APPLIED DATA SCIENCE

IBM NAAN MUDHALVAN Phase-4

TEAM MEMBERS

- DHANULESH.S
- SURIYA.V
- NARENTHIRA KUMAR.G
- MAHESWAREN.R
- ARYAN REDDY.R

PROJECT TITLE : PRODUCT DEMAND PREDICTION WITH MACHINE LEARNING

PHASE 3 : DEVELOPMENT PART 2

TOPIC : Start building the product demand prediction model by Feature engineering ,Model training and evaluation dataset. Collect and preprocess the historical sales data and external factors for analysis.

INTRODUCTION:

Feature engineering, model training, and dataset evaluation are crucial aspects of a data science or machine learning project. Let's provide an introduction to each of these components:

1. Feature Engineering:

- Feature engineering is the process of selecting, transforming, or creating new features (variables) from the raw data to improve the performance of a machine learning model.
- Feature engineering involves domain knowledge and creativity to extract meaningful information from the dataset.
- Typical feature engineering tasks include handling missing data, encoding categorical variables, scaling numeric features, creating interaction terms, and more.
- Well-engineered features can make models more effective, leading to better predictions and insights.

2. Model Training:

- Model training is the stage in which you use your prepared dataset to build a predictive or descriptive model.
- This process involves selecting an appropriate algorithm (e.g., regression, decision trees, neural networks) and using the data to fit the model.
- During training, the model learns the relationships between the features and the target variable through optimization techniques (e.g., gradient descent) and by minimizing a specific loss function.
- The trained model can make predictions on new, unseen data.

3. Dataset Evaluation:

- Dataset evaluation focuses on assessing the quality and performance of the dataset before and after feature engineering, as well as the model after training.
- Common evaluation metrics depend on the type of problem, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) for regression, and accuracy, precision, recall, F1-score

for classification.
- Cross-validation techniques are often used to evaluate a model's performance by partitioning the dataset into training and testing subsets.
- Model performance evaluation ensures that the model generalizes well to new, unseen data.

In a typical workflow, feature engineering precedes model training, as creating informative features is essential for building a successful model. Once the model is trained, it undergoes evaluation using relevant metrics and techniques to determine its predictive accuracy and reliability. This iterative process may involve going back to feature engineering or fine-tuning the model to achieve the best possible results.

GIVEN DATASET:
https://www.kaggle.com/datasets/chakradharmattapalli/product-demand-prediction-with-machine-learning

PROCEDURE FOR THE DEVELOPMENT PART-2

Certainly, here is a high-level procedure for the entire process of feature engineering, model training, and dataset evaluation in a machine learning project:

1. Data Collection and Understanding:
- Obtain the dataset that contains relevant information about the problem you want to solve.
- Understand the data sources, features, and the target variable.

2. Data Preprocessing:
- Handle missing data by imputing or removing rows/columns as needed.
- Address outliers that may skew the results.
- Encode categorical variables using techniques like one-hot encoding or label encoding.
- Normalize or scale numerical features to ensure they are on a similar scale.
- Explore the dataset through descriptive statistics and visualization to gain insights.

3. Feature Engineering:

- Create new features based on domain knowledge and data analysis.
- Perform feature selection to choose the most relevant features for the model.
- Transform or engineer features as needed (e.g., log transformation, interaction terms).
- Handle class imbalance issues if you're dealing with classification.

4. Data Split:

- Split the dataset into training, validation, and testing sets. Common ratios are 70-30 or 80-20 for training and testing, with the validation set used for hyperparameter tuning.

5. Model Selection:

- Choose an appropriate machine learning algorithm based on the nature of the problem (regression, classification, etc.) and the dataset size.

6. Model Training:

- Train the selected model using the training dataset.
- Use hyperparameter tuning techniques to optimize the model's performance.

7. Model Evaluation:

- Evaluate the model's performance using metrics relevant to the problem type (e.g., MSE, RMSE, accuracy, precision, recall).
- Use cross-validation to estimate model performance and assess its generalization ability.
- Analyze the model's predictions and errors to gain insights into its strengths and weaknesses.

8. Model Fine-Tuning:

- Adjust hyperparameters and model features based on the evaluation results.
- Repeat model training and evaluation as necessary.

9. Deployment:

- Deploy the trained model in a production environment to make predictions on new, unseen data.

