Walmart Sales Prediction Documentation

By **OUBAZIZ Wassim** and **CHELGHAM Mounsif**

1 Introduction

This project aims to predict weekly sales for Walmart stores using machine learning algorithms. The dataset contains various features such as store type, size, department, temperature, fuel price, CPI, unemployment rate, and markdowns.

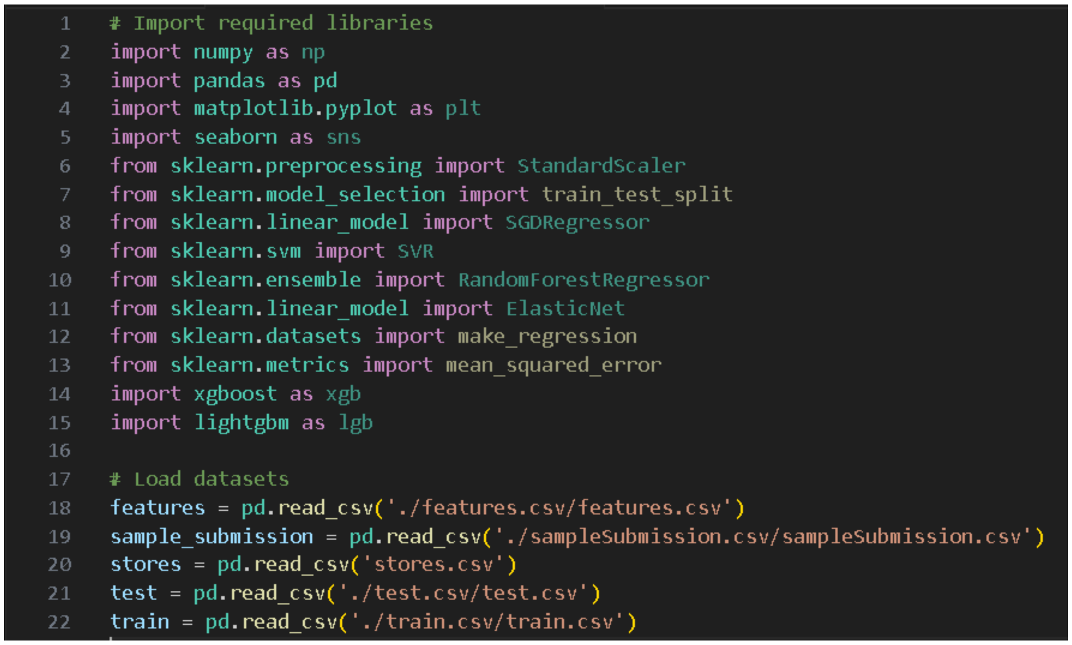
1.1 Objectives

- Predict weekly sales for Walmart stores.
- Identify factors influencing sales.
- Optimize model performance.

2 Project Setup

The project begins with importing necessary libraries and loading the datasets.

```
# Python code snippet for Importing libraries and Loading datasets
```



```python
# Import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDRegressor
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import ElasticNet
from sklearn.datasets import make_regression
from sklearn.metrics import mean_squared_error
import xgboost as xgb
import lightgbm as lgb

# Load datasets
features = pd.read_csv('./features.csv/features.csv')
sample_submission = pd.read_csv('./sampleSubmission.csv/sampleSubmission.csv')
stores = pd.read_csv('stores.csv')
test = pd.read_csv('./test.csv/test.csv')
train = pd.read_csv('./train.csv/train.csv')
```

