Problem Statement

In this data science project, the objective is to predict product demand using machine learning. To achieve this goal, we will analyze the provided dataset containing information about product sales, including Store ID, Total Price, Base Price, and Units Sold. The primary challenge is to build a predictive model that can forecast product demand based on the available features.

Dataset Details

Dataset Information
ID: Unique identifier for each sales record.
Store ID: Identifies the store where the product was sold.
Total Price: The actual total price of the product.
Base Price: The base price of the product without any discounts.
Units Sold: The number of units of the product sold in each transaction.

Data Exploration

Before proceeding with the analysis and modeling, we will explore the dataset to gain a better understanding of its characteristics. This includes checking for missing values, data distribution, and statistical summary.

Data Preprocessing

To prepare the data for machine learning, we will need to perform various preprocessing tasks such as handling missing values, encoding categorical features (if any), and scaling/normalizing numerical features. This step is crucial to ensure the data is ready for model training and evaluation.

Feature Selection

Selecting the most relevant features for the prediction task is essential for model accuracy. We will identify and choose the features that have the most impact on product demand.

Model Building

In this phase, we will build and evaluate machine learning models for product demand prediction. Various regression algorithms will be considered, and their performance will be assessed using appropriate metrics (e.g., Mean Absolute Error, Root Mean Squared Error).

Model Evaluation

To determine the best model for predicting product demand, we will evaluate the models' performance on a test dataset and compare their results. Model evaluation will involve assessing accuracy, precision, recall, and other relevant metrics.

Importing Libraries

```
import matplotlib.pyplot as plt
import pandas as pd
import pylab as pl
import numpy as np
import sklearn
%matplotlib inline
```

Loading the Data Set

```
df = pd.read_csv("PoductDemand.csv")
print(df)
```

```
        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]
```

Exploring the Data Set

print(df.head())

```
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
```

print(df.tail())

```
 ID  Store ID  Total Price  Base Price  Units Sold
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
```

print(df.info())

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150150 entries, 0 to 150149
Data columns (total 5 columns):
 #   Column       Non-Null Count   Dtype
---  ------       -------------   -----
 0   ID           150150 non-null  int64
 1   Store ID     150150 non-null  int64
 2   Total Price  150149 non-null  float64
 3   Base Price   150150 non-null  float64
 4   Units Sold   150150 non-null  int64
dtypes: float64(2), int64(3)
memory usage: 5.7 MB
None
```

print(df.describe())

```
ID       Store ID    Total Price     Base Price  \
count  150150.000000  150150.000000  150149.000000  150150.000000
mean   106271.555504   9199.422511    206.626751    219.425927
std     61386.037861    615.591445    103.308516    110.961712
min         1.000000   8023.000000     41.325000     61.275000
25%     53111.250000   8562.000000    130.387500    133.237500
50%    106226.500000   9371.000000    198.075000    205.912500
75%    159452.750000   9731.000000    233.700000    234.412500
max    212644.000000   9984.000000    562.162500    562.162500

         Units Sold
count  150150.000000
mean       51.674206
std        60.207904
min         1.000000
25%        20.000000
50%        35.000000
75%        62.000000
max      2876.000000
```

Identifying null Values

print(df.isnull())

```
 ID  Store ID  Total Price  Base Price  Units Sold
0      False    False      False     False     False
1      False    False      False     False     False
2      False    False      False     False     False
3      False    False      False     False     False
4      False    False      False     False     False
...      ...      ...        ...       ...       ...
150145  False    False      False     False     False
150146  False    False      False     False     False
150147  False    False      False     False     False
150148  False    False      False     False     False
150149  False    False      False     False     False

[150150 rows x 5 columns]
```

```
c = df.isnull().sum()
print(c)
```

```
ID            0
Store ID      0
Total Price   1
Base Price    0
Units Sold    0
dtype: int64
```

```
print('Total Sum of null values in the Data set = ',c.sum())
```

```
Total Sum of null values in the Data set =  1
```

Data Preprocessing - Replacing the null values

```
df.drop_duplicates()
```

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 × 5 columns

df.fillna(0)

```
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 × 5 columns
```

Data Normalization

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df['Values_standardized'] = scaler.fit_transform(df[['Total Price']])
df['Values_standardized1'] = scaler.fit_transform(df[['Base Price']])
print(df)
```

