
R-squared: 0.12
------------------------------
Model: Gradient Boosting
Mean Squared Error: 1885.63
R-squared: 0.44
------------------------------
```

ARIMA MODEL:

LINK:

https://colab.research.google.com/drive/1gNQWTceGFUHsXDgdAMRgesCE5hFBYAAt?usp=sharing

CONCLUSION:

Random Forest and Gradient Boosting model typically perform well in a variety of datasets due to their ability to capture complex patterns in the data.

Support Vector Machine (SVM) might perform well if the dataset has high dimensionality and complex relationships, although it might require fine-tuning of hyper parameter for optimal results.

Linear Regression provides a basic understanding of the relationships between variables but might not capture intricate patterns present in the data.