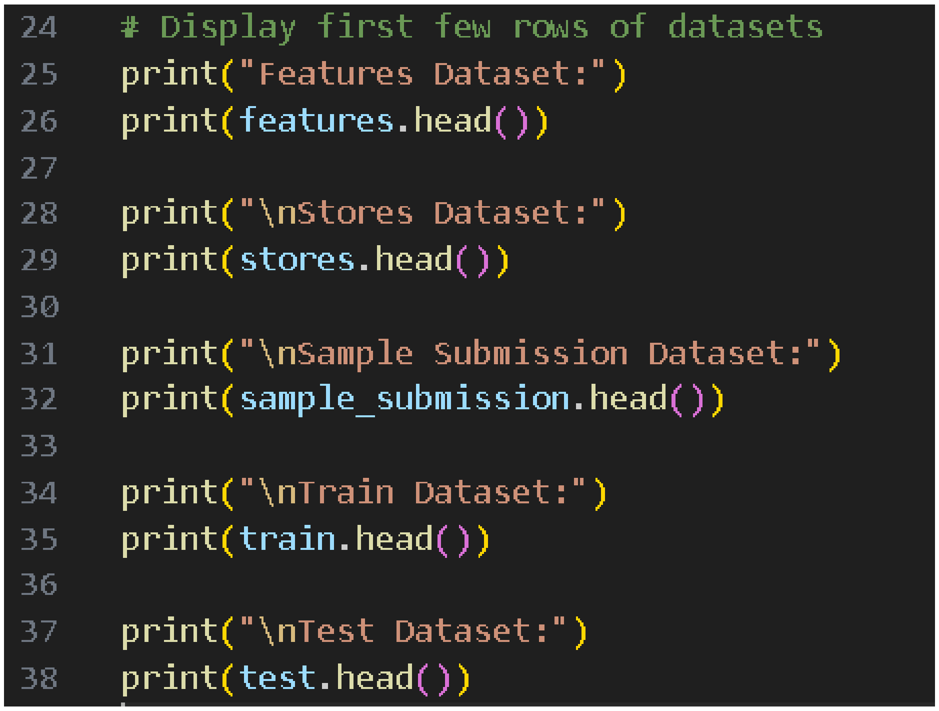
Figure 1: Loading datasets

3 Data Exploration

We perform exploratory data analysis (EDA) to gain insights into the dataset.

3.1 Overview of Data

```
# Python code snippet of displaying headers of datasets
```

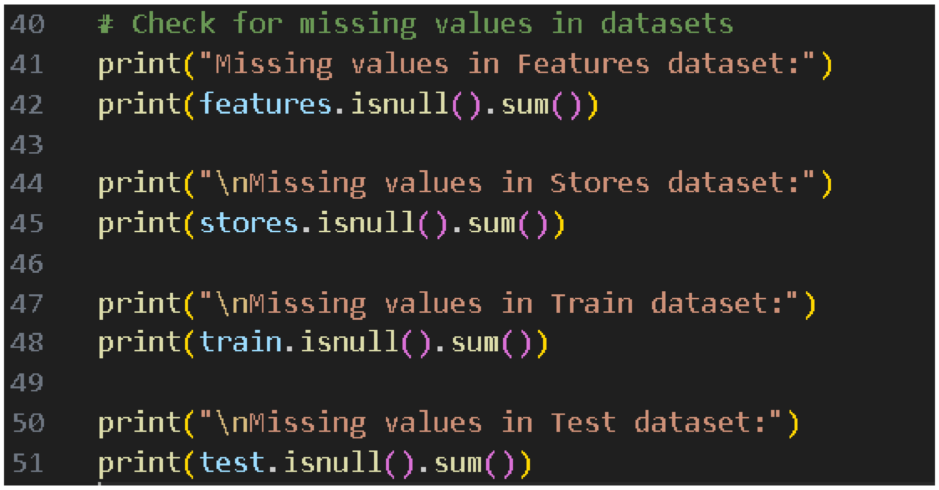


```python
# Display first few rows of datasets
print("Features Dataset:")
print(features.head())

print("\nStores Dataset:")
print(stores.head())

print("\nSample Submission Dataset:")
print(sample_submission.head())

print("\nTrain Dataset:")
print(train.head())

print("\nTest Dataset:")
print(test.head())
```

Figure 2: Overview of Data

3.2 Missing Values

```
# Python code snippet for checking missing values
```

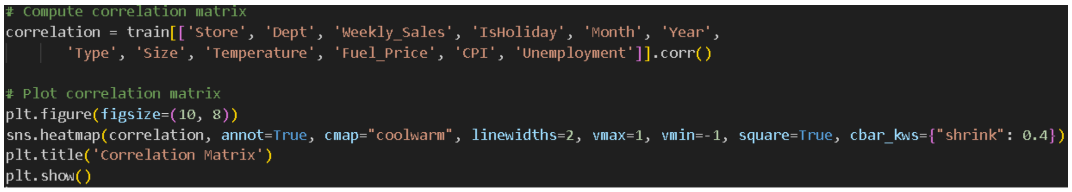


```python
# Check for missing values in datasets
print("Missing values in Features dataset:")
print(features.isnull().sum())

print("\nMissing values in Stores dataset:")
print(stores.isnull().sum())

print("\nMissing values in Train dataset:")
print(train.isnull().sum())

print("\nMissing values in Test dataset:")
print(test.isnull().sum())
```

