CSCE 5300

Introduction to Big Data and Data Science

Walmart Data Analytics: Data Exploration, Visualization, and Prediction

Authors:

Manpreet Singh , Bhawick Ghutla, , Reuben Lilo Jnr, Aesaan F S Mohammed and Mahmood A Rashid

<u>Project Group - 13:</u>

Eeshwar Vannemreddy

Nitin Narayanan Kokkoori

Sharikhirfaan Mohammed

Sai Bhavani Munnangi

Vivek Reddy Nadavaluru

I. ABSTRACT

The retail industry has undergone a significant transformation in recent years, largely due to the growth and adoption of technology. In today's world, data is king, and retailers are no exception. With the ever-increasing amount of data being generated, retailers must find ways to harness it and make it work for them. This is where data analysis and machine learning come into play.

Traditional sales analysis methods like manual data entry and spreadsheet-based analysis are slow, prone to errors, and not scalable for large volumes of data. The emergence of machine learning algorithms has become a key tool for analyzing sales data as they can automatically learn patterns and relationships in data to make accurate predictions and recommendations. Machine learning can be used to forecast sales trends, predict customer behavior, identify key drivers of sales performance, and optimize sales strategies. By using machine learning, businesses can gain a deeper understanding of their sales data and make informed decisions that drive growth and profitability.

The project aims to provide a comprehensive solution for retailers to identify key drivers impacting sales and predict sales trends. It uses machine learning algorithms to analyze large datasets and predict sales based on various factors. By analyzing sales trends and identifying key drivers, retailers can make informed decisions about which departments to focus on and which promotions to run to meet their sales and marketing goals.

The project involves the use of advanced analytics and machine learning techniques to gain insights into customer behavior and preferences, as well as the impact of various external factors on sales. By combining historical data with real-time data, we can build predictive models that can accurately forecast future sales trends.

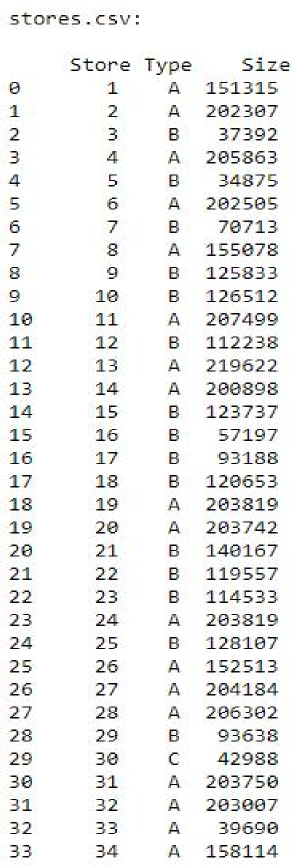
The outcome of this project will enable retailers to optimize their sales strategies and promotions to maximize revenue and profitability. With the ability to predict sales trends and identify key drivers, retailers can make informed decisions about where to focus their efforts and resources, ultimately leading to greater success in the competitive retail market.

II. INPUT DATASET

The algorithm uses the below 4 datasets to predict the sales analysis.

1. stores.csv
2. train.csv
3. test.csv
4. features.csv

1. **STORES.CSV:** This dataset contains information about the 45 Walmart stores that are part of the problem. The file contains three columns:
    - **Store**: A numerical identifier for the store, ranging from 1 to 45.
    - **Type**: A categorical variable indicating the type of store. There are three types of stores in this dataset, labeled A, B, and C.
    - **Size**: The size of the store, measured in square feet.

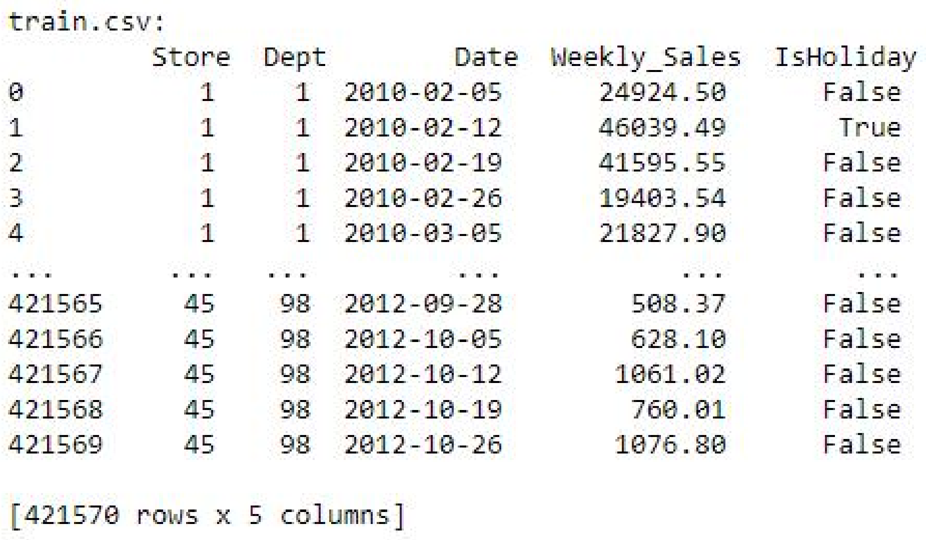


```
stores.csv:

     Store Type    Size
0        1    A  151315
1        2    A  202307
2        3    B   37392
3        4    A  205863
4        5    B   34875
5        6    A  202505
6        7    B   70713
7        8    A  155078
8        9    B  125833
9       10    B  126512
10      11    A  207499
11      12    B  112238
12      13    A  219622
13      14    A  200898
14      15    B  123737
15      16    B   57197
16      17    B   93188
17      18    B  120653
18      19    A  203819
19      20    A  203742
20      21    B  140167
21      22    B  119557
22      23    B  114533
23      24    A  203819
24      25    B  128107
25      26    A  152513
26      27    A  204184
27      28    A  206302
28      29    B   93638
29      30    C   42988
30      31    A  203750
31      32    A  203007
32      33    A   39690
33      34    A  158114
```

2. **train.csv**: This dataset is a historical training dataset that contains information about weekly sales for each department in each store. The dataset covers the period from 2010-02-05 to 2012-11-01. The file contains the following columns:

   - **Store**: An integer representing the store number. There are 45 stores in total, numbered from 1 to 45.
   - **Dept**: An integer representing the department number. Each store contains multiple departments, numbered from 1 to 99.
   - **Date**: A string representing the date of the week in the format of YYYY-MM-DD.
   - **Weekly_Sales**: A float representing the total weekly sales for the given department in the given store.
   - **IsHoliday**: A Boolean value indicating whether the week contains a special holiday. The holidays include the Super Bowl, Labor Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks.

There are 421,570 rows in the train.csv file, each representing a unique combination of store, department, and date.

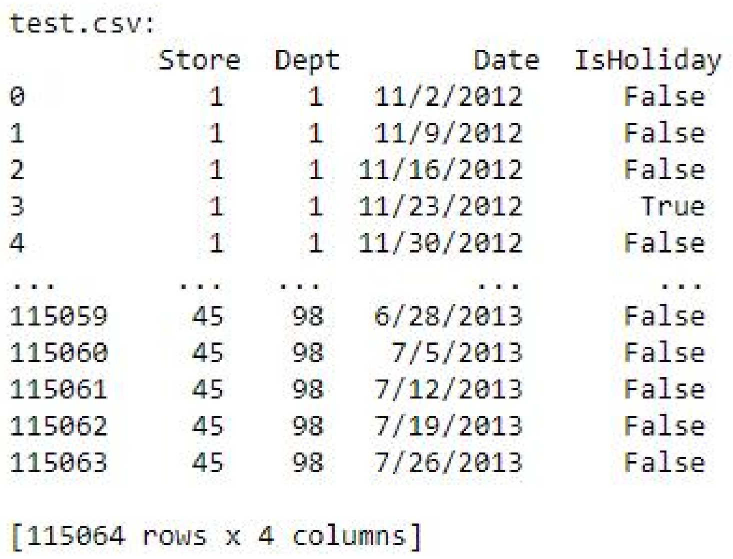


```
train.csv:
        Store  Dept        Date  Weekly_Sales  IsHoliday
0           1     1  2010-02-05      24924.50      False
1           1     1  2010-02-12      46039.49       True
2           1     1  2010-02-19      41595.55      False
3           1     1  2010-02-26      19403.54      False
4           1     1  2010-03-05      21827.90      False
...       ...   ...         ...           ...        ...
421565     45    98  2012-09-28        508.37      False
421566     45    98  2012-10-05        628.10      False
421567     45    98  2012-10-12       1061.02      False
421568     45    98  2012-10-19        760.01      False
421569     45    98  2012-10-26       1076.80      False

[421570 rows x 5 columns]
```

3. **test.csv**: The test.csv file is similar to the train.csv file, but with the weekly sales data removed. It contains data for the same 45 Walmart stores and their departments, but covers a later period, from 2012to 2013.

   The file has 115,064 rows, with each row representing a unique combination of store, department, and week. It has four columns, including:

   - **Store**: the store number (1-45)
   - **Dept**: the department number (1-98)
   - **Date**: the week ending date (in mm/dd/yyyy format)
   - **IsHoliday**: whether the week contains a special holiday (True or False)



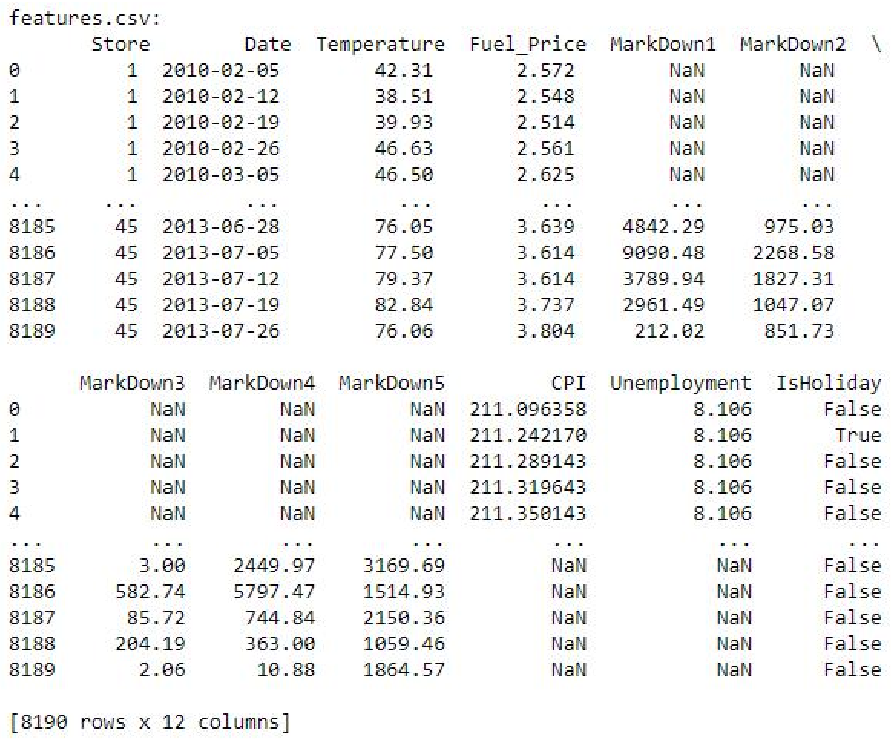
```
test.csv:
        Store  Dept        Date  IsHoliday
0           1     1   11/2/2012      False
1           1     1   11/9/2012      False
2           1     1  11/16/2012      False
3           1     1  11/23/2012       True
4           1     1  11/30/2012      False
...       ...   ...         ...        ...
115059     45    98   6/28/2013      False
115060     45    98    7/5/2013      False
115061     45    98   7/12/2013      False
115062     45    98   7/19/2013      False
115063     45    98   7/26/2013      False

[115064 rows x 4 columns]
```

4. **features.csv**: This file contains additional data related to the stores, departments, and dates in the training and testing datasets. Specifically, it contains the following columns:

- **Store**: An integer representing the store number (1-45).
- **Date**: The date (in yyyy-mm-dd format) of the week that the row's data corresponds to.
- **Temperature**: The average temperature (in Fahrenheit) on the day of the week that the row's data corresponds to.
- **Fuel_Price**: The cost of fuel (in dollars per gallon) on the day of the week that the row's data corresponds to.
- **MarkDown1-5:** Five columns representing any markdowns (discounts) that the store offered on the day of the week that the row's

data corresponds to. These values may be missing if the store did not offer any markdowns.

- **CPI**: The consumer price index (CPI) on the day of the week that the row's data corresponds to.



```
features.csv:
      Store        Date  Temperature  Fuel_Price  MarkDown1  MarkDown2  \
0         1  2010-02-05        42.31       2.572        NaN        NaN
1         1  2010-02-12        38.51       2.548        NaN        NaN
2         1  2010-02-19        39.93       2.514        NaN        NaN
3         1  2010-02-26        46.63       2.561        NaN        NaN
4         1  2010-03-05        46.50       2.625        NaN        NaN
...     ...         ...          ...         ...        ...        ...
8185     45  2013-06-28        76.05       3.639    4842.29     975.03
8186     45  2013-07-05        77.50       3.614    9090.48    2268.58
8187     45  2013-07-12        79.37       3.614    3789.94    1827.31
8188     45  2013-07-19        82.84       3.737    2961.49    1047.07
8189     45  2013-07-26        76.06       3.804     212.02     851.73

      MarkDown3  MarkDown4  MarkDown5         CPI  Unemployment  IsHoliday
0           NaN        NaN        NaN  211.096358         8.106      False
1           NaN        NaN        NaN  211.242170         8.106       True
2           NaN        NaN        NaN  211.289143         8.106      False
3           NaN        NaN        NaN  211.319643         8.106      False
4           NaN        NaN        NaN  211.350143         8.106      False
...         ...        ...        ...         ...           ...        ...
8185       3.00    2449.97    3169.69         NaN           NaN      False
8186     582.74    5797.47    1514.93         NaN           NaN      False
8187      85.72     744.84    2150.36         NaN           NaN      False
8188     204.19     363.00    1059.46         NaN           NaN      False
8189       2.06      10.88    1864.57         NaN           NaN      False

[8190 rows x 12 columns]
```

III. INTRODUCTION

In today's rapidly changing business environment, companies face a multitude of challenges in trying to stay ahead of their competitors. One of the key challenges is effectively managing and analyzing the vast amounts of sales data that businesses generate daily. To make sense of this data and gain valuable insights into customer behavior, market trends, and other key metrics, companies need powerful tools and techniques that can handle large volumes of data, identify patterns and trends, and provide actionable insights.