10. Monitoring and Maintenance:

- Continuously monitor the model's performance in production and retrain it as needed with new data.
- Maintain documentation on model updates, changes in data, and any retraining procedures.

11. Documentation:

- Thoroughly document the entire process, including data sources, preprocessing steps, feature engineering, model selection, and evaluation results.

The process is often iterative, with feedback loops between feature engineering, model training, and evaluation. The goal is to create a well-performing and robust machine learning model that can make accurate predictions or classifications.

MACHINE LEARNING MODELS FOR MODEL TRAINING

- LINEAR REGRESSION
- RANDOM FOREST TREE
- EXTREME GRADIENT BOOSTING
- LASSO REGRESSION
- RIDGE REGRESSION

PROGRAMS:

Linear Regression

Import libraries

In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load the dataset

In [4]:
```
data = pd.read_csv('PoductDemand.csv')
data.head()
```

Out[4]:

	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

Data Cleaning

In [9]:
```
data.fillna(0, inplace=True)
```

Define features and target variable

In [10]:
```
X = data[['Store ID', 'Total Price', 'Base Price']]
y = data['Units Sold']
```

Data Split

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection

In [12]:
```
model = LinearRegression()
```

Model Training

In [13]:
```
model.fit(X_train, y_train)
```

Out[13]:
```
LinearRegression()
```

Model Evaluation

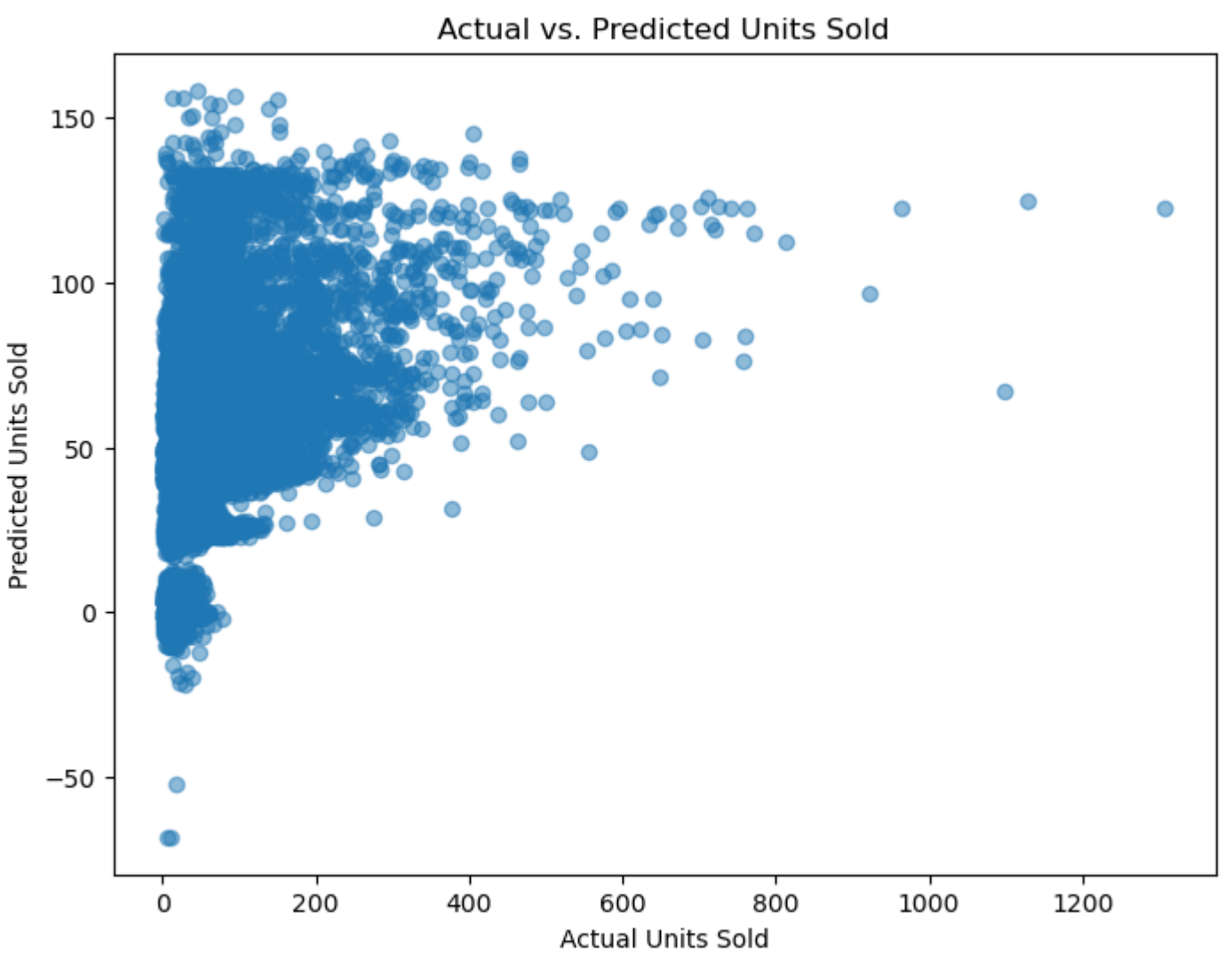
In [14]:
```
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
```