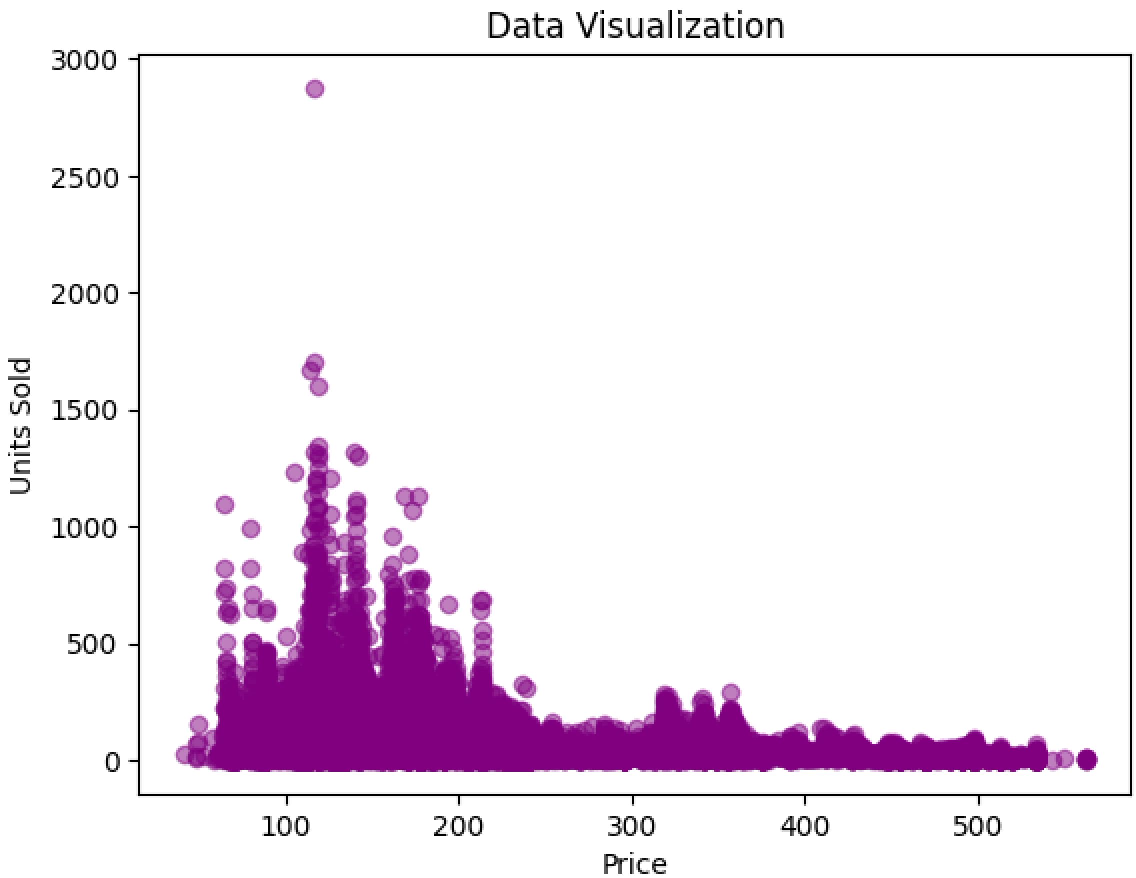
```
ID  Store ID  Total Price  Base Price  Units Sold  \
0         1     8091      99.0375    111.8625        20
1         2     8091      99.0375     99.0375        28
2         3     8091     133.9500    133.9500        19
3         4     8091     133.9500    133.9500        44
4         5     8091     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
```

```
       Values_standardized  Values_standardized1
0                -1.041440             -0.969377
1                -1.041440             -1.084958
2                -0.703495             -0.770322
3                -0.703495             -0.770322
4                -0.634526             -0.706110
...                    ...                   ...
150145            0.282754              0.147904
150146            0.282754              0.147904
150147            1.462113              2.382466
150148           -0.627629             -0.250208
150149            0.268960              0.135061

[150150 rows x 7 columns]
```

Data Visualisation

```
plt.scatter(df['Total Price'], df['Units Sold'], alpha=0.5, color='purple')
plt.title('Data Visualization')
plt.xlabel('Price')
plt.ylabel('Units Sold')
plt.show()
```



Data Analysis using different models

Simple Linear Regression

cdf = df[['Total Price','Base Price','Units Sold']]
cdf.head(9)