In this project, we have analyzed the data sets of one of the world's largest retailers, Walmart Store, to determine the business drivers and predict which departments are affected by different scenarios, such as temperature, fuel price, and holidays, and their impact on sales at stores' various locations. By using big data applications, we can enable these retail organizations to use prior years' data to better forecast and predict the coming year's sales. It also enables retailers with valuable and analytical insights, especially determining customers with desired products at desired times in a particular store at different geographical locations.

Our project presents a machine learning algorithm for sales analysis, which aims to predict the sales revenue of a company based on various factors such as advertising spend, promotional activities, and customer demographics. The algorithm presented uses the Random Forest Regressor (RFR) algorithm, which is a popular machine learning algorithm used for regression tasks. The RFR algorithm is an ensemble learning method that combines multiple decision trees to create a strong learner. The algorithm works by randomly selecting a subset of features and training decision trees on each subset, then combining the predictions of each tree to make the final prediction.

The dataset used for training and testing the algorithm is a sales dataset, which contains information about the sales revenue of a company over a period. The dataset has several features, including the amount spent on advertising, the number of promotional activities, and the demographic profile of customers.

In conclusion, our project demonstrates the use of the Random Forest Regressor algorithm for sales analysis, which can help companies predict their sales revenue based on various factors. The algorithm achieved high accuracy on the sales dataset, which shows its potential for real-world applications. The code can be used as a starting point for building more complex machine learning models for sales analysis.

IV. DESCRIPTION OF THE MODEL APPROACH

Our approach implements a regression model to predict the weekly sales of different stores. The model uses various approaches to preprocess the data, visualize the data, and build and evaluate the regression models.

Phases in the model approach:

1. **Data Preprocessing**: Reading input data from CSV files, merging data from multiple files, dropping columns with missing values and negative or zero sales, encoding categorical variables using label encoding, and creating new columns for year, month, and week.

2. **Feature Engineering**: Creating new columns for year, month, and week from the date column and encoding categorical variables using label encoding.
3. **Data Visualization**: Creating various charts to explore relationships between variables, including a pie chart, a correlation heatmap, bar plots, line plots, and point plots.
4. **Model Building**: Training three regression models using scikit-learn library: Random Forest Regressor, Gradient Boosting Regressor, and Decision Tree Regressor.
5. **Hyperparameters Tuning**: Choosing hyperparameters based on trial and error and evaluating model performance using cross-validation.
6. **Model Evaluation**: Evaluating model performance using R-squared score and Mean Absolute Error (MAE) for both the test set and cross-validation.

Let us discuss the phases in detail with respect to our predictive model.

1. Data Preprocessing:

The first step is to read the input data from the csv files. The data is divided into four files: 'stores.csv', 'train.csv', 'test.csv', and 'features.csv'. The 'train.csv' file contains the training data, which is merged with 'features.csv' and 'stores.csv' using the 'merge' function to get additional features. The 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', and 'MarkDown5' columns, which have many missing values, are dropped. The rows with negative or zero sales are also removed. New columns for year, month, and week are created by converting the 'Date' column to datetime format.

Categorical variables like 'IsHoliday' and 'Type' are encoded using label encoding. Label encoding is a technique used to convert categorical variables into numerical values so that they can be used as input in our model.

2. Feature Engineering:

In our approach, we created new columns for year, month, and week from the 'Date' column and encoded categorical variables using label encoding. These techniques helped to improve the predictive power of our regression models.

3. Data Visualization:

The data is visualized using various charts to get insights and explore the relationships between the variables. A pie chart is used to show the distribution of store types. A correlation heatmap is plotted to see the correlation between different features. Bar plots are used to visualize the variation of fuel prices and weekly sales by store. A line plot is used to show the variation of unemployment by store. A point plot is used to show the total weekly sales for each department. Another line plot is used to show the total weekly sales for each month in each year.

4. Model Building:

The data is split into training and testing sets using the 'train_test_split' function. The input variables and target variable are defined for the training and testing data. Three regression models are built using the scikit-learn library: Random Forest Regressor, Gradient Boosting Regressor, and Decision Tree Regressor.

The Random Forest Regressor and Gradient Boosting Regressor models are used because they have been shown to perform well on similar prediction tasks. The Decision Tree Regressor is also used as a baseline model for comparison. The hyperparameters for each model are chosen based on trial and error and may be further optimized using more sophisticated techniques such as grid search or Bayesian optimization.

- The Random Forest Regressor model is trained with 100 estimators and a maximum depth of 10.
- The Gradient Boosting Regressor model is trained with 100 estimators, a maximum depth of 5, and a learning rate of 0.1.
- The Decision Tree Regressor model is trained with a maximum depth of 10.

5. Hyperparameters Tuning:

The hyperparameters for each model were chosen based on trial and error, and the performance of the models was evaluated using cross-validation.

6. Model Evaluation:

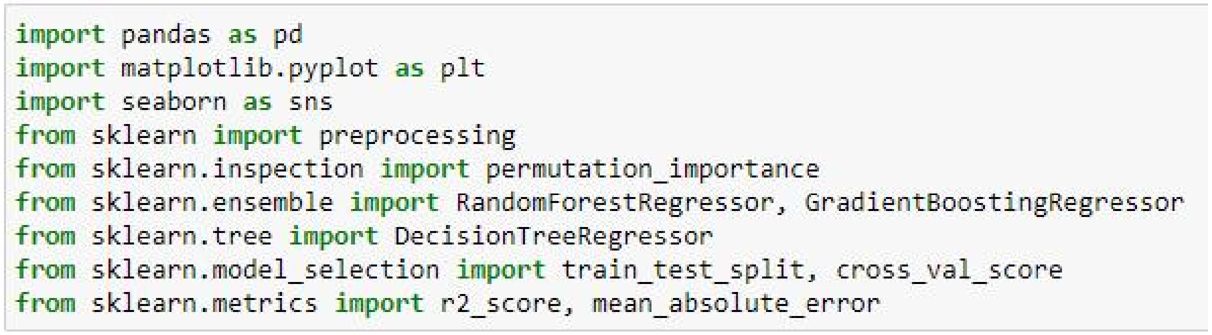
The performance of the models is evaluated using various metrics. The R-squared score and Mean Absolute Error (MAE) are used to evaluate the performance of the models on the test set. Cross-validation is used to evaluate the performance of the models on the training set. The R-squared score and MAE are calculated for each fold, and the mean value of these scores is used as the final evaluation metric.

The performance of the models will be given in the output. The R-squared score and MAE are reported for the test set and cross-validation.

Overall, the code implements a complete pipeline for data preprocessing, exploratory data analysis, model training, and evaluation for predicting weekly sales for different stores and departments. It provides a starting point for more advanced machine learning tasks and can be further extended and optimized based on the specific requirements of the problem at hand.

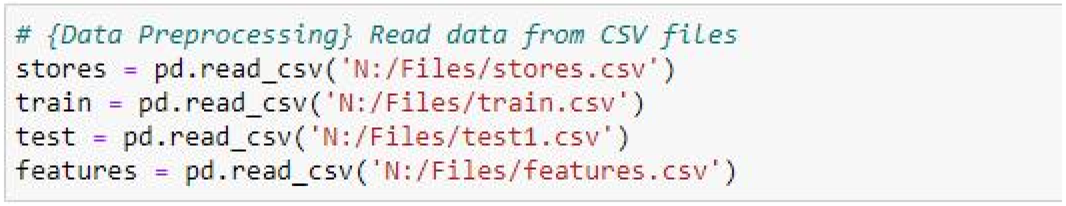
V. CODE SCREENSHOTS:

1. Importing the required modules and packages

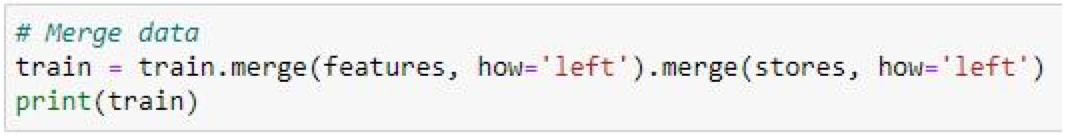


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.inspection import permutation_importance
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import r2_score, mean_absolute_error
```

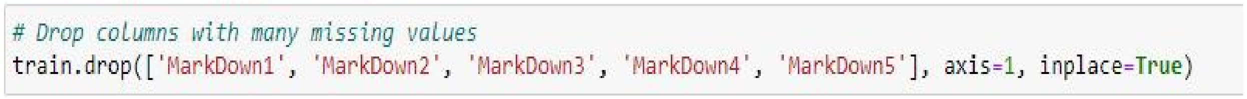
2. Data preprocessing



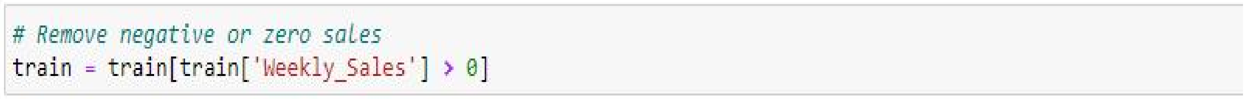
```python
# {Data Preprocessing} Read data from CSV files
stores = pd.read_csv('N:/Files/stores.csv')
train = pd.read_csv('N:/Files/train.csv')
test = pd.read_csv('N:/Files/test1.csv')
features = pd.read_csv('N:/Files/features.csv')
```



```python
# Merge data
train = train.merge(features, how='left').merge(stores, how='left')
print(train)
```

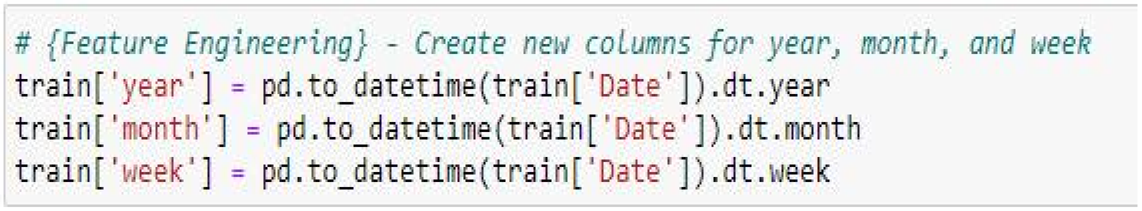


```python
# Drop columns with many missing values
train.drop(['MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5'], axis=1, inplace=True)
```



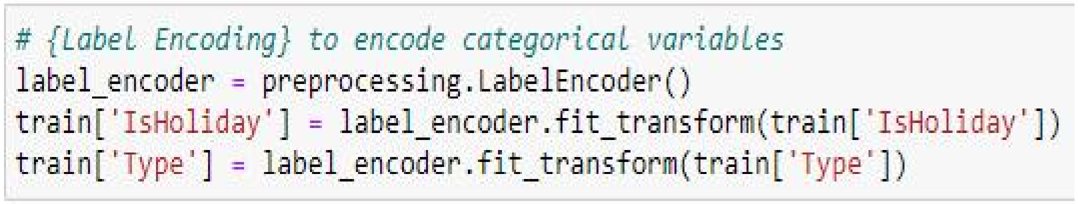
```python
# Remove negative or zero sales
train = train[train['Weekly_Sales'] > 0]
```

3. Feature engineering



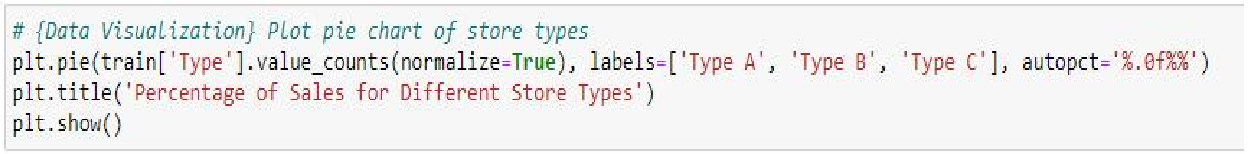
```python
# {Feature Engineering} - Create new columns for year, month, and week
train['year'] = pd.to_datetime(train['Date']).dt.year
train['month'] = pd.to_datetime(train['Date']).dt.month
train['week'] = pd.to_datetime(train['Date']).dt.week
```

4. Label Encoding

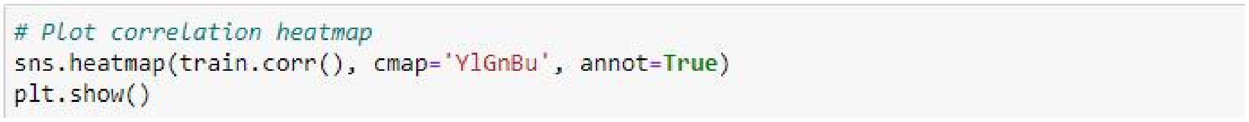