In [15]:
```
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 32.626495087826754
Mean Squared Error: 2780.795549996203
Root Mean Squared Error: 52.73324899905375
R-squared: 0.15248746523857437
```

Visualization - Scatter plot of Actual vs. Predicted

In [16]:
```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual Units Sold')
plt.ylabel('Predicted Units Sold')
plt.title('Actual vs. Predicted Units Sold')
plt.show()
```



Random Forest Regressor

Import libraries

In [17]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load the dataset

In [18]:
```
data = pd.read_csv('PoductDemand.csv')
data.head()
```

Out[18]:

	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

Data Cleaning

In [19]:
```
data.fillna(0, inplace=True)
```

Define features and target variable

In [20]:
```
X = data[['Store ID', 'Total Price', 'Base Price']]
y = data['Units Sold']
```

Data Split

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection and Training

In [22]:
```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[22]:
```
RandomForestRegressor(random_state=42)
```

Model Evaluation

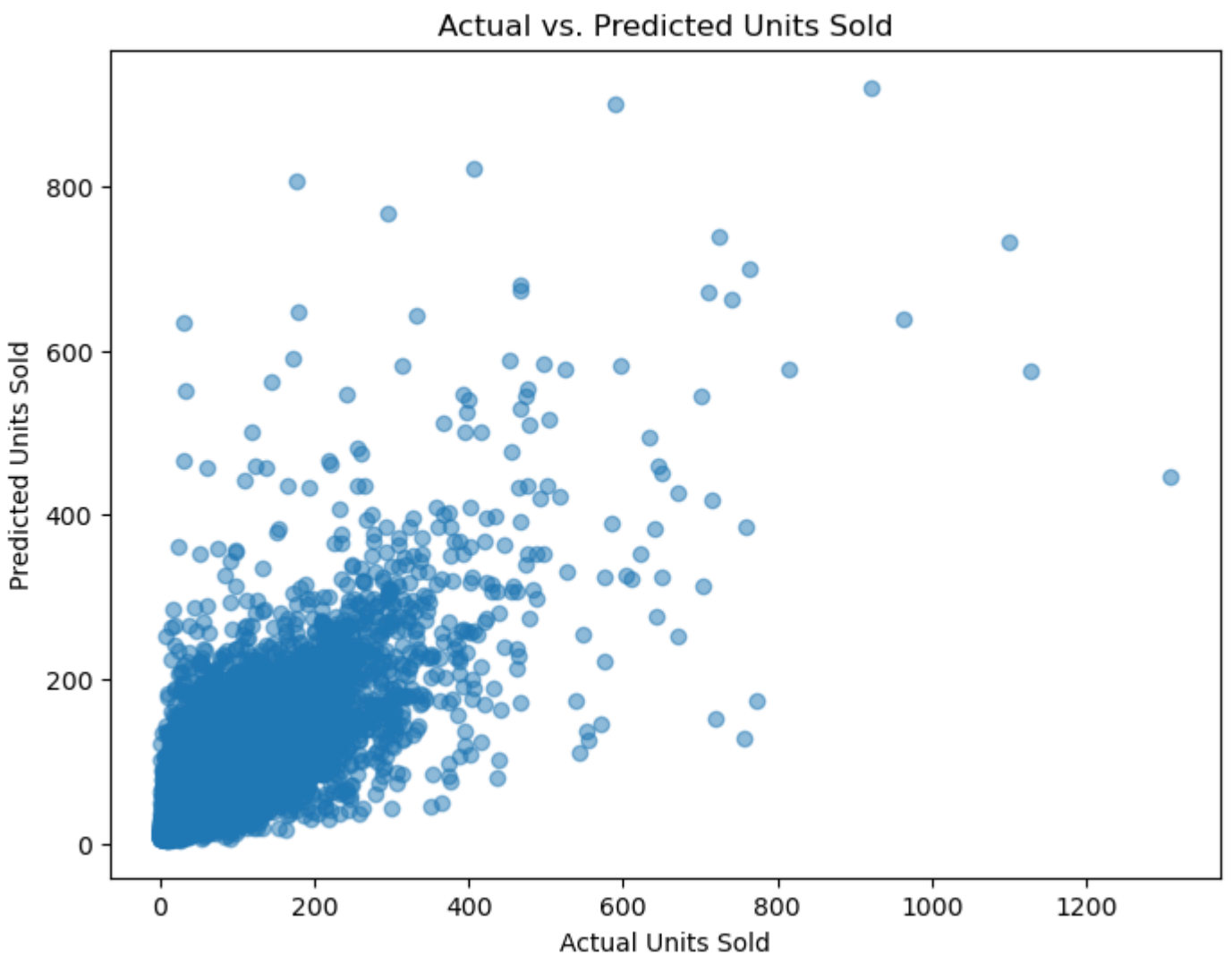
In [23]:
```
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
```