	Total Price	Base Price	Units Sold
0	99.0375	111.8625	20
1	99.0375	99.0375	28
2	133.9500	133.9500	19
3	133.9500	133.9500	44
4	141.0750	141.0750	52
5	227.2875	227.2875	18
6	327.0375	327.0375	47
7	210.9000	210.9000	50
8	190.2375	234.4125	82

```
test = test.dropna()
y_hat= regr.predict(test[['Total Price', 'Base Price']])
x = np.asanyarray(test[['Total Price', 'Base Price']])
y = np.asanyarray(test[['Units Sold']])
print("Mean Squared Error (MSE) : %.2f"
% np.mean((y_hat - y) ** 2))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % regr.score(x, y))
```

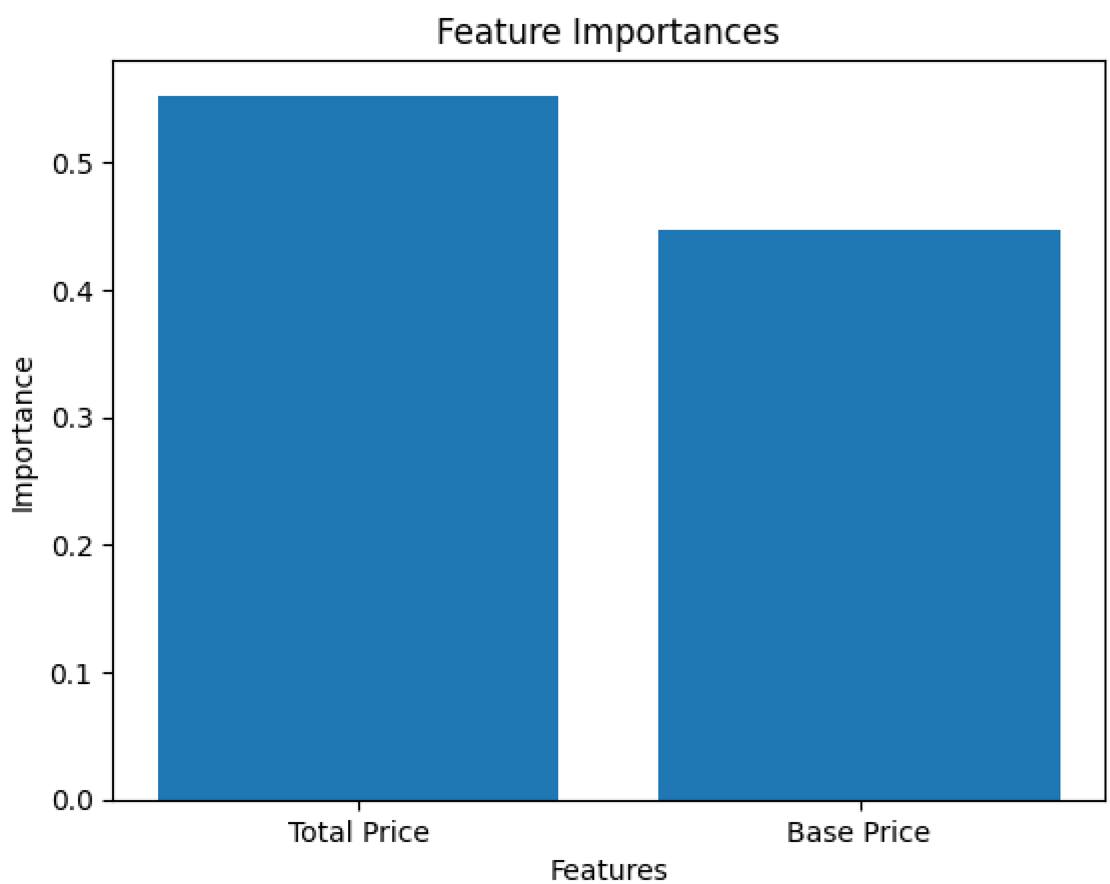
```
Mean Squared Error (MSE) : 2948.93
Variance score: 0.15
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LinearRegression was fitted without feature names
  warnings.warn(
```