```python
# {Label Encoding} to encode categorical variables
label_encoder = preprocessing.LabelEncoder()
train['IsHoliday'] = label_encoder.fit_transform(train['IsHoliday'])
train['Type'] = label_encoder.fit_transform(train['Type'])
```

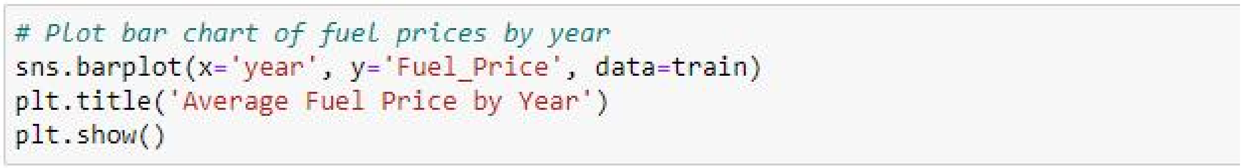
5. Data Visualization



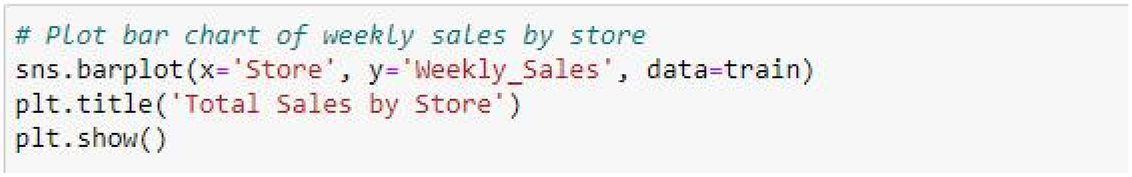
```python
# {Data Visualization} Plot pie chart of store types
plt.pie(train['Type'].value_counts(normalize=True), labels=['Type A', 'Type B', 'Type C'], autopct='%.0f%%')
plt.title('Percentage of Sales for Different Store Types')
plt.show()
```



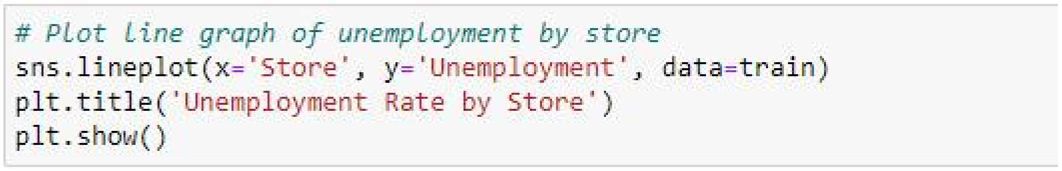
```python
# Plot correlation heatmap
sns.heatmap(train.corr(), cmap='YlGnBu', annot=True)
plt.show()
```



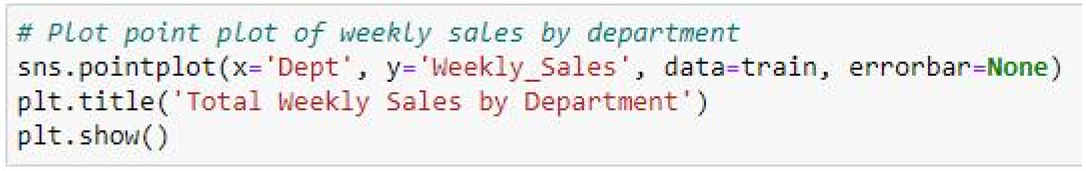
```python
# Plot bar chart of fuel prices by year
sns.barplot(x='year', y='Fuel_Price', data=train)
plt.title('Average Fuel Price by Year')
plt.show()
```



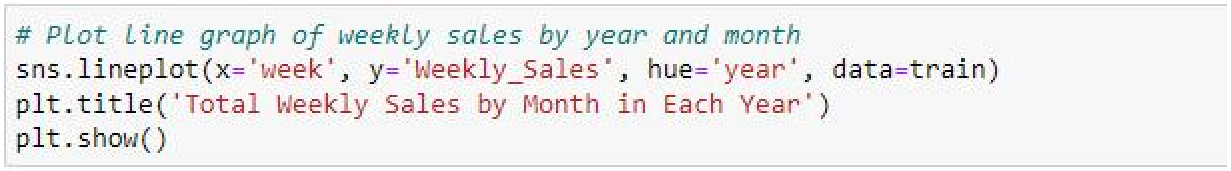
```python
# Plot bar chart of weekly sales by store
sns.barplot(x='Store', y='Weekly_Sales', data=train)
plt.title('Total Sales by Store')
plt.show()
```



```python
# Plot line graph of unemployment by store
sns.lineplot(x='Store', y='Unemployment', data=train)
plt.title('Unemployment Rate by Store')
plt.show()
```

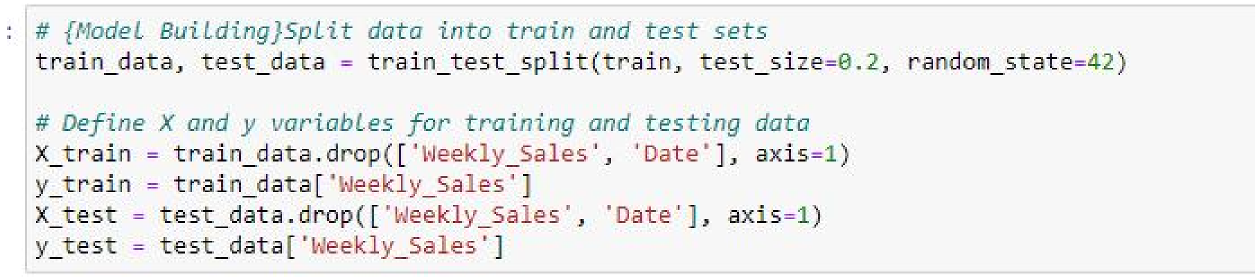


```python
# Plot point plot of weekly sales by department
sns.pointplot(x='Dept', y='Weekly_Sales', data=train, errorbar=None)
plt.title('Total Weekly Sales by Department')
plt.show()
```



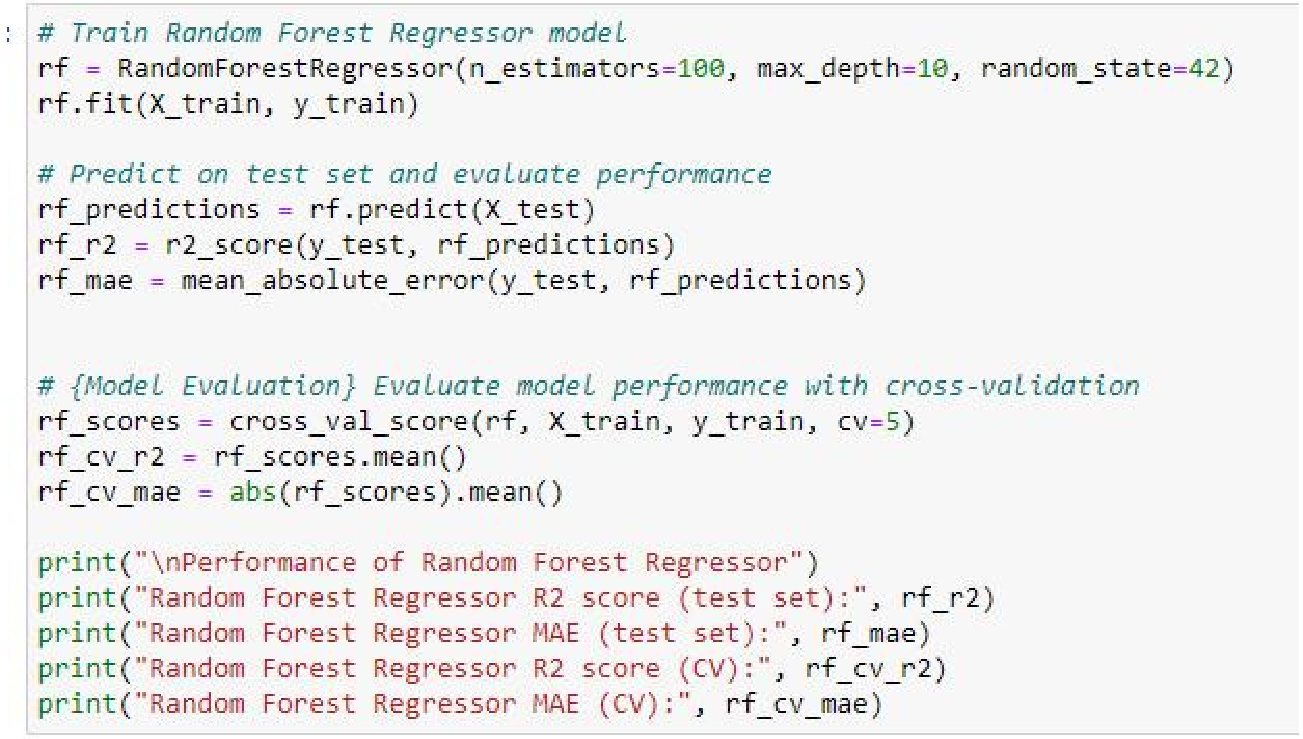
```python
# Plot line graph of weekly sales by year and month
sns.lineplot(x='week', y='Weekly_Sales', hue='year', data=train)
plt.title('Total Weekly Sales by Month in Each Year')
plt.show()
```

6. Model Building



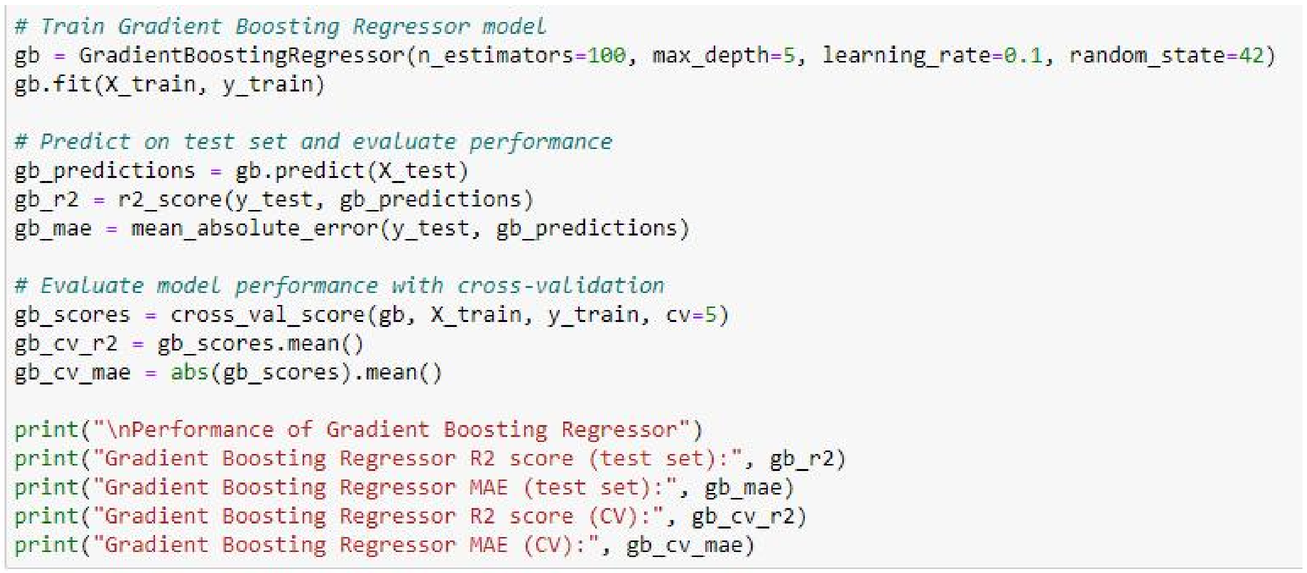
```
# {Model Building}Split data into train and test sets
train_data, test_data = train_test_split(train, test_size=0.2, random_state=42)

# Define X and y variables for training and testing data
X_train = train_data.drop(['Weekly_Sales', 'Date'], axis=1)
y_train = train_data['Weekly_Sales']
X_test = test_data.drop(['Weekly_Sales', 'Date'], axis=1)
y_test = test_data['Weekly_Sales']
```

7. Training the data by Random Regressor Model and model evaluation and printing the results:



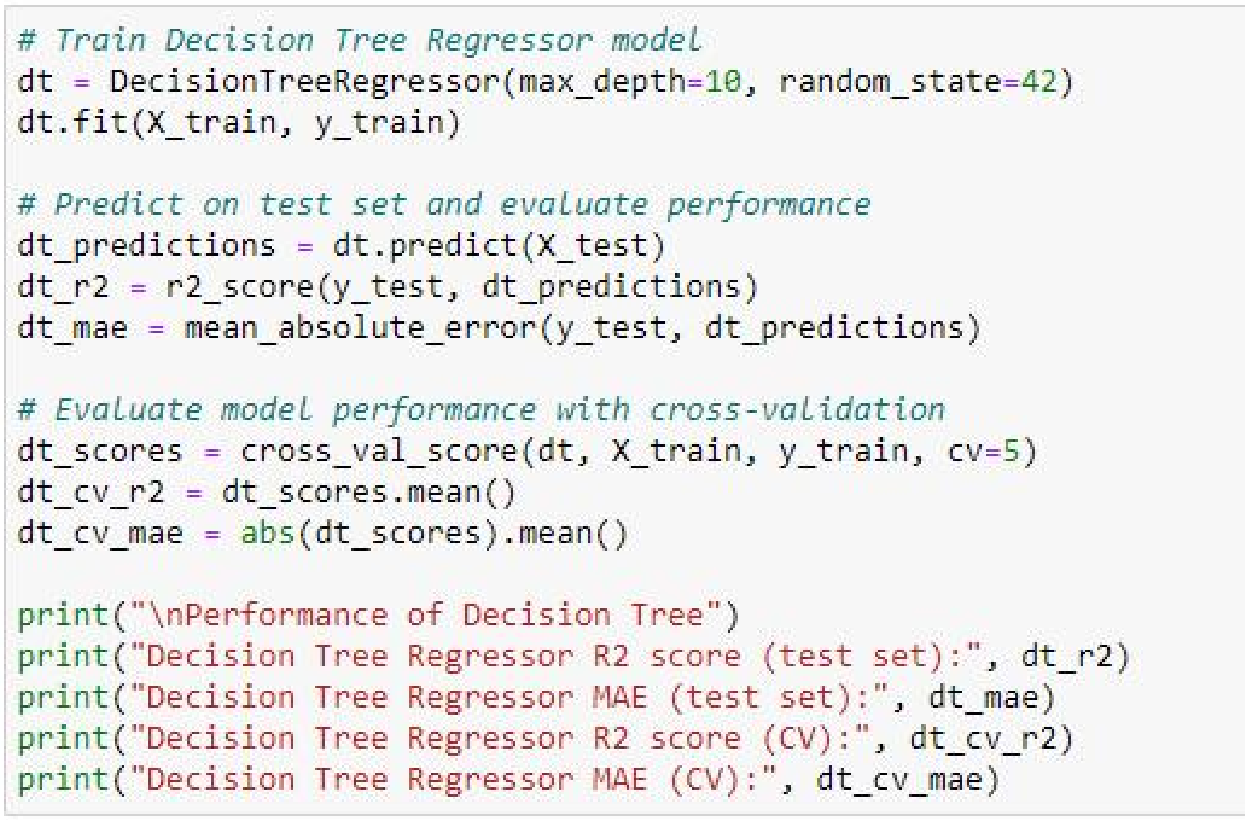
```
# Train Random Forest Regressor model
rf = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42)
rf.fit(X_train, y_train)