In [24]:
```
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 19.003132154269775
Mean Squared Error: 1251.6950005899928
Root Mean Squared Error: 35.37930186691072
R-squared: 0.618516649776832
```

Visualization - Scatter plot of Actual vs. Predicted

In [25]:
```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual Units Sold')
plt.ylabel('Predicted Units Sold')
plt.title('Actual vs. Predicted Units Sold')
plt.show()
```



Extreme Gradient Boosting

Import libraries

In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load the dataset

In [3]:
```
data = pd.read_csv('PoductDemand.csv')
data.head()
```

Out[3]:

	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

Data Cleaning

In [4]:
```
data.fillna(0, inplace=True)
```

Define features and target variable

In [5]:
```
X = data[['Store ID', 'Total Price', 'Base Price']]
y = data['Units Sold']
```

Data Split

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection and Training

In [8]:
```
model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[8]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=100, n_jobs=None,
             num_parallel_tree=None, random_state=42, ...)
```

Model Evaluation

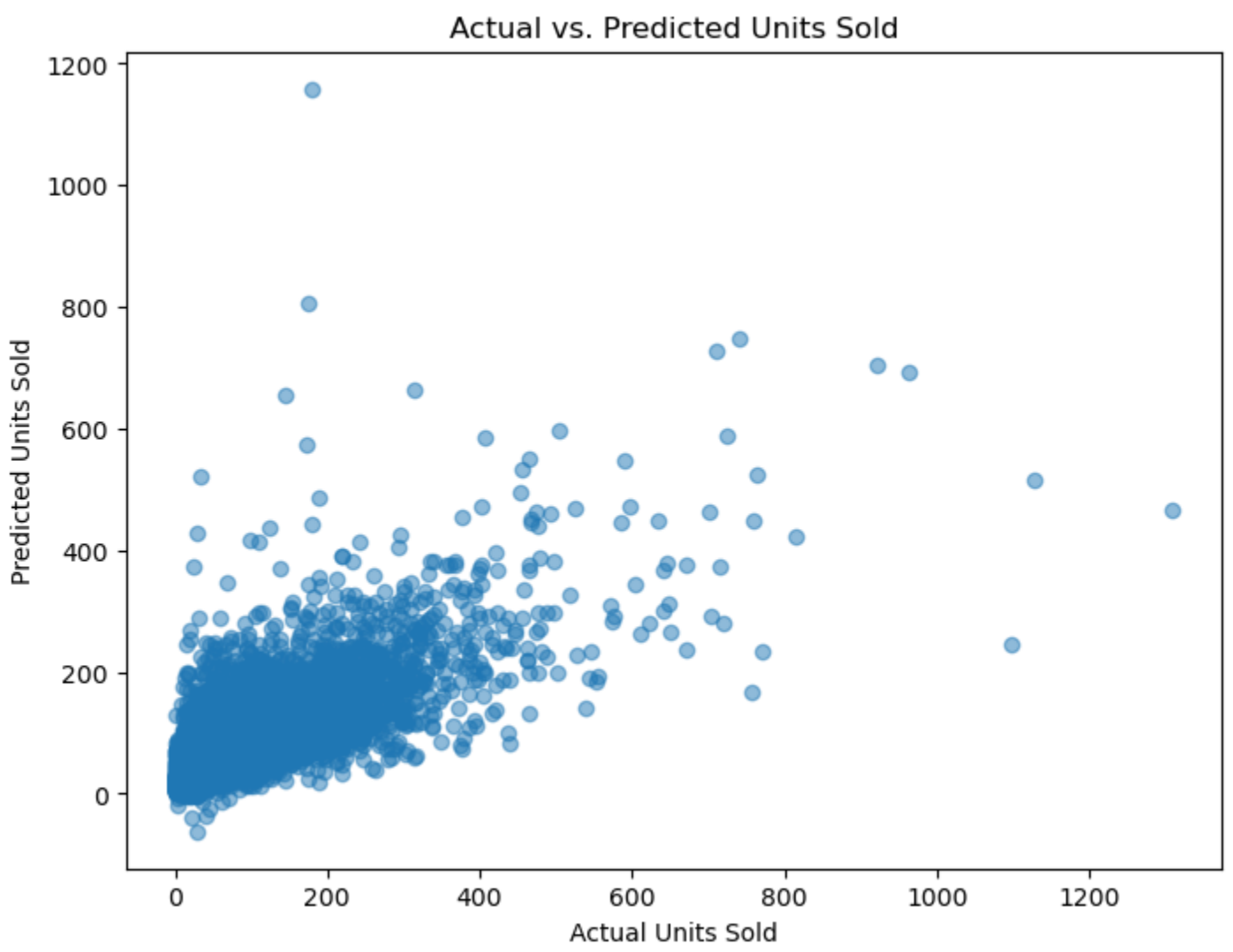