Figure 3: Checking Missing Values

3.3 Correlation Matrix

```
# Python code snippet for Correlation Matrix
```



```
# Compute correlation matrix
correlation = train[['Store', 'Dept', 'Weekly_Sales', 'IsHoliday', 'Month', 'Year',
        'Type', 'Size', 'Temperature', 'Fuel_Price', 'CPI', 'Unemployment']].corr()

# Plot correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation, annot=True, cmap="coolwarm", linewidths=2, vmax=1, vmin=-1, square=True, cbar_kws={"shrink": 0.4})
plt.title('Correlation Matrix')
plt.show()
```

Figure 4: Correlation Matrix Code

Execution...

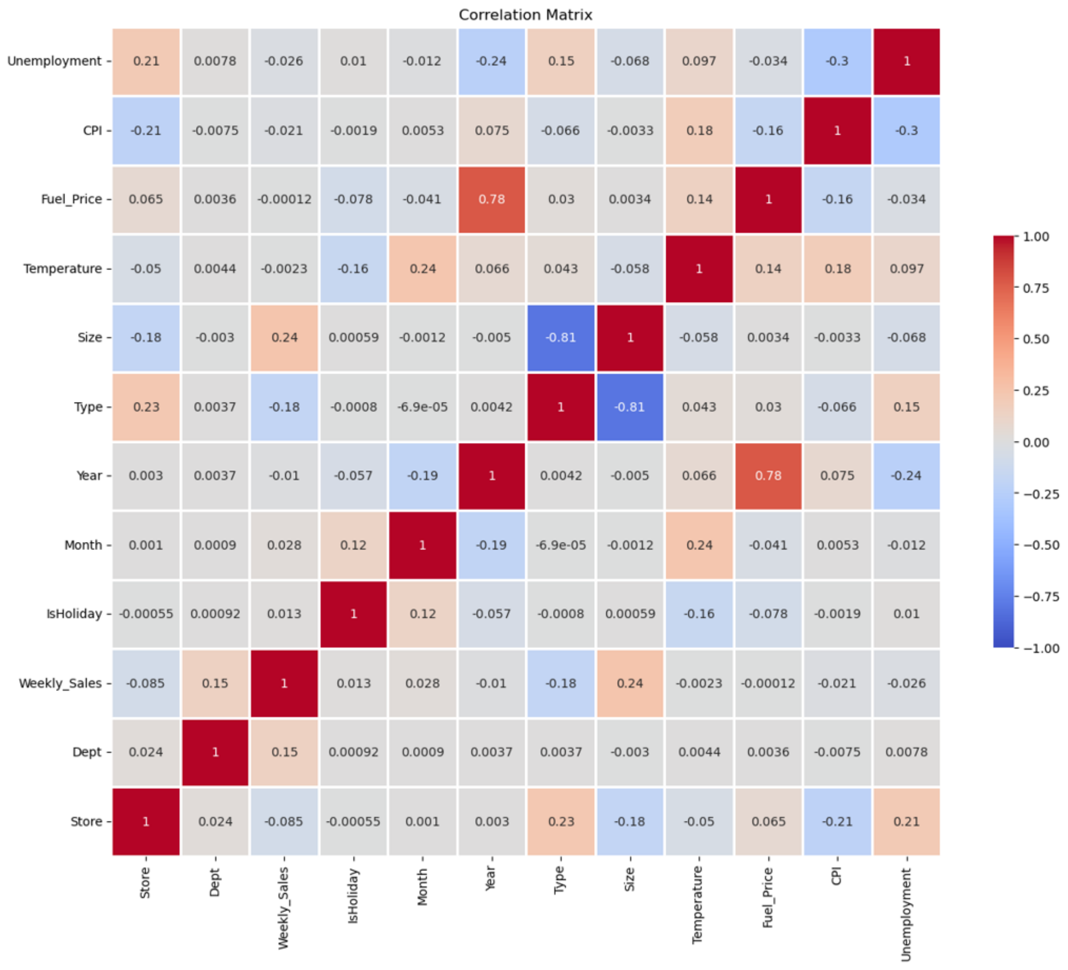
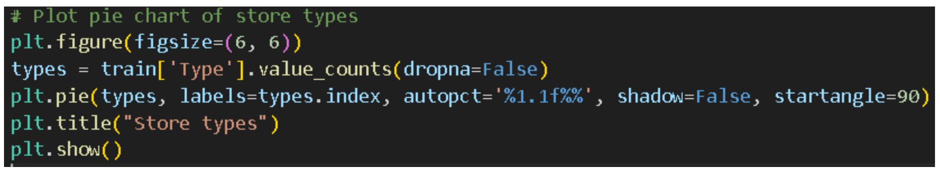