# Predict on test set and evaluate performance
rf_predictions = rf.predict(X_test)
rf_r2 = r2_score(y_test, rf_predictions)
rf_mae = mean_absolute_error(y_test, rf_predictions)


# {Model Evaluation} Evaluate model performance with cross-validation
rf_scores = cross_val_score(rf, X_train, y_train, cv=5)
rf_cv_r2 = rf_scores.mean()
rf_cv_mae = abs(rf_scores).mean()

print("\nPerformance of Random Forest Regressor")
print("Random Forest Regressor R2 score (test set):", rf_r2)
print("Random Forest Regressor MAE (test set):", rf_mae)
print("Random Forest Regressor R2 score (CV):", rf_cv_r2)
print("Random Forest Regressor MAE (CV):", rf_cv_mae)
```

8. Training the data by Gradient Boosting Regressor Model and model evaluation & printing the results:



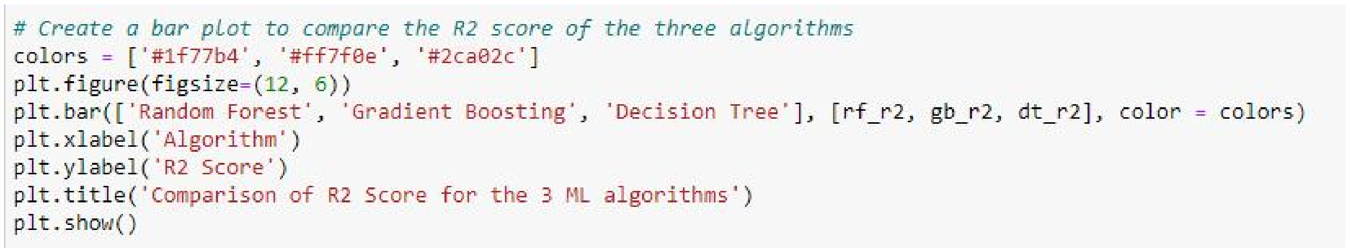
```
# Train Gradient Boosting Regressor model
gb = GradientBoostingRegressor(n_estimators=100, max_depth=5, learning_rate=0.1, random_state=42)
gb.fit(X_train, y_train)

# Predict on test set and evaluate performance
gb_predictions = gb.predict(X_test)
gb_r2 = r2_score(y_test, gb_predictions)
gb_mae = mean_absolute_error(y_test, gb_predictions)

# Evaluate model performance with cross-validation
gb_scores = cross_val_score(gb, X_train, y_train, cv=5)
gb_cv_r2 = gb_scores.mean()
gb_cv_mae = abs(gb_scores).mean()

print("\nPerformance of Gradient Boosting Regressor")
print("Gradient Boosting Regressor R2 score (test set):", gb_r2)
print("Gradient Boosting Regressor MAE (test set):", gb_mae)
print("Gradient Boosting Regressor R2 score (CV):", gb_cv_r2)
print("Gradient Boosting Regressor MAE (CV):", gb_cv_mae)
```

9. Training the data by Decision Tree Model and model evaluation & printing the results:



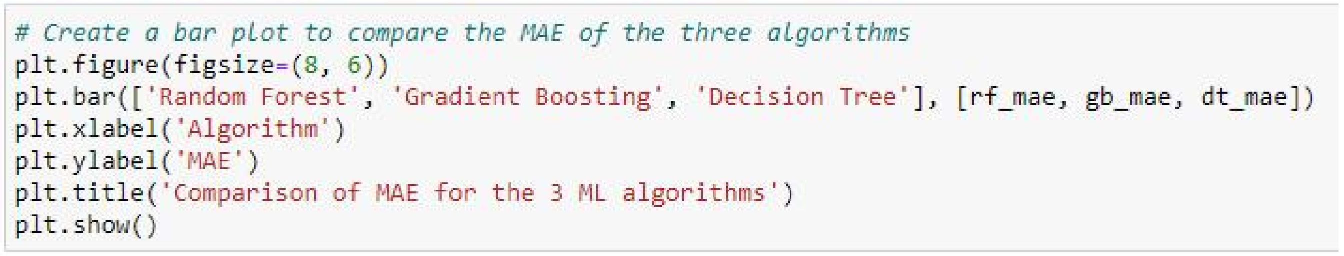
```
# Train Decision Tree Regressor model
dt = DecisionTreeRegressor(max_depth=10, random_state=42)
dt.fit(X_train, y_train)

# Predict on test set and evaluate performance
dt_predictions = dt.predict(X_test)
dt_r2 = r2_score(y_test, dt_predictions)
dt_mae = mean_absolute_error(y_test, dt_predictions)

# Evaluate model performance with cross-validation
dt_scores = cross_val_score(dt, X_train, y_train, cv=5)
dt_cv_r2 = dt_scores.mean()
dt_cv_mae = abs(dt_scores).mean()

print("\nPerformance of Decision Tree")
print("Decision Tree Regressor R2 score (test set):", dt_r2)
print("Decision Tree Regressor MAE (test set):", dt_mae)
print("Decision Tree Regressor R2 score (CV):", dt_cv_r2)
print("Decision Tree Regressor MAE (CV):", dt_cv_mae)
```

10. Comparing the R2 scores of all the 3 algorithms:



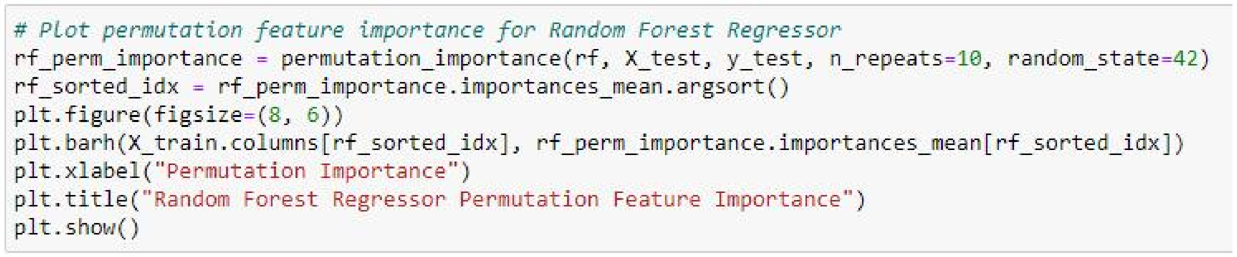
```python
# Create a bar plot to compare the R2 score of the three algorithms
colors = ['#1f77b4', '#ff7f0e', '#2ca02c']
plt.figure(figsize=(12, 6))
plt.bar(['Random Forest', 'Gradient Boosting', 'Decision Tree'], [rf_r2, gb_r2, dt_r2], color = colors)
plt.xlabel('Algorithm')
plt.ylabel('R2 Score')
plt.title('Comparison of R2 Score for the 3 ML algorithms')
plt.show()
```

11. Comparing the MAE scores of 3 algorithms:

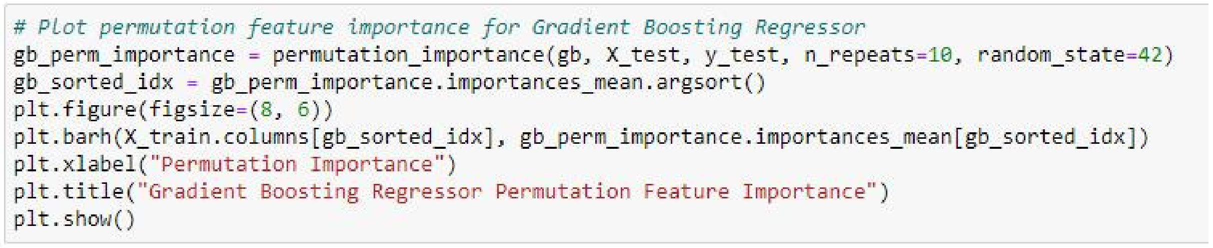


```python
# Create a bar plot to compare the MAE of the three algorithms
plt.figure(figsize=(8, 6))
plt.bar(['Random Forest', 'Gradient Boosting', 'Decision Tree'], [rf_mae, gb_mae, dt_mae])
plt.xlabel('Algorithm')
plt.ylabel('MAE')
plt.title('Comparison of MAE for the 3 ML algorithms')
plt.show()
```

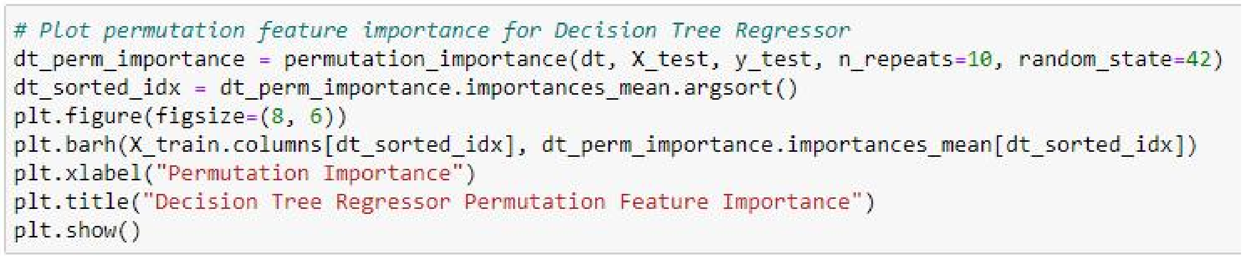
12. Showing the features importance in each algorithm:



```python
# Plot permutation feature importance for Random Forest Regressor
rf_perm_importance = permutation_importance(rf, X_test, y_test, n_repeats=10, random_state=42)
rf_sorted_idx = rf_perm_importance.importances_mean.argsort()
plt.figure(figsize=(8, 6))
plt.barh(X_train.columns[rf_sorted_idx], rf_perm_importance.importances_mean[rf_sorted_idx])
plt.xlabel("Permutation Importance")
plt.title("Random Forest Regressor Permutation Feature Importance")
plt.show()
```

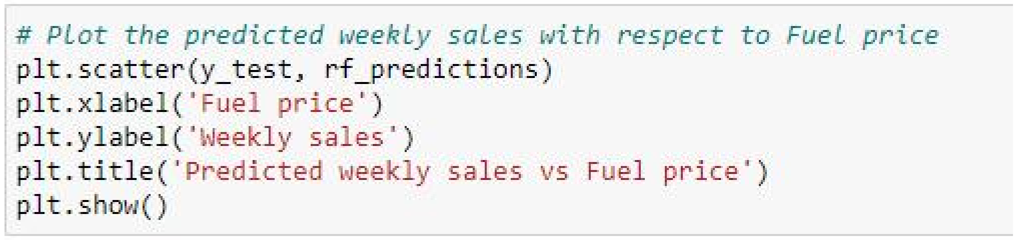


```python
# Plot permutation feature importance for Gradient Boosting Regressor
gb_perm_importance = permutation_importance(gb, X_test, y_test, n_repeats=10, random_state=42)
gb_sorted_idx = gb_perm_importance.importances_mean.argsort()
plt.figure(figsize=(8, 6))
plt.barh(X_train.columns[gb_sorted_idx], gb_perm_importance.importances_mean[gb_sorted_idx])
plt.xlabel("Permutation Importance")
plt.title("Gradient Boosting Regressor Permutation Feature Importance")
plt.show()
```

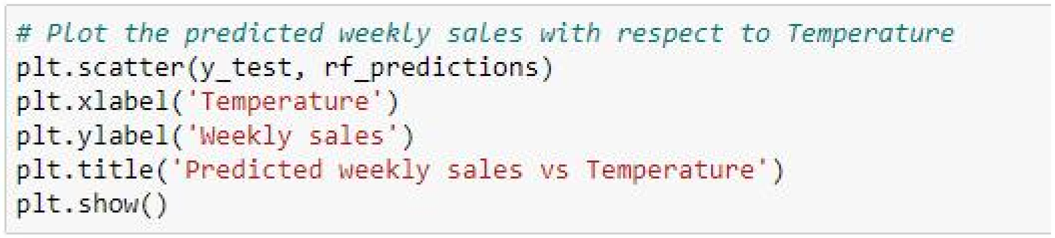


