WALMART DATASET ANALYSIS

Team Number: 389

Members: Aasika S Kunal Kalra Riddhika J

1. **Introduction**

1.1 Overview

Our project involves predicting Walmart sales for a given week using regression techniques. Regression is a statistical method used to model the relationship between a dependent variable (sales in this case) and one or more independent variables (such as temperature, CPI, holiday, etc.).The goal of your project is to develop a predictive model that can accurately estimate Walmart sales based on historical data. By analyzing the relationships and patterns within the data, the model will be able to make predictions for future weeks.

1.2 Purpose

The purpose of this project is to develop a regression-based predictive model for estimating Walmart's weekly sales. By analyzing historical data and incorporating relevant factors like promotions, holidays, and weather conditions, the model aims to provide accurate sales predictions. This can help Walmart optimize inventory management, plan promotions effectively, and make data-driven decisions to enhance overall sales performance.

2. **Literature Survey**

Existing Solution: Predicting Sales in Retail Using Machine Learning Techniques" by Shixing Yan, Hongxu Chen, and Zhehan Yi. This paper explores the use of various regression techniques, including linear regression, decision trees, and support vector regression, for predicting sales in the retail industry. It discusses feature selection, model evaluation, and compares the performance of different algorithms.

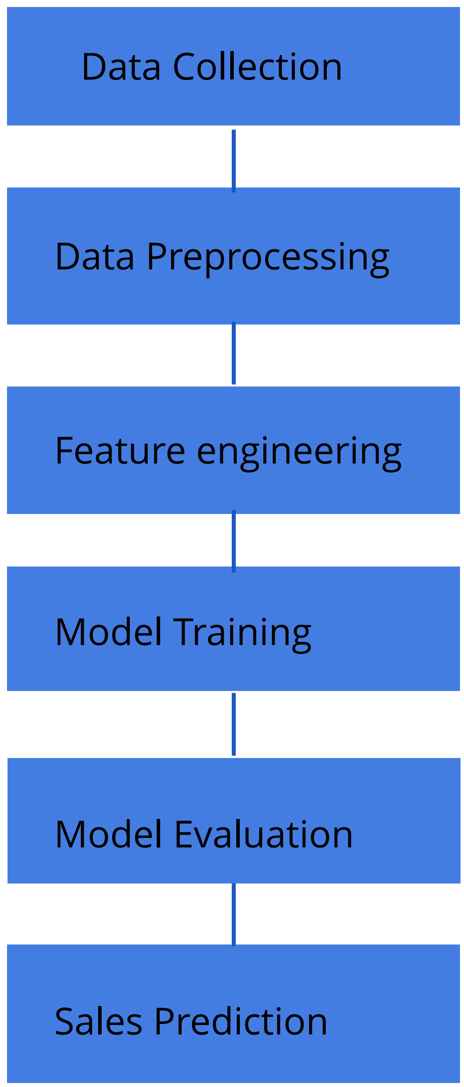
Predicting Sales for Retail Chain Stores: A Case Study of Walmart Inc." by Prakash Khadka and Qiang Pan. This research paper presents a case study on sales prediction for Walmart stores. It discusses the use of time series analysis and regression techniques, including linear regression and decision trees, to forecast sales accurately.

Proposed solution:

We tried to predict the sales of the Walmart using regressor called as random forest. We also performed feature engineering such as removing moderate dependency, standardization techniques to improve the model's accuracy in predicting the result. This predicted the data with good accuracy.

3. Theoretical Analysis

3.1 Block diagram



3.2 Hardware and software designing

Hardware Requirements: **Computer:** A computer system with sufficient processing power and memory to handle data preprocessing, model training, and evaluation **Storage:** Adequate storage capacity to store the dataset, intermediate files, and the trained models.

Software Requirements:

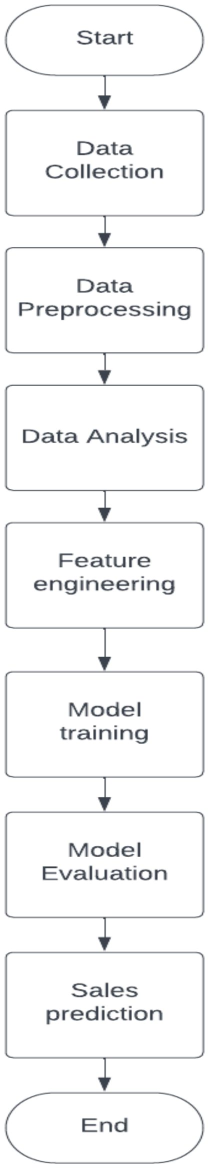
1. **Python:** Python is a popular programming language for data analysis and machine learning.
2. **Python Libraries:** Python libraries such as Pandas for data manipulation, NumPy for numerical operations, scikit-learn for regression modeling, and Matplotlib or Seaborn for data visualization.
3. **Integrated Development Environment (IDE):** We used spyder IDE to deploy our model
4. **Data Analysis and Visualization Tools:** We used Microsoft Excel for initial data exploration and analysis.
5. **Version Control:** We used Git ti track changes and collaborate on our project
6. **Documentation:** Zoho docs was used for documentation.
7. **Flask:** Flask library was used to deploy the model

4. Experimental Investigations

Data Analysis:

We made various visual analysis such as histplot, boxplot, subplot, pairplot, joinplot and heatmap to identify the correlation between the variables. Descriptive statistical analysis such as median ,mode, skew, kurtosis and standard deviation was made for various fields in the dataset.

5. Flowchart



6. Result

We predicted the sales using random forest regressor. It gave us a good accuracy. Our findings revealed that certain factors, such as temperature and fuel price, had a significant impact on sales, while others, like cpi , showed a moderate influence.

The following picture shows the evaluation metrics of our regressor.

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print("R2 score =", r2_score(ytest, ypred))
print("MSE =",mean_squared_error(ytest, ypred))
print("RMSE =", (mean_squared_error(ytest, ypred))**0.5)
print("MAE =",mean_absolute_error(ytest, ypred))
```

```
R2 score = 0.933590611628628
MSE = 21327120848.369434
RMSE = 146038.08013107209
MAE = 75651.95067545056
```

7. Advantages and disadvantages

Advantages:

1. **Improved Decision-making:** By accurately predicting Walmart's weekly sales, the project enables better decision-making within the company. It helps in inventory management, resource allocation, and strategic planning, leading to more informed business decisions.
2. **Efficient Resource Utilization:** Accurate sales predictions allow Walmart to optimize its resources, such as staffing, inventory levels, and marketing efforts. This can help minimize waste, reduce costs, and improve overall operational efficiency.
3. **Targeted Marketing and Promotions:** Accurate sales predictions enable Walmart to identify trends and customer preferences. This information can be utilized to tailor marketing campaigns and promotions, resulting in more targeted and effective customer engagement.

Disadvantages:

1. **Dynamic Market Conditions:** Market dynamics and external factors such as economic fluctuations, seasonal variations, or unforeseen events can impact sales patterns. The model's accuracy may be affected by changes in consumer behavior or market conditions that are not fully captured by historical data.
2. **Model Maintenance and Updates:** ML models require continuous monitoring, maintenance, and updates. As Walmart's business and market evolve, the model needs to be regularly re-evaluated and retrained to ensure its predictions remain accurate and relevant.

8. Applications

1. **Pricing and Promotion Strategies:** Sales predictions can aid in determining optimal pricing and promotion strategies. By understanding customer demand patterns and price elasticity, Walmart can make informed decisions about product pricing, discounts, and promotional campaigns to maximize sales and profitability.
2. **Supply Chain Optimization:** Sales predictions play a vital role in supply chain optimization. Walmart can use the predictions to optimize procurement, production, and logistics processes. This includes streamlining supplier relationships, improving transportation and warehousing operations, and reducing lead times, ultimately enhancing supply chain efficiency.

9. Conclusion

In conclusion, our project focused on predicting Walmart sales for a given week using regression techniques. We explored different regression techniques, including linear regression, decision trees, random forests, and neural networks, and evaluated their performance using appropriate metrics such as R2 score, mean squared error (MSE) and mean absolute error (MAE) and we found that randomforest regressor predicts the test data well. Overall, our project successfully demonstrated the potential of regression techniques in predicting Walmart sales.

10. Future Scope

In future, we can implement mechanisms to update the regression model periodically to incorporate new data and adapt to changing trends and seasonality patterns. This can help maintain the model's accuracy and effectiveness over time. We can integrate this sales prediction model with supply chain management systems to optimize inventory levels, improve demand forecasting, and streamline logistics operations. This can lead to more efficient inventory management and cost savings.

11. Bibliography

References:

[1] Niu, Y. (2020, October). Walmart sales forecasting using xgboost algorithm and feature engineering. In *2020 International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)* (pp.

458-461). IEEE.

[2] Mounika, S., Sahithi, Y., Grishmi, D., Sindhu, M., & Ganesh, P. (2021). Walmart Gross Sales Forecasting Using Machine Learning. *J. Adv. Res. Technol. Manag. Sci*, *3*, 22-27.

A. Appendix

Source Code:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [29]:

```
data = pd.read_csv('Walmart.csv')
data.head()
```

Out[29]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106
2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

In [3]:

```
data.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6435 entries, 0 to 6434
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype
```

```
---  ------        -------------  -----
 0   Store         6435 non-null   int64
 1   Date          6435 non-null   object
 2   Weekly_Sales  6435 non-null   float64
 3   Holiday_Flag  6435 non-null   int64
 4   Temperature   6435 non-null   float64
 5   Fuel_Price    6435 non-null   float64
 6   CPI           6435 non-null   float64
 7   Unemployment  6435 non-null   float64
dtypes: float64(5), int64(2), object(1)
memory usage: 402.3+ KB
```

In [4]:

```
data.describe()
```

Out[4]:

	Store	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
count	6435.000000	6.435000e+03	6435.000000	6435.000000	6435.000000	6435.000000	6435.000000
mean	23.000000	1.046965e+06	0.069930	60.663782	3.358607	171.578394	7.999151
std	12.988182	5.643666e+05	0.255049	18.444933	0.459020	39.356712	1.875885
min	1.000000	2.099862e+05	0.000000	-2.060000	2.472000	126.064000	3.879000
25%	12.000000	5.533501e+05	0.000000	47.460000	2.933000	131.735000	6.891000
50%	23.000000	9.607460e+05	0.000000	62.670000	3.445000	182.616521	7.874000
75%	34.000000	1.420159e+06	0.000000	74.940000	3.735000	212.743293	8.622000
max	45.000000	3.818686e+06	1.000000	100.140000	4.468000	227.232807	14.313000

In [5]:

```
data.median()
```

Out[5]:

```
Store              23.000000
Weekly_Sales    960746.040000
Holiday_Flag         0.000000
Temperature         62.670000
```

```
Fuel_Price       3.445000
CPI            182.616521
Unemployment     7.874000
dtype: float64
```

In [6]:

```
data.skew()
```

Out[6]:

```
Store          0.000000
Weekly_Sales   0.668362
Holiday_Flag   3.373499
Temperature   -0.336768
Fuel_Price    -0.096158
CPI            0.063492
Unemployment   1.188144
dtype: float64
```

In [7]:

```
data.kurtosis()
```

Out[7]:

```
Store         -1.201187
Weekly_Sales   0.053141
Holiday_Flag   9.383410
Temperature   -0.612801
Fuel_Price    -1.177378
CPI           -1.839813
Unemployment   2.639712
dtype: float64
```

In [8]:

```
data.isnull().sum()
```

Out[8]:

```
Store          0
Date           0
Weekly_Sales   0
Holiday_Flag   0
Temperature    0
Fuel_Price     0
CPI            0
Unemployment   0
dtype: int64
```

In [9]:

```
plt.subplot(1,2,1)
sns.histplot(x=data['Weekly_Sales'])
plt.subplot(1,2,2)
sns.countplot(x=data['Holiday_Flag'])
```

Out[9]:

```
<Axes: xlabel='Holiday_Flag', ylabel='count'>
```

In [10]:

```
plt.subplot(1,2,1)
sns.distplot(x=data['Temperature'])
plt.subplot(1,2,2)
sns.distplot(x=data['Fuel_Price'])
```

Out[10]:

```
<Axes: ylabel='Density'>
```

In [11]:

```
plt.subplot(1,2,1)
sns.distplot(x=data['CPI'])
plt.subplot(1,2,2)
sns.distplot(x=data['Unemployment'])
```

Out[11]:

```
<Axes: ylabel='Density'>
```

In [12]:

```
sns.pairplot(data)
```

Out[12]:

```
<seaborn.axisgrid.PairGrid at 0x1db99395300>
```

In [13]:

```
sns.heatmap(data.corr() ,annot=True)
```

Out[13]:

```
<Axes: >
```

In [14]:

```
plt.figure(figsize = (10,10))
plt.subplot(2,2,1)
sns.boxplot(x=data['Temperature'])
plt.subplot(2,2,2)
sns.boxplot(x=data['Fuel_Price'])
plt.subplot(2,2,3)
sns.boxplot(x=data['CPI'])
plt.subplot(2,2,4)
sns.boxplot(x=data['Unemployment'])
```

Out[14]:

```
<Axes: xlabel='Unemployment'>
```

In [15]:

```
sns.jointplot(x=data['Temperature'] ,y=data['Weekly_Sales'])
```

Out[15]:

```
<seaborn.axisgrid.JointGrid at 0x1db9e3b81f0>
```

In [16]:

```
p5 = np.percentile(data.Temperature, 1)
print(p5)
p = np.where(data.Temperature<p5,p5, data.Temperature)
data.Temperature = p
sns.boxplot(x = data.Temperature)
```

18.523600000000002

Out[16]:

<Axes: xlabel='Temperature'>

In [17]:

```
sns.jointplot(x=data['Unemployment'] ,y=data['Weekly_Sales'])
```

Out[17]:

<seaborn.axisgrid.JointGrid at 0x1db9e8add80>

In [18]:

```
p5 = np.percentile(data.Unemployment, 2)
p90 = np.percentile(data.Unemployment, 94)
print(p5, p90)
p = np.where(data.Unemployment<p5, p5, data.Unemployment)
p= np.where(p>p90, p90, p)

data.Unemployment = p
sns.boxplot(x = data.Unemployment)
```

4.42 10.926

Out[18]:

<Axes: xlabel='Unemployment'>

In [19]:

```
data.head()
```

Out[19]:

	Store	Date	Weekly_Sales	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	1	05-02-2010	1643690.90	0	42.31	2.572	211.096358	8.106
1	1	12-02-2010	1641957.44	1	38.51	2.548	211.242170	8.106

2	1	19-02-2010	1611968.17	0	39.93	2.514	211.289143	8.106
3	1	26-02-2010	1409727.59	0	46.63	2.561	211.319643	8.106
4	1	05-03-2010	1554806.68	0	46.50	2.625	211.350143	8.106

In [20]:

```
x = data.drop(columns=['Weekly_Sales', 'Date'], axis=1)
y = data.Weekly_Sales
```

Scaling the dataset

In [21]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(x)
a = scaler.transform(x)
x = pd.DataFrame(a, columns = x.columns)
x.head()
```

Out[21]:

	Store	Holiday_Flag	Temperature	Fuel_Price	CPI	Unemployment
0	0.0	0.0	0.291441	0.050100	0.840500	0.566554
1	0.0	1.0	0.244882	0.038076	0.841941	0.566554
2	0.0	0.0	0.262281	0.021042	0.842405	0.566554
3	0.0	0.0	0.344372	0.044589	0.842707	0.566554
4	0.0	0.0	0.342779	0.076653	0.843008	0.566554

Splitting the dataset

In [22]:

```
from sklearn.model_selection import train_test_split
```

```
xtrain,xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state = 0)
```

Lasso Regression

In [23]:

```
from sklearn.linear_model import Lasso
# Create a Lasso regression model
lasso = Lasso(alpha=0.1)  # Set the regularization parameter alpha

# Fit the model on the training data
lasso.fit(xtrain, ytrain)

# Predict on the test data
ypred = lasso.predict(xtest)

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print("R2 score =", r2_score(ytest, ypred))
print("MSE =",mean_squared_error(ytest, ypred))
print("RMSE =", (mean_squared_error(ytest, ypred))**0.5)
print("MAE =",mean_absolute_error(ytest, ypred))
```

```
R2 score = 0.15731293626158727
MSE = 270625724561.7728
RMSE = 520216.9975709875
MAE = 426198.8308701095
```

Ridge regression

In [24]:

```
from sklearn.linear_model import Ridge

# Create a Ridge regression model
ridge = Ridge(alpha=0.1)  # Set the regularization parameter alpha

# Fit the model on the training data
ridge.fit(xtrain, ytrain)

# Predict on the test data
ypred = ridge.predict(xtest)

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print("R2 score =", r2_score(ytest, ypred))
print("MSE =",mean_squared_error(ytest, ypred))
print("RMSE =", (mean_squared_error(ytest, ypred))**0.5)
print("MAE =",mean_absolute_error(ytest, ypred))
```

```
R2 score = 0.15730763657348756
MSE = 270627426536.31787
```

RMSE = 520218.63339976384
MAE = 426201.56849984237

Decision Tree regression

In [25]:

```
from sklearn.tree import DecisionTreeRegressor

# Create a Decision Tree Regressor object
tree_reg = DecisionTreeRegressor()

# Fit the model to the training data
tree_reg.fit(xtrain, ytrain)

# Predict on the test data
ypred = tree_reg.predict(xtest)

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print("R2 score =", r2_score(ytest, ypred))
print("MSE =",mean_squared_error(ytest, ypred))
print("RMSE =", (mean_squared_error(ytest, ypred))**0.5)
print("MAE =",mean_absolute_error(ytest, ypred))
```