Figure 5: Correlation Matrix

3.4 Store Type Distribution

```
# Python code snippet for displaying a pie chart of store types
```



```
# Plot pie chart of store types
plt.figure(figsize=(6, 6))
types = train['Type'].value_counts(dropna=False)
plt.pie(types, labels=types.index, autopct='%1.1f%%', shadow=False, startangle=90)
plt.title("Store types")
plt.show()
```

Figure 6: Code of Store Type Distribution

Execution...

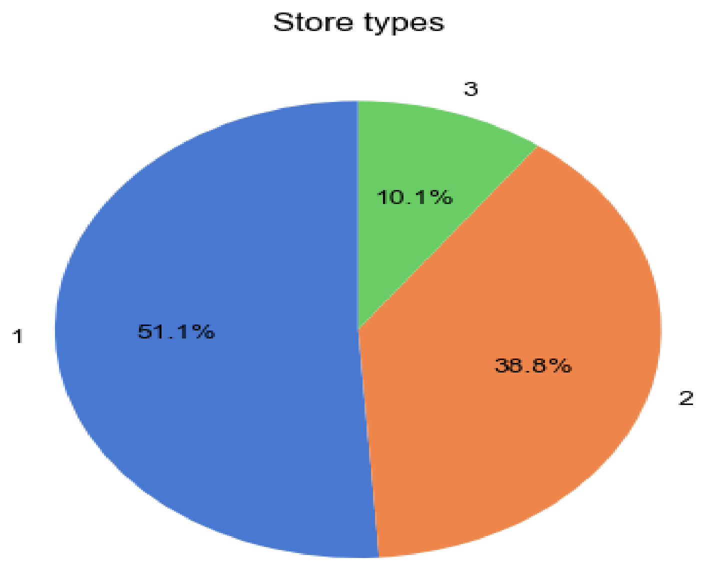


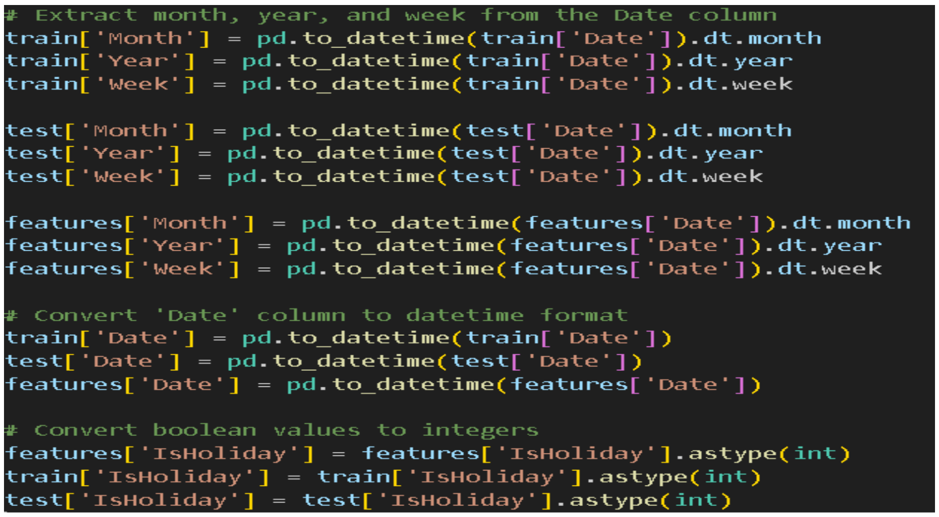
Figure 7: Store Type Distribution

4 Data Preprocessing

We preprocess the data to prepare it for model training.