```python
# Plot permutation feature importance for Decision Tree Regressor
dt_perm_importance = permutation_importance(dt, X_test, y_test, n_repeats=10, random_state=42)
dt_sorted_idx = dt_perm_importance.importances_mean.argsort()
plt.figure(figsize=(8, 6))
plt.barh(X_train.columns[dt_sorted_idx], dt_perm_importance.importances_mean[dt_sorted_idx])
plt.xlabel("Permutation Importance")
plt.title("Decision Tree Regressor Permutation Feature Importance")
plt.show()
```

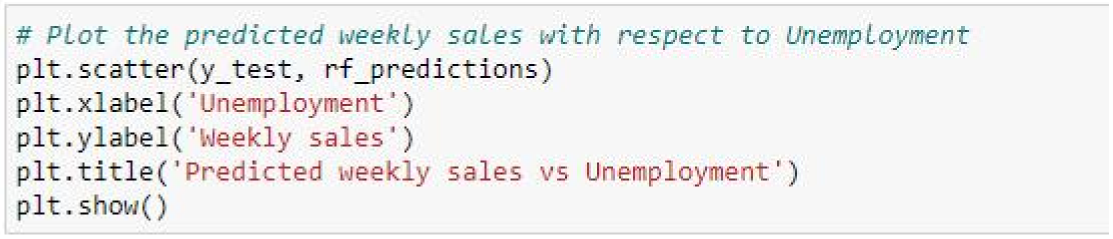
13.Predicting the sales values for different factors:



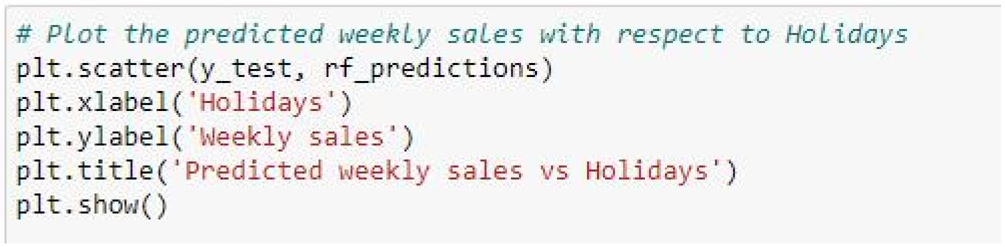
```python
# Plot the predicted weekly sales with respect to Fuel price
plt.scatter(y_test, rf_predictions)
plt.xlabel('Fuel price')
plt.ylabel('Weekly sales')
plt.title('Predicted weekly sales vs Fuel price')
plt.show()
```



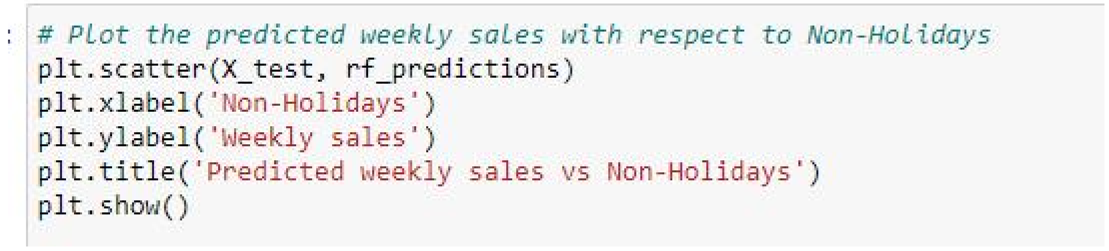
```python
# Plot the predicted weekly sales with respect to Temperature
plt.scatter(y_test, rf_predictions)
plt.xlabel('Temperature')
plt.ylabel('Weekly sales')
plt.title('Predicted weekly sales vs Temperature')
plt.show()
```



```python
# Plot the predicted weekly sales with respect to Unemployment
plt.scatter(y_test, rf_predictions)
plt.xlabel('Unemployment')
plt.ylabel('Weekly sales')
plt.title('Predicted weekly sales vs Unemployment')
plt.show()
```



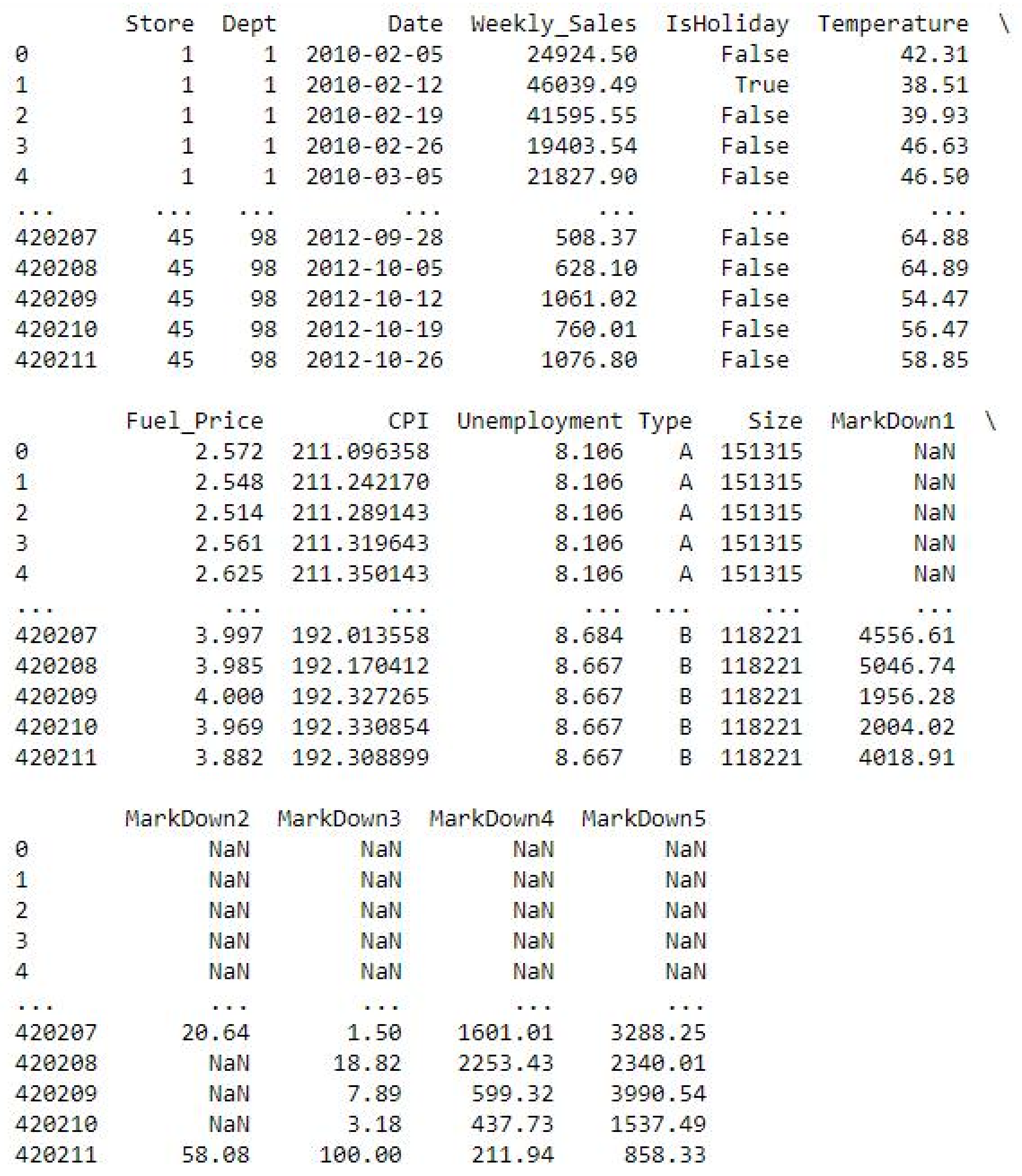
```python
# Plot the predicted weekly sales with respect to Holidays
plt.scatter(y_test, rf_predictions)
plt.xlabel('Holidays')
plt.ylabel('Weekly sales')
plt.title('Predicted weekly sales vs Holidays')
plt.show()
```



```python
# Plot the predicted weekly sales with respect to Non-Holidays
plt.scatter(X_test, rf_predictions)
plt.xlabel('Non-Holidays')
plt.ylabel('Weekly sales')
plt.title('Predicted weekly sales vs Non-Holidays')
plt.show()
```

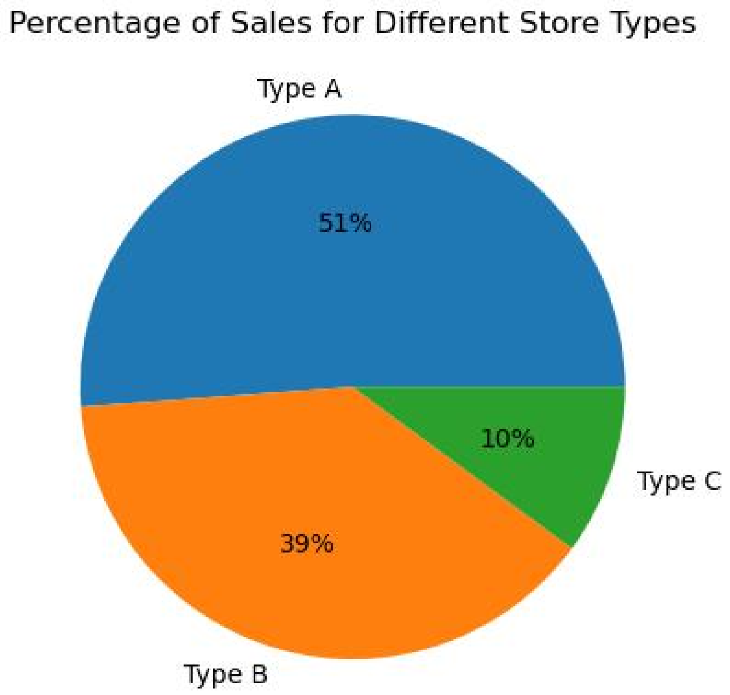
VI. RESULTS:

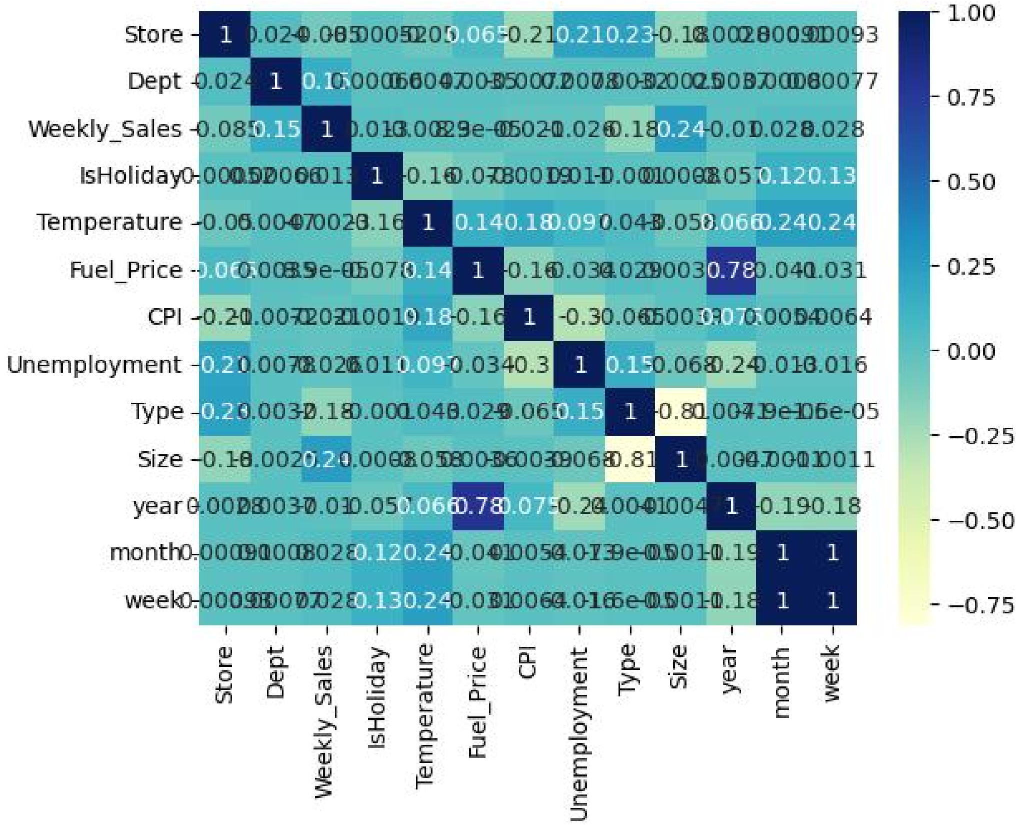
1. Showing the Merged Data:

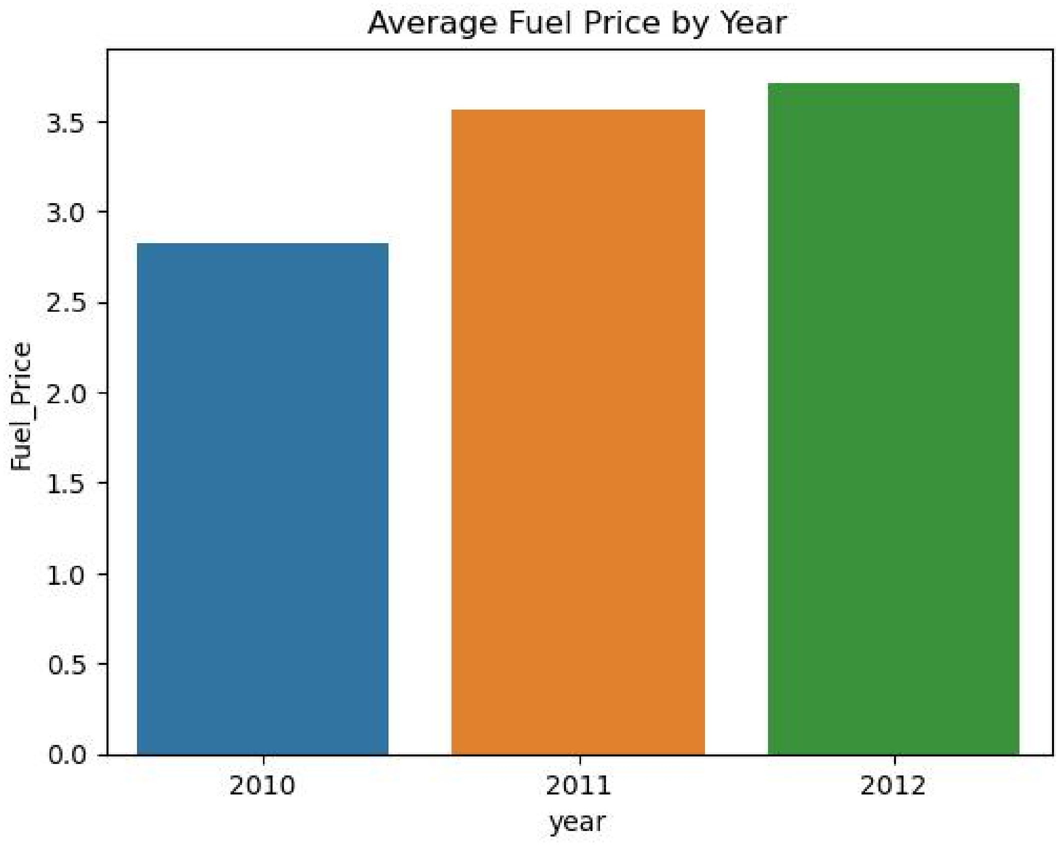