Random forest Algorithm

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
# Load your dataset from the CSV file
data = pd.read_csv("PoductDemand.csv") # Adjust the
# Handle missing values by filling with the mean
data = data.fillna(data.mean())
features = data[['Total Price', 'Base Price']]
target = data['Units Sold']
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42)
# Create and train the Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = rf_model.predict(X_test)
# Calculate model performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
# Visualize the feature importances
feature_importances = rf_model.feature_importances_
plt.bar(features.columns, feature_importances)
plt.xlabel("Features")
plt.ylabel("Importance")
plt.title("Feature Importances")
plt.show()
```

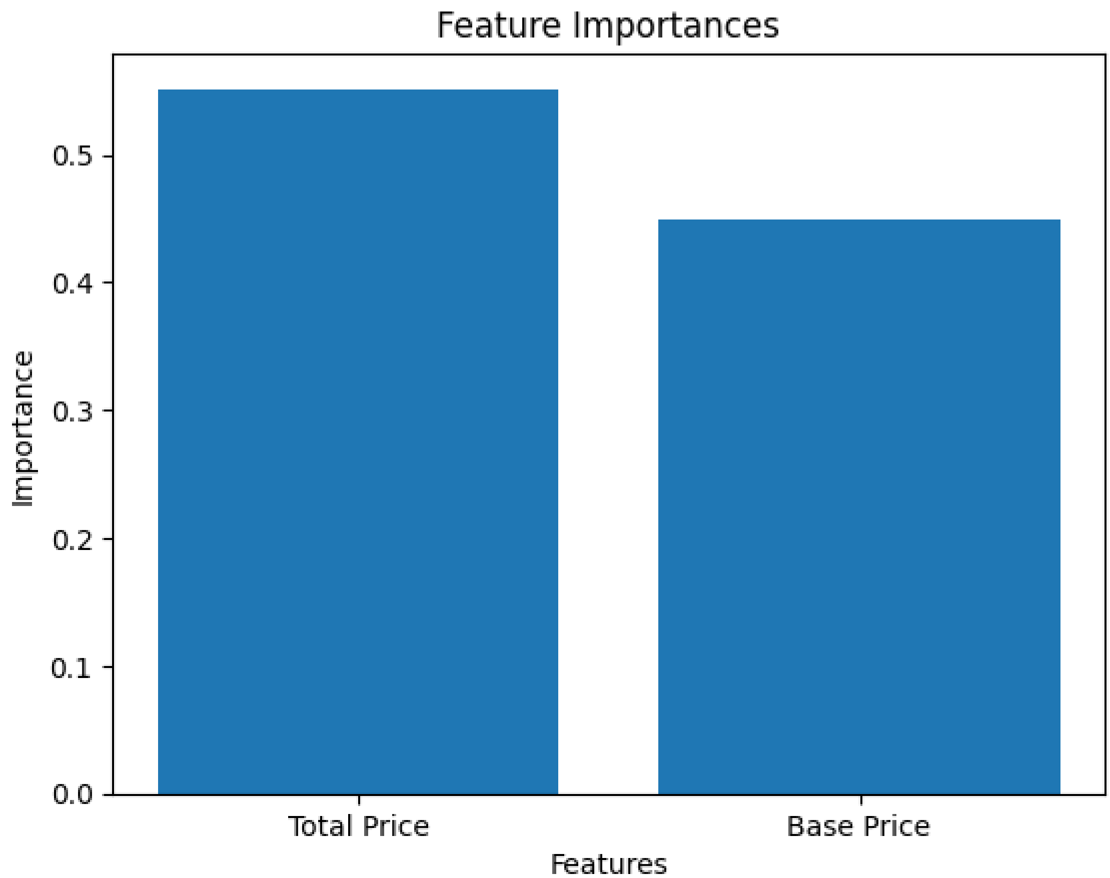
Mean Squared Error: 1885.60
R-squared (R2) Score: 0.43



Regression with the Gradient Boosting algorithm

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
# Load your dataset from the CSV file
data = pd.read_csv("PoductDemand.csv")
# Handle missing values by filling with the mean
data = data.fillna(data.mean())
# Select the relevant features (SO2 and NO2) and the target variable
features = data[['Total Price', 'Base Price']]
target = data['Units Sold']
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=42)
# Create and train the Gradient Boosting model
gb_model = GradientBoostingRegressor(n_estimators=100,
random_state=42)
gb_model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = gb_model.predict(X_test)
# Calculate model performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared (R2) Score: {r2:.2f}")
# Visualize the feature importances
feature_importances = gb_model.feature_importances_
plt.bar(features.columns, feature_importances)
plt.xlabel("Features")
plt.ylabel("Importance")
plt.title("Feature Importances")
plt.show()
```

Mean Squared Error: 1987.34
R-squared (R2) Score: 0.39



Model Tuning

Fine-tuning of the selected model(s) will be carried out to optimize their hyperparameters and enhance their predictive power.

Model Deployment

Once a satisfactory model is developed, we will deploy it to make real-time predictions on new data, allowing the business to anticipate product demand more effectively.

Conclusion

This data science project aims to provide a solution for product demand prediction, which can have significant implications for inventory management and sales optimization. The final outcome of the project will be a machine learning model capable of forecasting product demand based on the given dataset.