R2 score = 0.8983535197050893
MSE = 32643378025.681038
RMSE = 180674.78525151493
MAE = 92099.16580484773

RandomForestRegressor

In [26]:

```
from sklearn.ensemble import RandomForestRegressor

# create regressor object
rand_forest = RandomForestRegressor(n_estimators=100, random_state=0)

# fit the regressor with x and y data
rand_forest.fit(xtrain, ytrain)
ypred = rand_forest.predict(xtest)

from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print("R2 score =", r2_score(ytest, ypred))
print("MSE =",mean_squared_error(ytest, ypred))
print("RMSE =", (mean_squared_error(ytest, ypred))**0.5)
print("MAE =",mean_absolute_error(ytest, ypred))
```

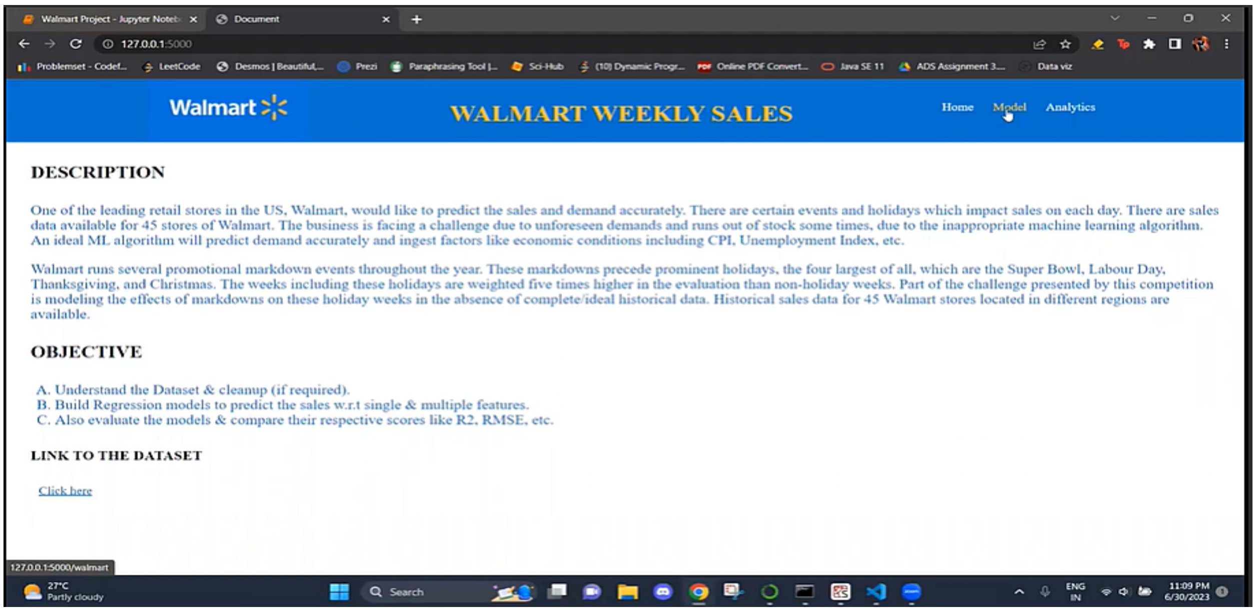
R2 score = 0.9340478577551995
MSE = 21180278005.241497