```
        Store  Dept        Date  Weekly_Sales  IsHoliday  Temperature  \
0           1     1  2010-02-05      24924.50      False        42.31
1           1     1  2010-02-12      46039.49       True        38.51
2           1     1  2010-02-19      41595.55      False        39.93
3           1     1  2010-02-26      19403.54      False        46.63
4           1     1  2010-03-05      21827.90      False        46.50
...       ...   ...         ...           ...        ...          ...
420207     45    98  2012-09-28        508.37      False        64.88
420208     45    98  2012-10-05        628.10      False        64.89
420209     45    98  2012-10-12       1061.02      False        54.47
420210     45    98  2012-10-19        760.01      False        56.47
420211     45    98  2012-10-26       1076.80      False        58.85

        Fuel_Price         CPI  Unemployment Type    Size  MarkDown1  \
0            2.572  211.096358         8.106    A  151315        NaN
1            2.548  211.242170         8.106    A  151315        NaN
2            2.514  211.289143         8.106    A  151315        NaN
3            2.561  211.319643         8.106    A  151315        NaN
4            2.625  211.350143         8.106    A  151315        NaN
...            ...         ...           ...  ...     ...        ...
420207       3.997  192.013558         8.684    B  118221    4556.61
420208       3.985  192.170412         8.667    B  118221    5046.74
420209       4.000  192.327265         8.667    B  118221    1956.28
420210       3.969  192.330854         8.667    B  118221    2004.02
420211       3.882  192.308899         8.667    B  118221    4018.91

        MarkDown2  MarkDown3  MarkDown4  MarkDown5
0             NaN        NaN        NaN        NaN
1             NaN        NaN        NaN        NaN
2             NaN        NaN        NaN        NaN
3             NaN        NaN        NaN        NaN
4             NaN        NaN        NaN        NaN
...           ...        ...        ...        ...
420207      20.64       1.50    1601.01    3288.25
420208        NaN      18.82    2253.43    2340.01
420209        NaN       7.89     599.32    3990.54
420210        NaN       3.18     437.73    1537.49
420211      58.08     100.00     211.94     858.33
```

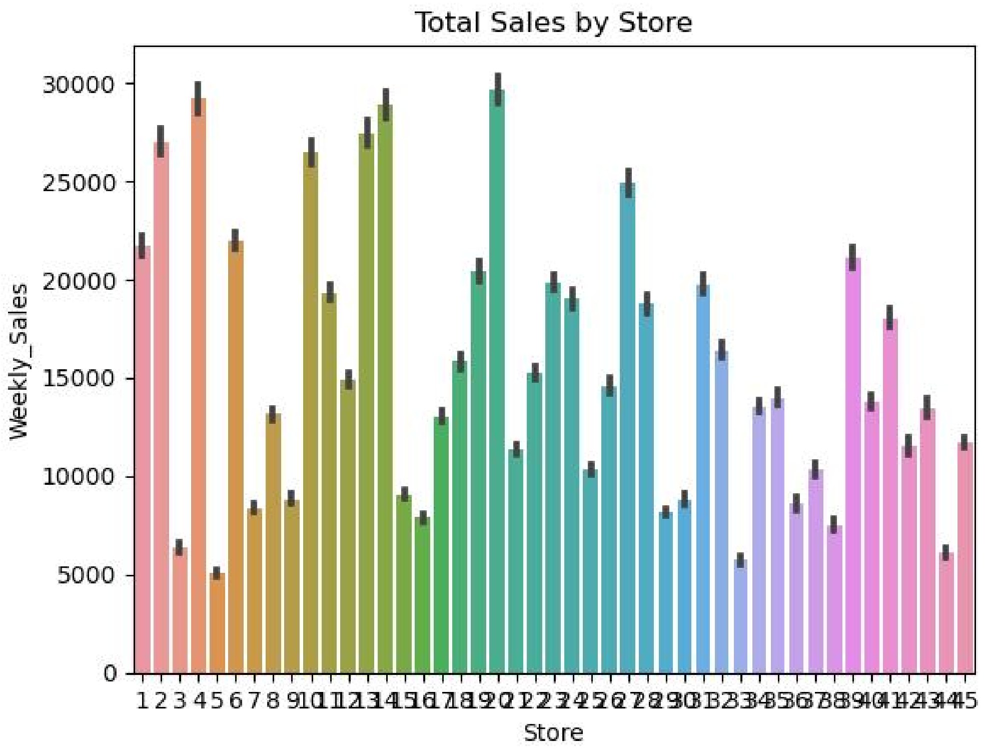
2. Showing all the plots with their titles :


Percentage of Sales for Different Store Types
Type A
51%
Type B
39%
Type C
10%

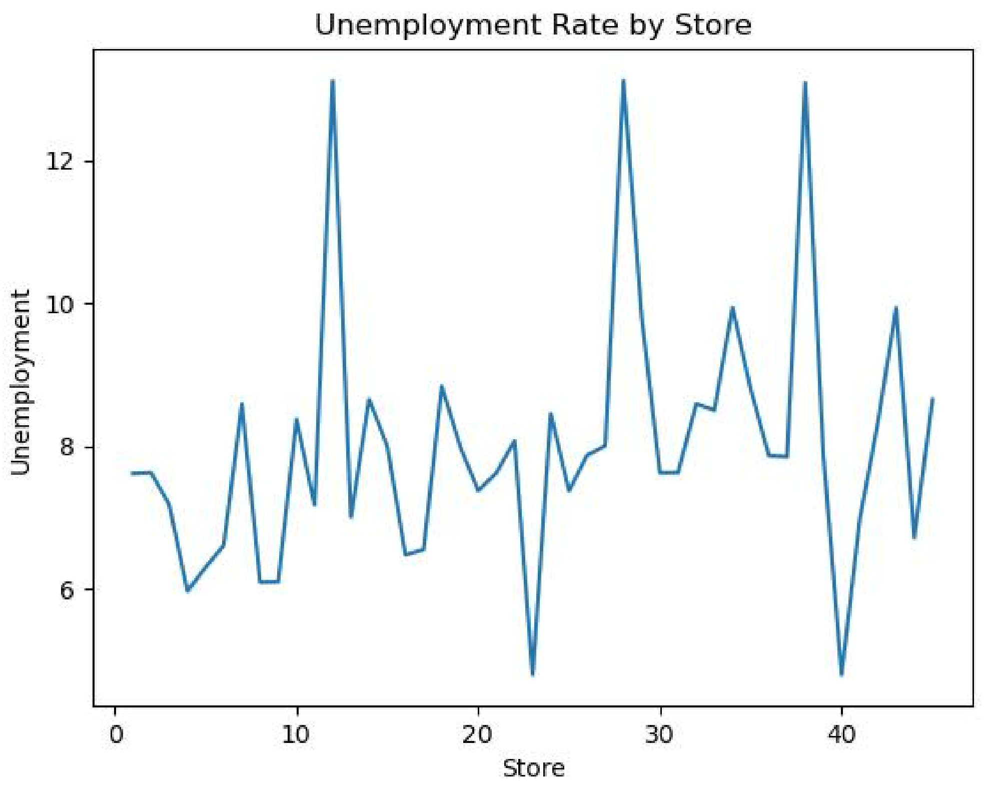

Store
Dept
Weekly_Sales
IsHoliday
Temperature
Fuel_Price
CPI
Unemployment
Type
Size
year
month
week
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

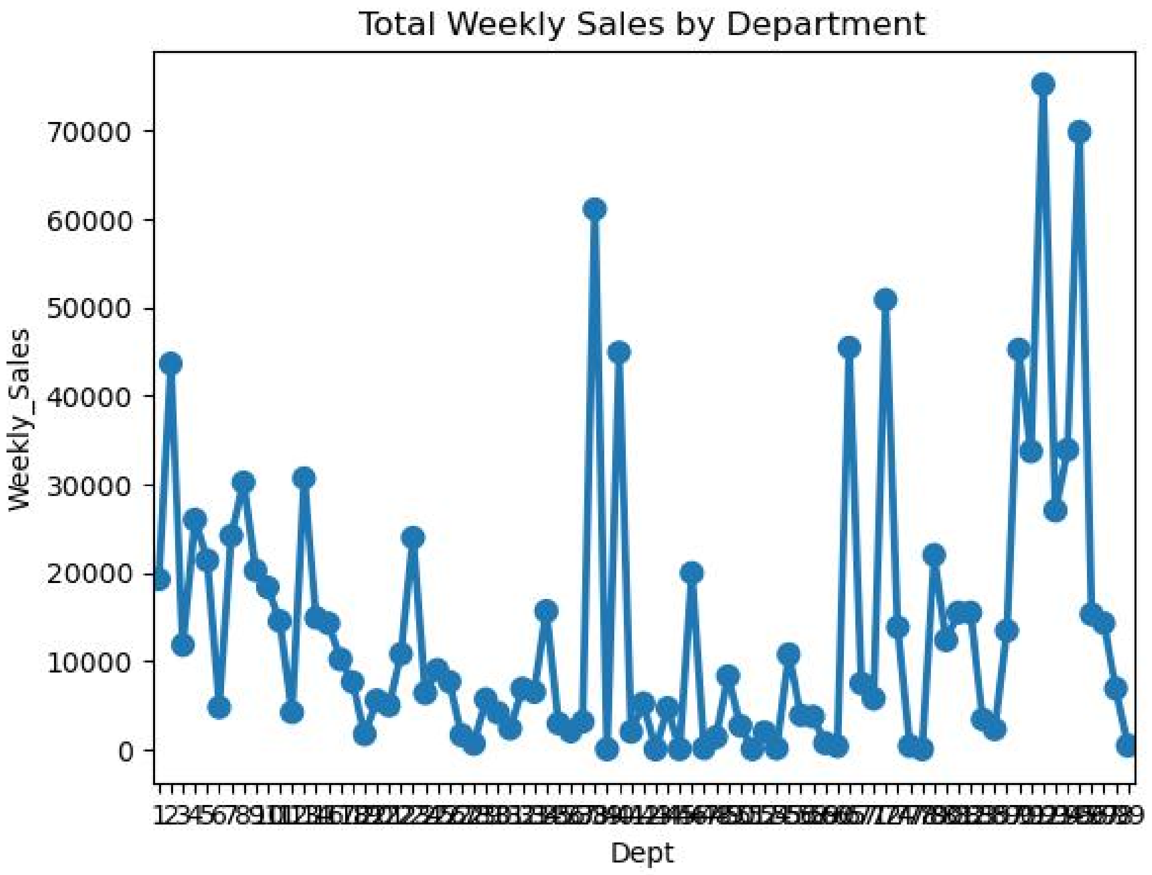


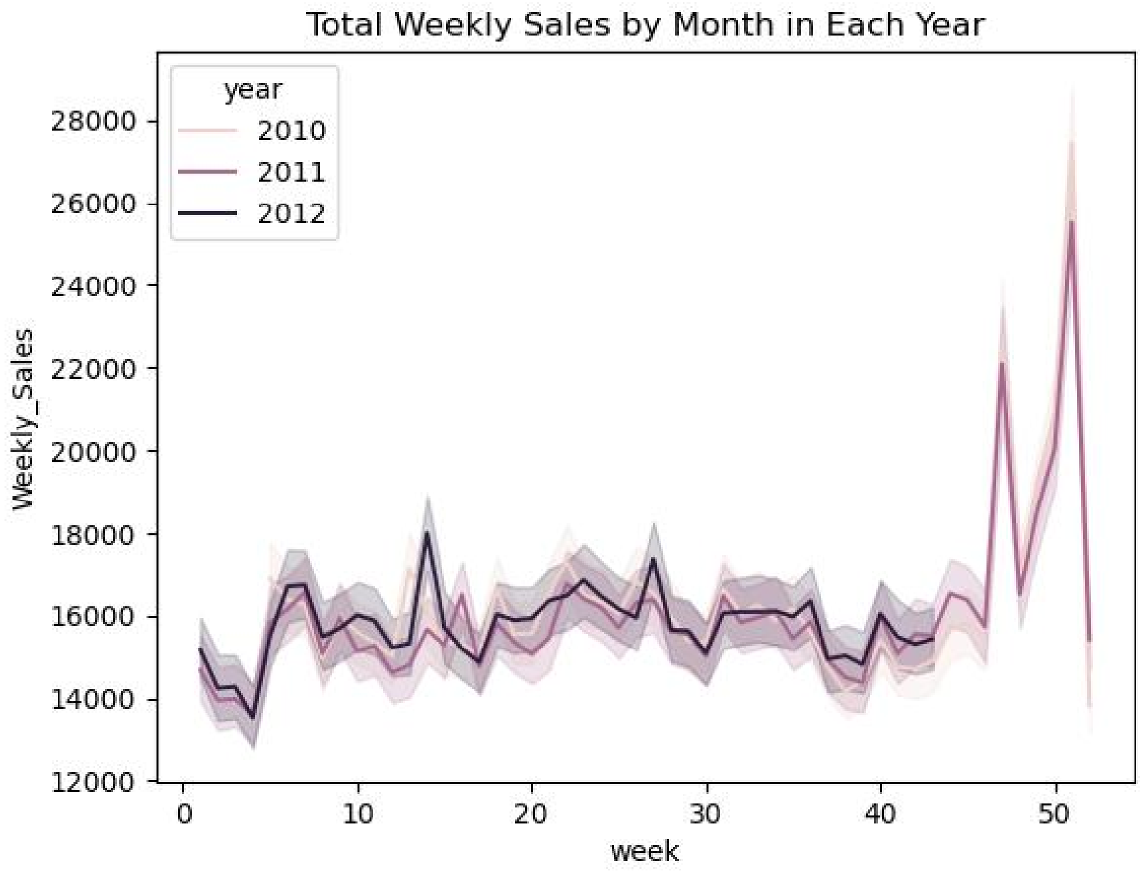
Average Fuel Price by Year
Fuel_Price
year
2010
2011
2012



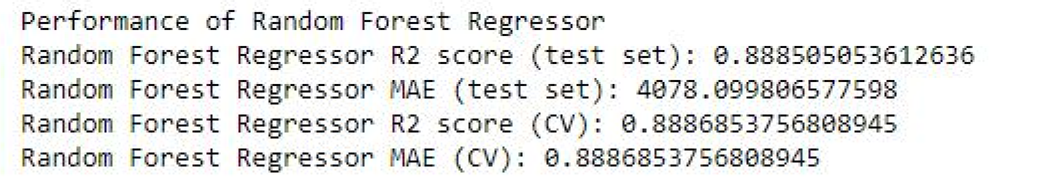
Total Sales by Store
Weekly_Sales
Store


Unemployment Rate by Store
Unemployment
Store
12
10
8
6
0
10
20
30
40

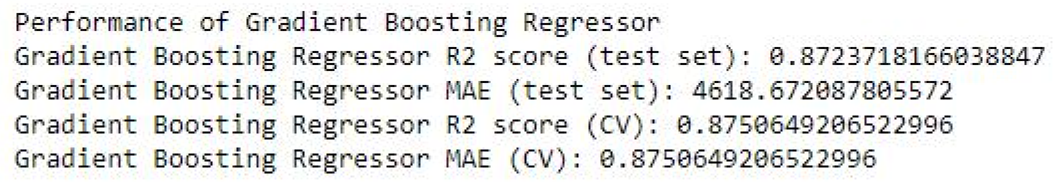

Total Weekly Sales by Department
Weekly_Sales
Dept
70000
60000
50000
40000
30000
20000
10000
0



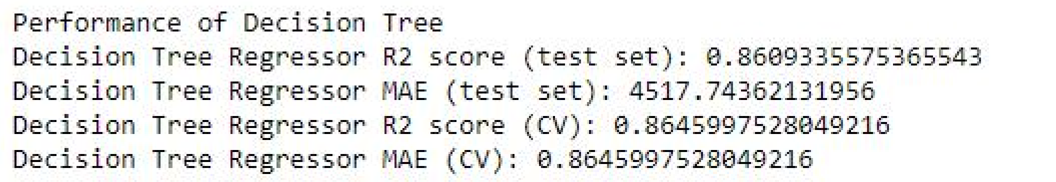
3. Viewing the result of all the 3 ML algorithms implemented:



```
Performance of Random Forest Regressor
Random Forest Regressor R2 score (test set): 0.888505053612636
Random Forest Regressor MAE (test set): 4078.099806577598
Random Forest Regressor R2 score (CV): 0.8886853756808945
Random Forest Regressor MAE (CV): 0.8886853756808945
```

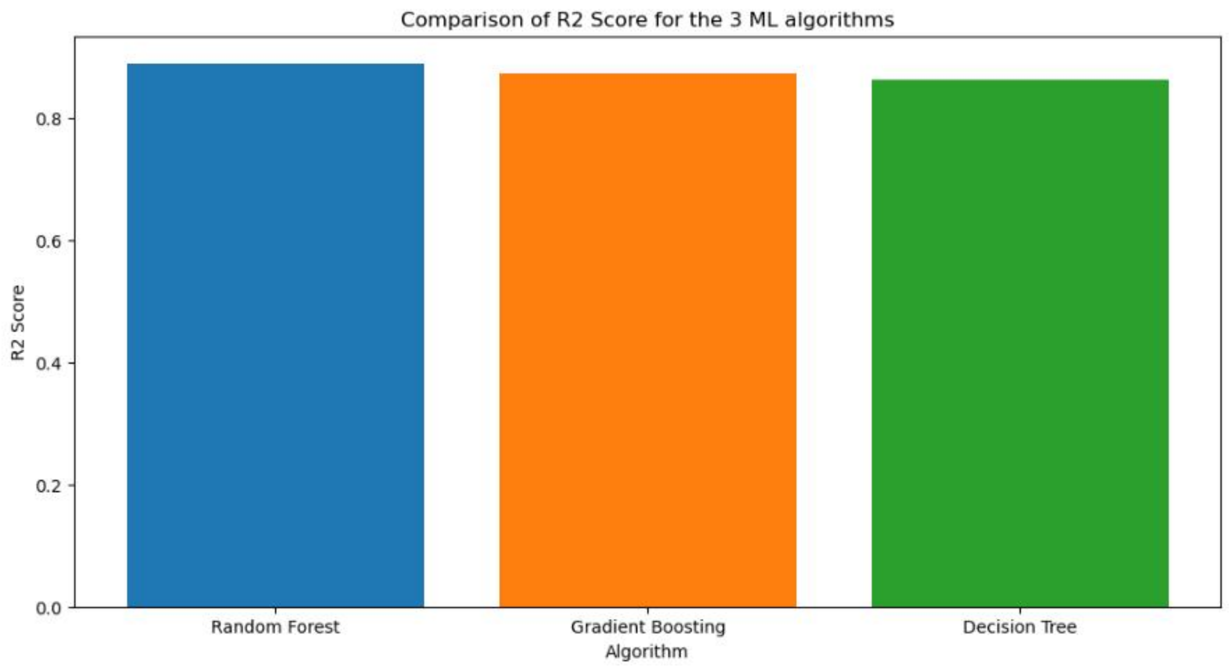


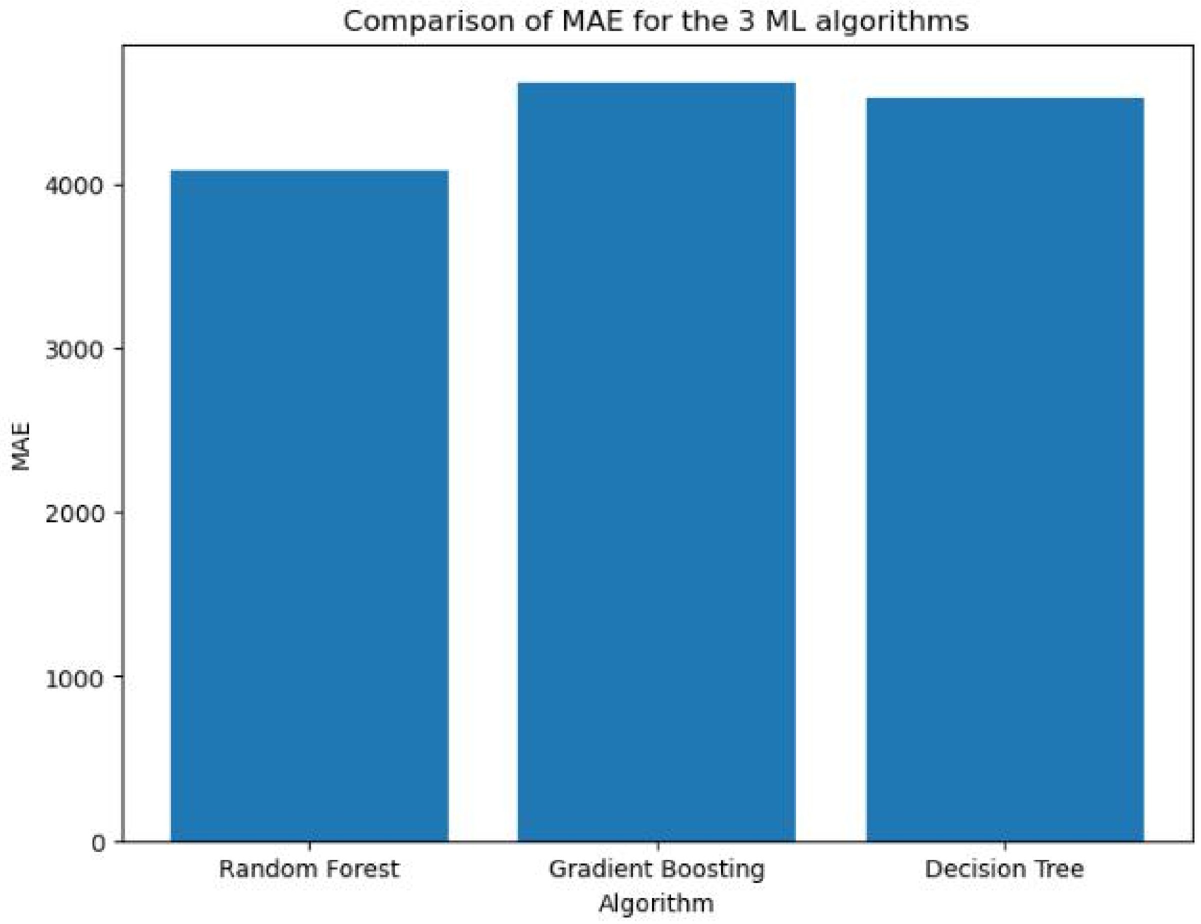
```
Performance of Gradient Boosting Regressor
Gradient Boosting Regressor R2 score (test set): 0.8723718166038847
Gradient Boosting Regressor MAE (test set): 4618.672087805572
Gradient Boosting Regressor R2 score (CV): 0.8750649206522996
Gradient Boosting Regressor MAE (CV): 0.8750649206522996
```



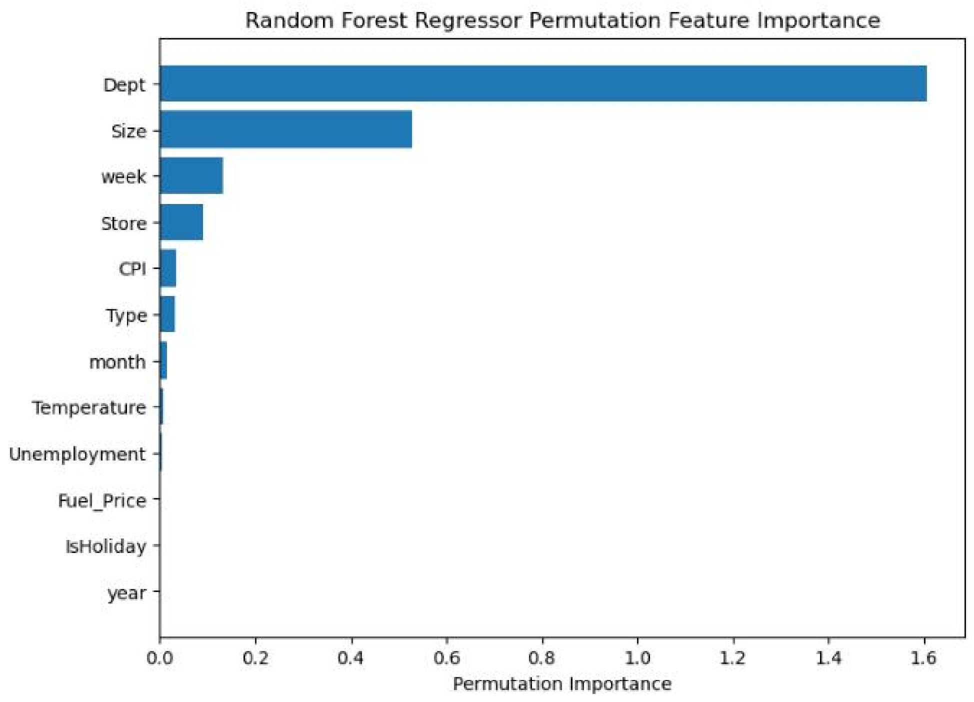
```
Performance of Decision Tree
Decision Tree Regressor R2 score (test set): 0.8609335575365543
Decision Tree Regressor MAE (test set): 4517.74362131956
Decision Tree Regressor R2 score (CV): 0.8645997528049216
Decision Tree Regressor MAE (CV): 0.8645997528049216
```

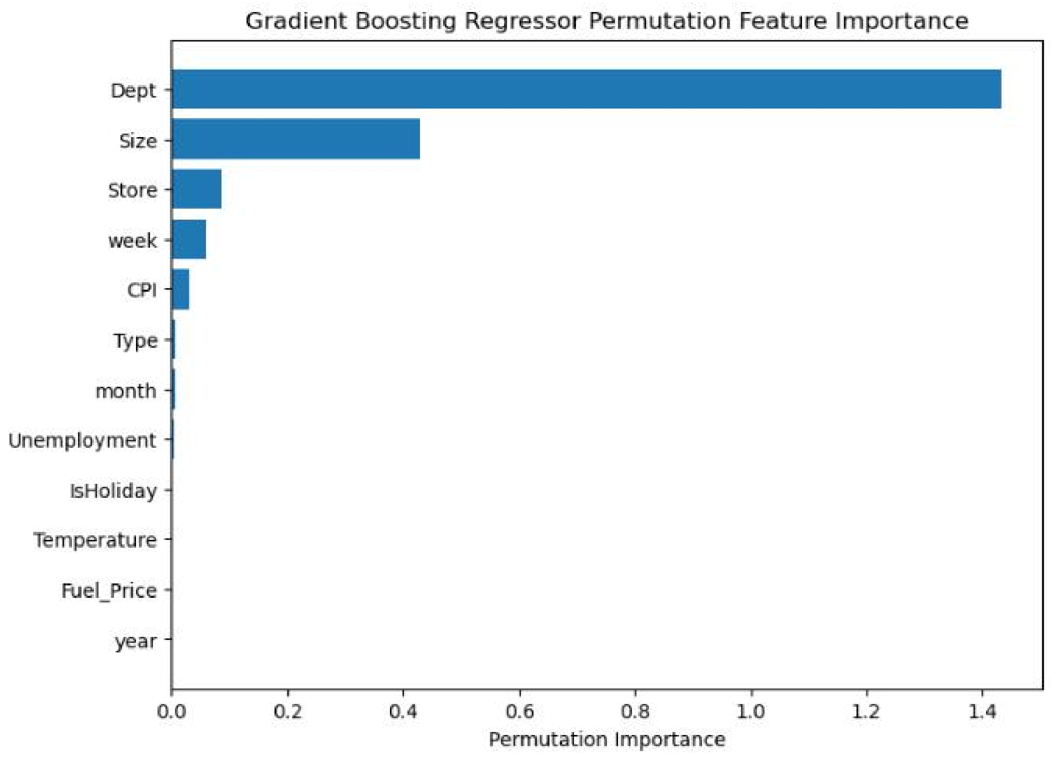
Comparing the results of the 3 ML algorithms:

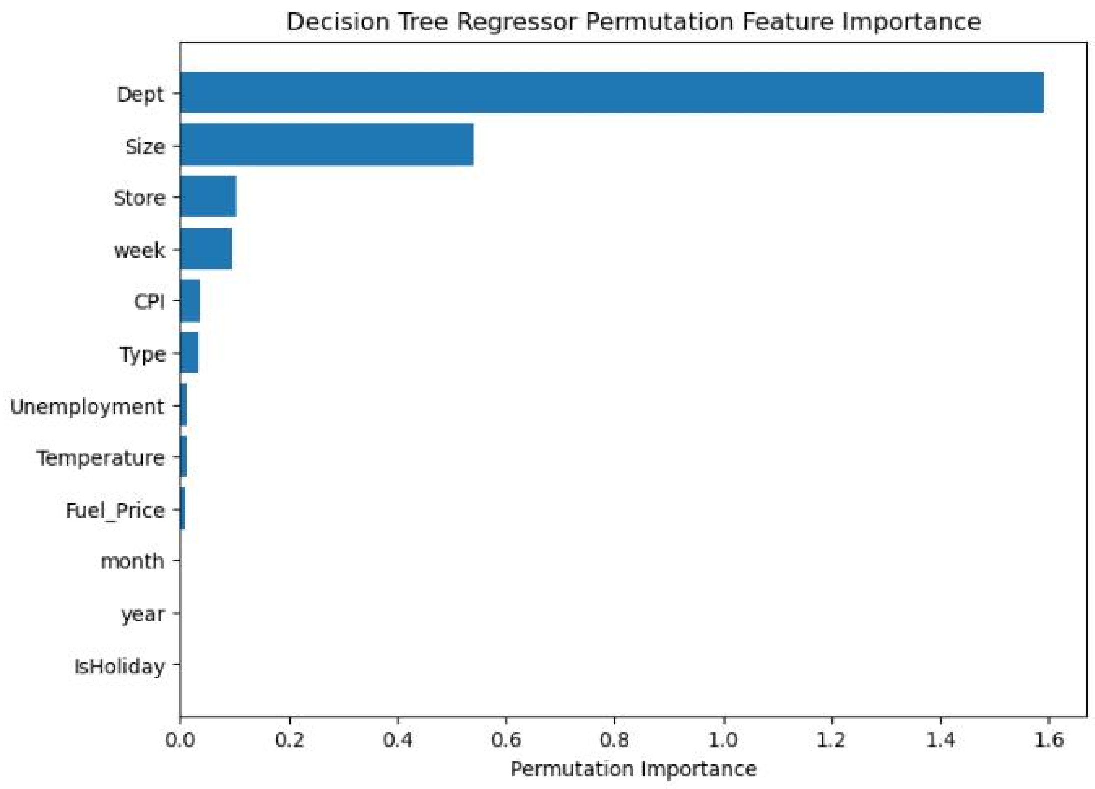




Viewing the Important features for each ML algorithm:


Random Forest Regressor Permutation Feature Importance
Dept
Size
week
Store
CPI
Type
month
Temperature
Unemployment
Fuel_Price
IsHoliday
year
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Permutation Importance


Gradient Boosting Regressor Permutation Feature Importance
Dept
Size
Store
week
CPI
Type
month
Unemployment
IsHoliday
Temperature
Fuel_Price
year
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Permutation Importance


Decision Tree Regressor Permutation Feature Importance
Dept
Size
Store
week
CPI
Type
Unemployment
Temperature
Fuel_Price
month
year
IsHoliday
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
Permutation Importance

VII. ADVANTAGES AND LIMITATIONS OF THE APPROACH

Our algorithm implements three different approaches for regression: Random Forest Regression, Gradient Boosting Regression, and Decision Tree Regression. Each of these algorithms has its own advantages and limitations.

1. **Random Forest Regression**: Random Forest Regression is an ensemble method that combines multiple decision trees to improve the accuracy of predictions. It randomly selects subsets of features and data points to create multiple decision trees, which are then combined to create a final prediction.

Usage in our approach: In our algorithm, the Random Forest Regression is used with 100 trees, which can result in high accuracy for the model.

2. Gradient Boosting Regressor: Gradient Boosting Regressor is an optimized implementation of gradient boosting, which is an ensemble method that combines multiple weak models (decision trees) to create a strong predictive model.

Usage in our approach: In our algorithm, the Gradient Boosting Regressor is used with a learning rate of 0.1 and 100 trees, which can result in high accuracy for the model.

3. Decision Trees: A decision tree is a tree-shaped model that represents a set of decisions and their possible consequences. It starts with a single node, which represents the initial decision or the input data. From there, it branches out into multiple nodes, each representing a decision or a possible outcome. The tree continues to grow until it reaches a final decision or a leaf node.

Usage in our approach: In our algorithm, the Decision Tree Regressor is used with a maximum depth of 3, which can result in a simple and interpretable model.

Advantages of the Approach:

- The approach is based on a well-established machine learning algorithm, namely the random forest algorithm, which is known for its high accuracy.
- The use of cross-validation ensures that the model is trained on a wide range of data and is not overfitting to the training data.
- The use of hyperparameter tuning helps to optimize the performance of the model and improve its accuracy.
- The use of feature importance helps to identify the most important features that are contributing to the prediction.

Limitations of the Approach:

- The model assumes that the relationships between the features and the target variable are linear, which may not always be the case in real-world scenarios.
- The model may not be able to capture complex non-linear relationships between the features and the target variable.
- The model may suffer from the problem of high bias or high variance, depending on the complexity of the model and the size of the training data.
- In a few cases, the model may not be able to handle missing or incomplete data, which can lead to inaccurate predictions.
- The model may be influenced by outliers in the data, which can have a significant impact on the accuracy of the predictions.

In terms of the nature of the prediction it is giving, each of these algorithms has its own strengths and limitations. Decision Trees are well suited for problems where the relationships between variables are relatively simple and easy to understand. Random Forest and Gradient Boost, on the other hand, are better suited for problems where the relationships between variables are more complex and difficult to understand.

Additionally, Random Forest and Gradient Boost can handle high-dimensional data, which can be useful for problems with many features. However, all these algorithms may struggle with imbalanced datasets, where one class is much more prevalent than the others. It is important to carefully evaluate the performance of each algorithm on the specific problem at hand to determine which algorithm is most appropriate for that problem.

VIII. PRACTICAL APPLICATIONS OF THE APPROACH

This application is a machine learning project that aims to predict the weekly sales of Walmart stores based on historical data. The project involves several steps, including data cleaning, data visualization, feature engineering, and model training and evaluation. The practical applications of this approach can be manifold, including optimizing inventory management, improving customer experience, and increasing revenue.

• One practical application of this approach is inventory management. By predicting the weekly sales of each store and department, Walmart can optimize its inventory management by ensuring that it stocks enough products to meet demand while minimizing excess inventory that can lead to waste and decreased profitability. For example, if the model predicts that a particular store and department will experience high sales during a specific week, Walmart can ensure that it has enough stock on hand to meet the expected demand. Conversely, if the model predicts low sales for a particular week, Walmart can adjust its inventory levels, accordingly, reducing the risk of overstocking and waste.