4.1 Feature Engineering

```
# Python code snippet for feature engineering
```



```
# Extract month, year, and week from the Date column
train['Month'] = pd.to_datetime(train['Date']).dt.month
train['Year'] = pd.to_datetime(train['Date']).dt.year
train['Week'] = pd.to_datetime(train['Date']).dt.week

test['Month'] = pd.to_datetime(test['Date']).dt.month
test['Year'] = pd.to_datetime(test['Date']).dt.year
test['Week'] = pd.to_datetime(test['Date']).dt.week

features['Month'] = pd.to_datetime(features['Date']).dt.month
features['Year'] = pd.to_datetime(features['Date']).dt.year
features['Week'] = pd.to_datetime(features['Date']).dt.week

# Convert 'Date' column to datetime format
train['Date'] = pd.to_datetime(train['Date'])
test['Date'] = pd.to_datetime(test['Date'])
features['Date'] = pd.to_datetime(features['Date'])

# Convert boolean values to integers
features['IsHoliday'] = features['IsHoliday'].astype(int)
train['IsHoliday'] = train['IsHoliday'].astype(int)
test['IsHoliday'] = test['IsHoliday'].astype(int)
```

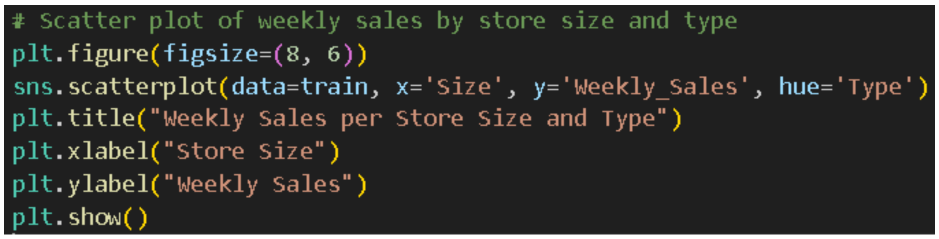
Figure 8: Feature Engineering Code

5 Exploratory Data Analysis (continued)

We continue exploring the data to understand the relationship between features and sales.

5.1 Weekly Sales vs. Store Size

```
# Python code snippet for plotting weekly sales vs. store size
```



```
# Scatter plot of weekly sales by store size and type
plt.figure(figsize=(8, 6))
sns.scatterplot(data=train, x='Size', y='Weekly_Sales', hue='Type')
plt.title("Weekly Sales per Store Size and Type")
plt.xlabel("Store Size")
plt.ylabel("Weekly Sales")
plt.show()
```

Figure 9: Weekly Sales vs. Store Size Code

Execution...