In [9]:
```
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
```

In [10]:
```
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 20.317752728781308
Mean Squared Error: 1289.1644075466604
Root Mean Squared Error: 35.9049356989629
R-squared: 0.6070969709493481
```

Visualization - Scatter plot of Actual vs. Predicted

In [11]:
```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual Units Sold')
plt.ylabel('Predicted Units Sold')
plt.title('Actual vs. Predicted Units Sold')
plt.show()
```



Lasso Regression:

Import libraries

In [13]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load the dataset

In [14]:
```
data = pd.read_csv('PoductDemand.csv')
data.head()
```

Out[14]:

	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

Data Cleaning

In [15]:
```
data.fillna(0, inplace=True)
```

Define features and target variable

In [16]:
```
X = data[['Store ID', 'Total Price', 'Base Price']]
y = data['Units Sold']
```

Data Split

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection and Training

In [18]:
```
model = Lasso(alpha=1.0)
model.fit(X_train, y_train)
```

Out[18]:
```
Lasso()
```

Model Evaluation

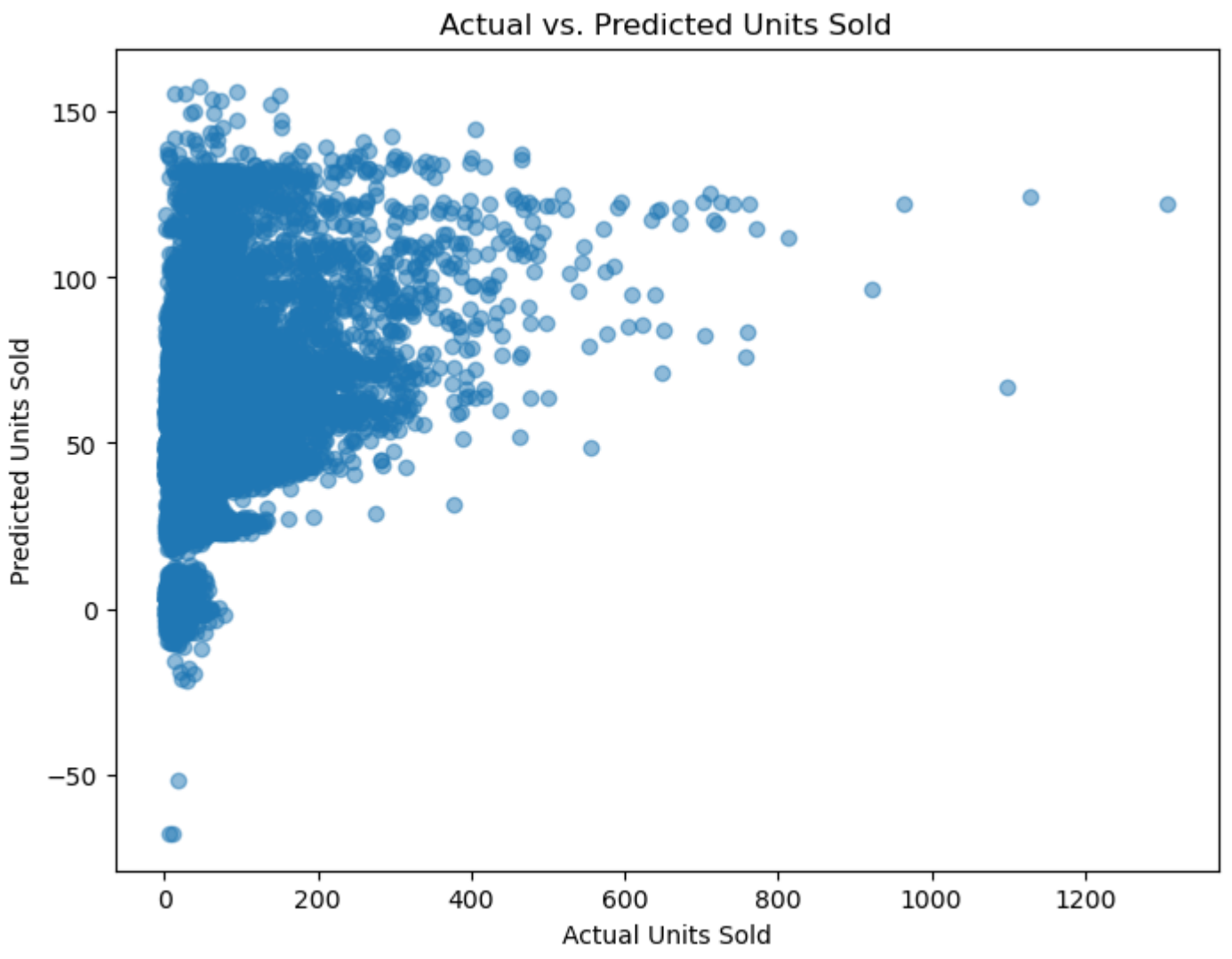
In [19]:
```