• Another practical application of this approach is improving customer experience. By analyzing historical data, the model can identify trends and patterns in customer behavior, such as buying habits, preferences, and seasonality. Walmart can use this information to tailor its marketing and promotional activities to specific customer segments, such as by offering discounts or promotions on products that are popular among certain groups of customers. This can lead to increased customer loyalty and satisfaction, as well as increased sales and revenue.

• Additionally, the model can help Walmart identify which stores and departments are underperforming and take corrective action to improve their performance. For example, if the model predicts that a particular store and department will have low sales during a particular week, Walmart can investigate the underlying reasons for the poor performance, such as insufficient staffing, poor product selection, or ineffective marketing, and take corrective action to address the issues.

• Furthermore, the model can help Walmart forecast sales and revenue more accurately, allowing the company to make more informed decisions about resource allocation and budgeting. For example, if the model predicts that a particular store and department will experience high sales during a particular week, Walmart can allocate additional resources to the store to meet the expected demand, such as by hiring additional staff, increasing inventory levels, or increasing marketing efforts. Conversely, if the model predicts low sales for a particular week, Walmart can adjust its resource allocation, accordingly, reducing the risk of overspending and maximizing profitability.

Some additional real-time approaches:

Personalized Marketing: By analyzing customer behavior and purchase history, retailers can create personalized marketing campaigns that target specific customers with relevant products and offers. This can be done through targeted email campaigns, personalized recommendations on the website or app, or even targeted advertising on social media platforms. By tailoring marketing messages to individual customers, retailers can increase the likelihood of a sale and improve customer loyalty.

Store Layout Optimization: By analyzing customer traffic patterns and behavior within a store, retailers can optimize the layout to improve the customer experience and increase sales. For example, retailers can use heat mapping to identify high-traffic areas within the store and place high-margin products in these areas. They can also use machine learning algorithms to analyze customer behavior and make real-time adjustments to product placement and store layout.

Predictive Maintenance: Retailers rely on a wide range of equipment, from refrigeration units to cash registers, to keep their stores running smoothly. By using machine learning algorithms to analyze data from these systems, retailers can identify potential problems before they occur and schedule maintenance to prevent downtime. This can help ensure that stores are always operating at peak efficiency and minimize the risk of lost sales due to equipment failures.

In summary, the practical applications of this approach are numerous and varied, including optimizing inventory management, improving customer experience, identifying, and correcting underperforming stores and departments, and forecasting sales and revenue more accurately. By leveraging machine learning techniques to analyze historical data and make predictions about future sales, Walmart can make more informed decisions about resource allocation, budgeting, and marketing, leading to increased profitability and customer satisfaction.

IX. CONCLUSION:

In conclusion, the regression models we created can help Walmart predict how much they're going to sell each week based on different things like the type of store, fuel prices, unemployment rates, and which departments are selling the most. We tested two different models, and they both did well at predicting sales.

Furthermore, our analysis showed that store type, fuel prices, and department were some of the most important factors affecting Walmart's weekly sales. By leveraging the insights from these models, Walmart can optimize their inventory, pricing, and marketing strategies to improve profitability and better meet the needs of their customers.

However, it's important to keep in mind that these models work best when there's a straight-forward relationship between the things we're looking at (like store type) and how much Walmart sells. They might not work as well if there are lots of complicated factors interacting with each other. Also, the models need a lot of data to work well, and they can be sensitive to mistakes or missing information in the data we're using.

Overall, these models can help Walmart make better decisions about how to stock their stores, set prices, and advertise their products. But it's important to use the models together with other kinds of analysis and expert knowledge to get the best results. We should also keep an eye on how well the models are doing and update them as needed to make sure they stay accurate over time.

References:

[1] M. Franco-Santos and M. Bourne, "The impact of performance targets on behaviour: a close look at sales force contexts," Research executive summaries series, vol. 5, 2009.

[2] D. Silverman, Interpreting Qualitative Data: Methods for Analyzing Talk, Text and Interaction 3rd Ed. Text and Interaction, Sage Publications Ltd: Methods for Analyzing Talk, 2006.

[3] UBM. (2003) Big Data analytics: Descriptive vs. predictive vs. prescriptive. [Accessed 17 September 2017]. [Online]. Available: www.informationweek.com/about-us/d/d-id/705542

[4] A. S. Harsoor and A. Patil, "Forecast of sales of walmart store using Big Data application," International Journal of Research in Engineering and Technology, vol. 4, p. 6, June 2015.

[5] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mccauley, M. J. Franklin, S. Shenker, and I. Stoica, "Fast and interactive analytics over Hadoop data with Spark," Usenix - The Advanced Computing Systems Association, 2012.

[6] J. Dean and S. Ghemawat, MapReduce: simplified data processing on large clusters. Association for Computing Machinery, 2008.

[7] A. Katal, M. Wazid, and R. H. Goudar, Big Data: Issues, Challenges, Tools and Good Practices, 2013.

[8] P. A. Riyaz and V. Surekha, Leveraging MapReduce With Hadoop for Weather Data Analytics. OSR Journal of Computer Engineering (IOSR, 2015. [9] P. Chouksey and A. S. Chauhan, A Review of Weather Data Analytics using Big Data. International Journal of Advanced Research in Computer and Communication Engineering, 2017.

[10] A. Zaslavsky, C. Perera, and D. Georgakopoulos, "Sensing as a service and Big Data," in Proceedings of the International Conference on Advances in Cloud Computing (ACC), I. Bangalore, Ed., July 2012.

Contribution:

Handled the ML algorithm implementation part along with the Model evaluation and the comparison.

Contribution in implementation:

- **Data Preprocessing**
- **Training and comparison of 3 ML algorithms (Modelling)**

Phases in Documentation.

- **Description of the Model Approach**
- **Results**