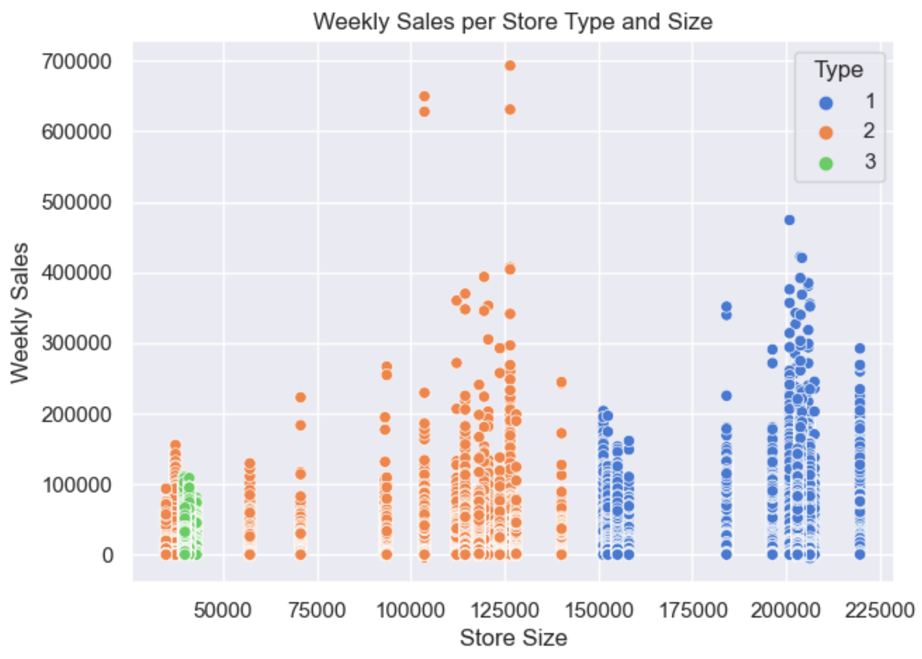


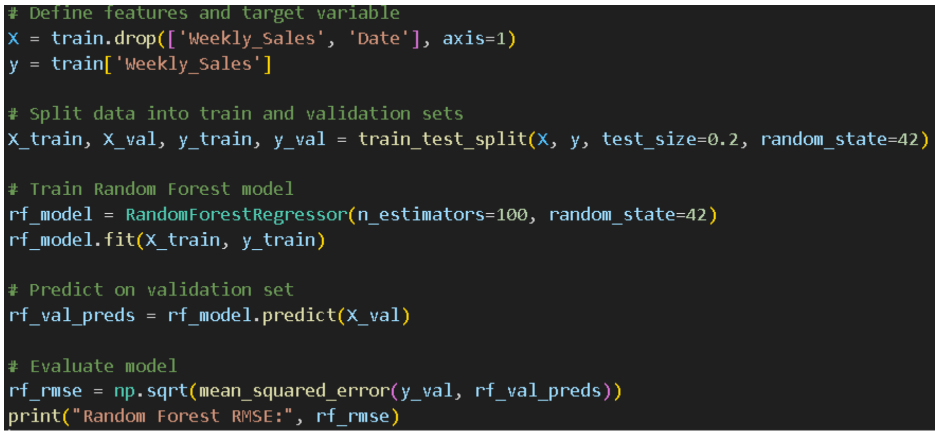
Figure 10: Weekly Sales vs. Store Size

6 Model Training

We train machine learning models to predict weekly sales.

6.1 Random Forest Regression

```
# Python code snippet for training Random Forest model
```



```python
# Define features and target variable
X = train.drop(['Weekly_Sales', 'Date'], axis=1)
y = train['Weekly_Sales']

# Split data into train and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict on validation set
rf_val_preds = rf_model.predict(X_val)

# Evaluate model
rf_rmse = np.sqrt(mean_squared_error(y_val, rf_val_preds))
print("Random Forest RMSE:", rf_rmse)
```

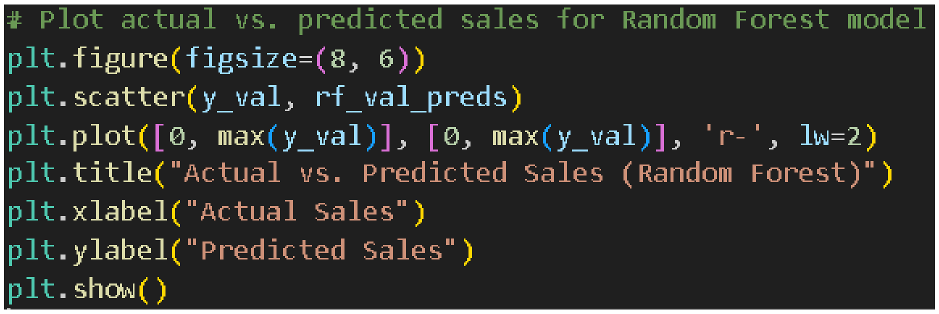
Figure 11: Training Random Forest Model Code

7 Model Evaluation

We evaluate the performance of the trained models.