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
```

In [20]:
```
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 32.617812650674445
Mean Squared Error: 2780.671174805353
Root Mean Squared Error: 52.73206969961783
R-squared: 0.1525253714892726
```

Visualization - Scatter plot of Actual vs. Predicted

In [21]:
```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual Units Sold')
plt.ylabel('Predicted Units Sold')
plt.title('Actual vs. Predicted Units Sold')
plt.show()
```



Ridge Regression:

Import libraries

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

Load the dataset

In [2]:
```
data = pd.read_csv('PoductDemand.csv')
data.head()
```

Out[2]:

	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

Data Cleaning

In [3]:
```
data.fillna(0, inplace=True)
```

Define features and target variable

In [4]:
```
X = data[['Store ID', 'Total Price', 'Base Price']]
y = data['Units Sold']
```

Data Split

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Model Selection and Training

In [6]:
```
model = Ridge(alpha=1.0)
model.fit(X_train, y_train)
```

Out[6]:
```
Ridge()
```

Model Evaluation

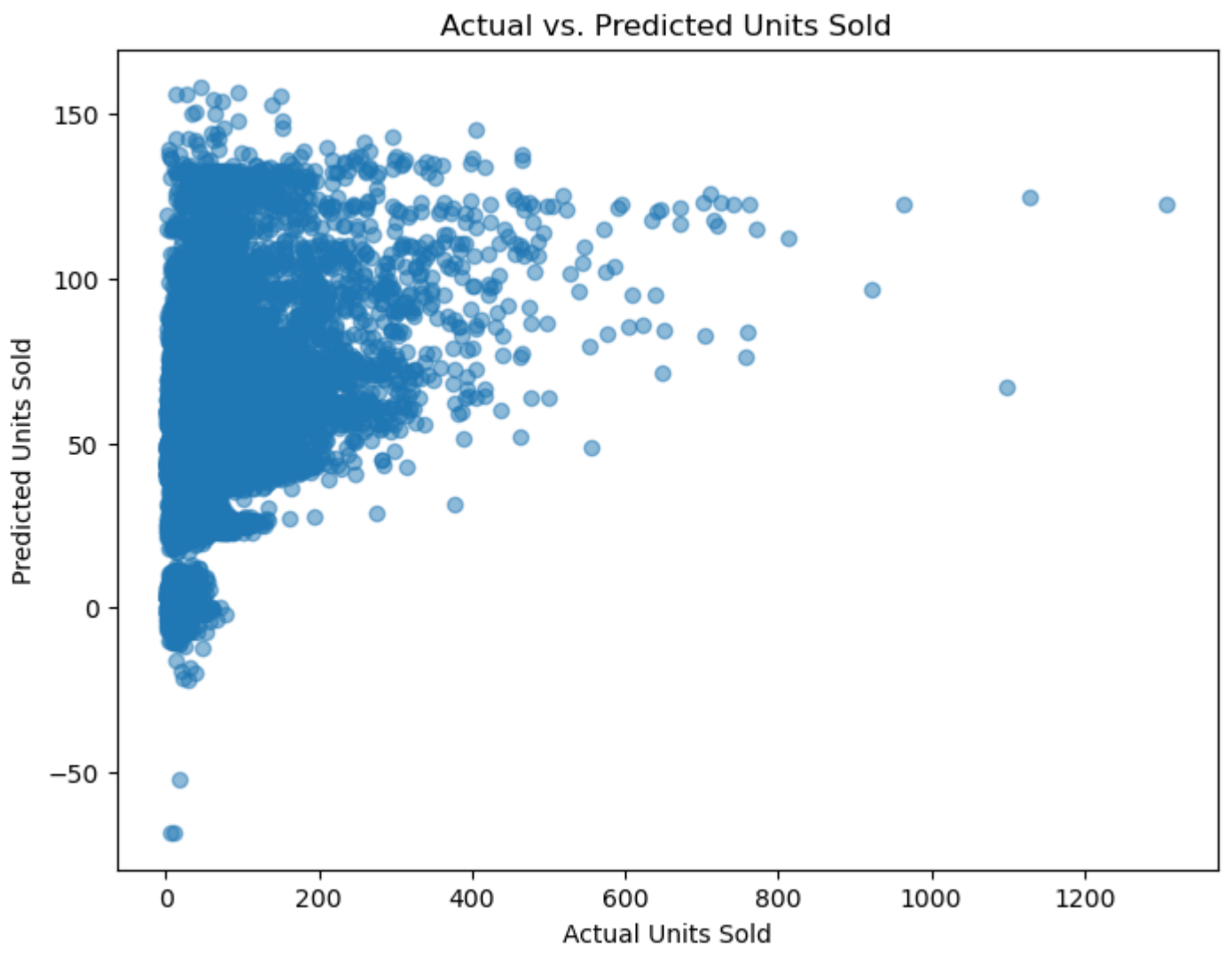
In [7]:
```
y_pred = model.predict(X_test)