7.1 Random Forest Regression Results

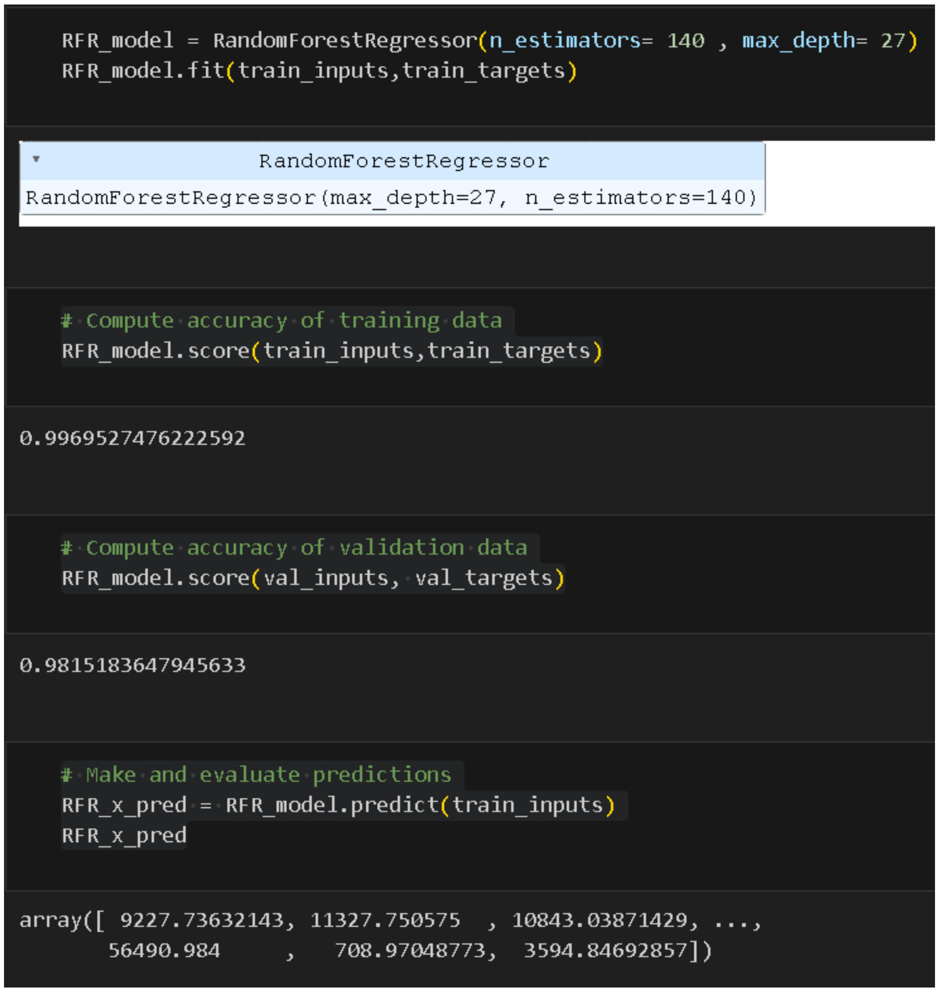
```
# Python code snippet for evaluating Random Forest model
```



```python
# Plot actual vs. predicted sales for Random Forest model
plt.figure(figsize=(8, 6))
plt.scatter(y_val, rf_val_preds)
plt.plot([0, max(y_val)], [0, max(y_val)], 'r-', lw=2)
plt.title("Actual vs. Predicted Sales (Random Forest)")
plt.xlabel("Actual Sales")
plt.ylabel("Predicted Sales")
plt.show()
```

Figure 12: Actual vs. Predicted Sales (Random Forest) Code

Execution...



```
RFR_model = RandomForestRegressor(n_estimators= 140 , max_depth= 27)
RFR_model.fit(train_inputs,train_targets)
```

```
RandomForestRegressor
RandomForestRegressor(max_depth=27, n_estimators=140)
```

```
# Compute accuracy of training data
RFR_model.score(train_inputs,train_targets)
```

```
0.9969527476222592
```

```
# Compute accuracy of validation data
RFR_model.score(val_inputs, val_targets)
```

```
0.9815183647945633
```

```
# Make and evaluate predictions
RFR_x_pred = RFR_model.predict(train_inputs)
RFR_x_pred
```

```
array([ 9227.73632143, 11327.750575  , 10843.03871429, ...,
        56490.984     ,   708.97048773,  3594.84692857])
```

Figure 13: Actual vs. Predicted Sales (Random Forest)

8 Conclusion

The Walmart Sales Prediction project demonstrates the application of machine learning techniques to forecast weekly sales accurately. By analyzing various features and employing advanced regression models like Random Forest, we can optimize sales predictions and assist in strategic decision-making for Walmart stores. Further model refinement and feature engineering could potentially enhance prediction accuracy and provide more valuable insights for the retail industry.