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = mean_squared_error(y_test, y_pred, squared=False)
r2 = r2_score(y_test, y_pred)
```

In [8]:
```
print("Mean Absolute Error:", mae)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("R-squared:", r2)
```

```
Mean Absolute Error: 32.62649503895547
Mean Squared Error: 2780.7955492856986
Root Mean Squared Error: 52.733248992316966
R-squared: 0.15248746545511727
```

Visualization - Scatter plot of Actual vs. Predicted

In [9]:
```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.xlabel('Actual Units Sold')
plt.ylabel('Predicted Units Sold')
plt.title('Actual vs. Predicted Units Sold')
plt.show()
```



CONCLUSION

- The procedure encompassing feature engineering, model training, and dataset evaluation is a fundamental framework in machine learning.
- It involves data preparation, creating informative features, model selection, and rigorous evaluation. By iteratively fine-tuning the model and assessing its performance, this process ensures the development of effective predictive models that can be deployed in real-world applications, continuously monitored, and maintained to adapt to evolving data and requirements. Thorough documentation is vital to facilitate collaboration and transparency throughout the project.
- The procedure, which involves feature engineering, model training, and dataset evaluation, is fundamental to the development of effective product demand prediction models in machine learning.
- This systematic approach ensures that data is properly prepared, informative features are created, suitable models are selected, and rigorous evaluations are conducted. Iterative fine-tuning and performance assessments lead to robust predictive models, ready for deployment in real-world scenarios. Continuous monitoring and maintenance allow these models to adapt to changing data and demand requirements. Comprehensive documentation is essential for transparency and collaboration in the product demand prediction project.