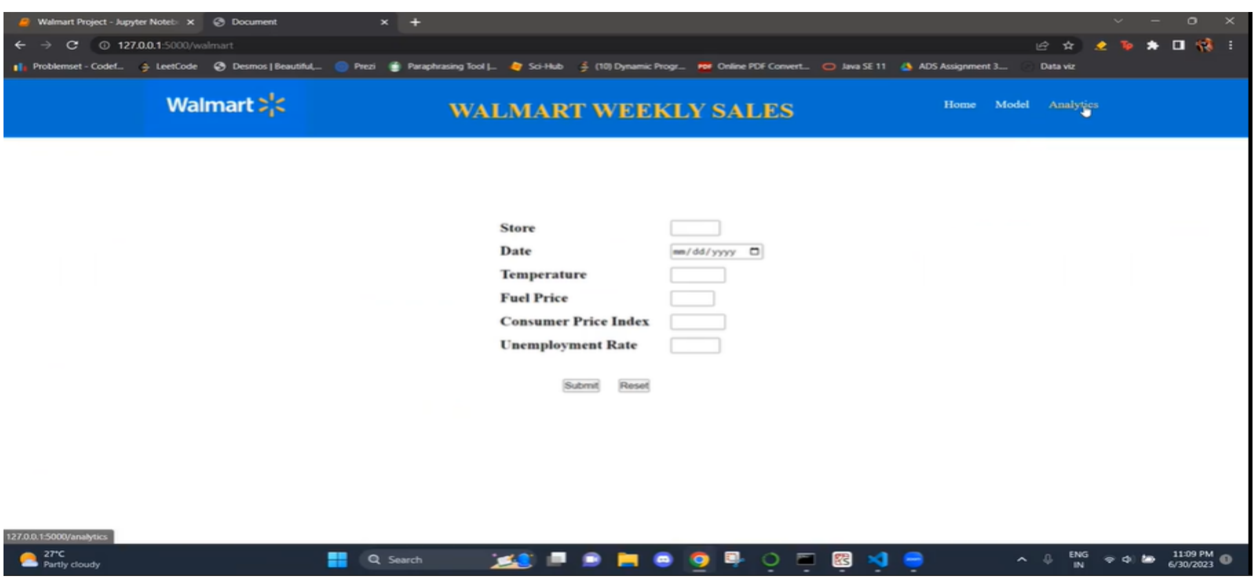
RMSE = 145534.45641923253
MAE = 75987.74288353013

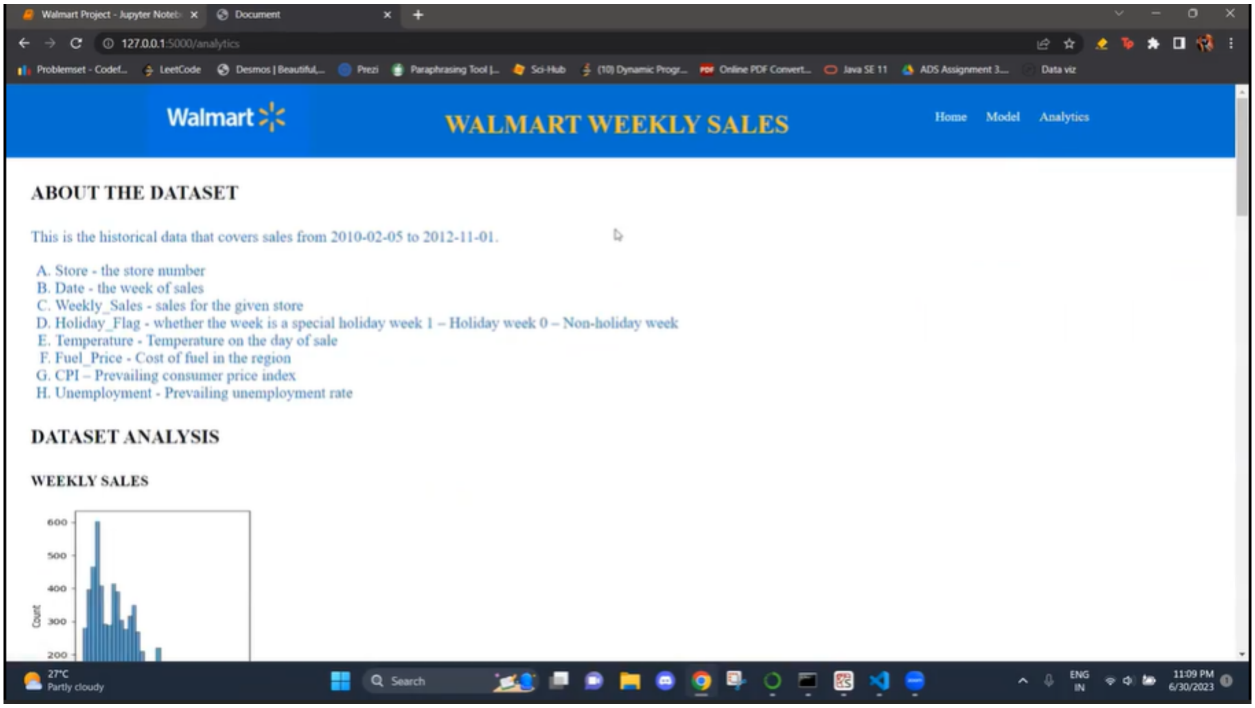
In [27]:

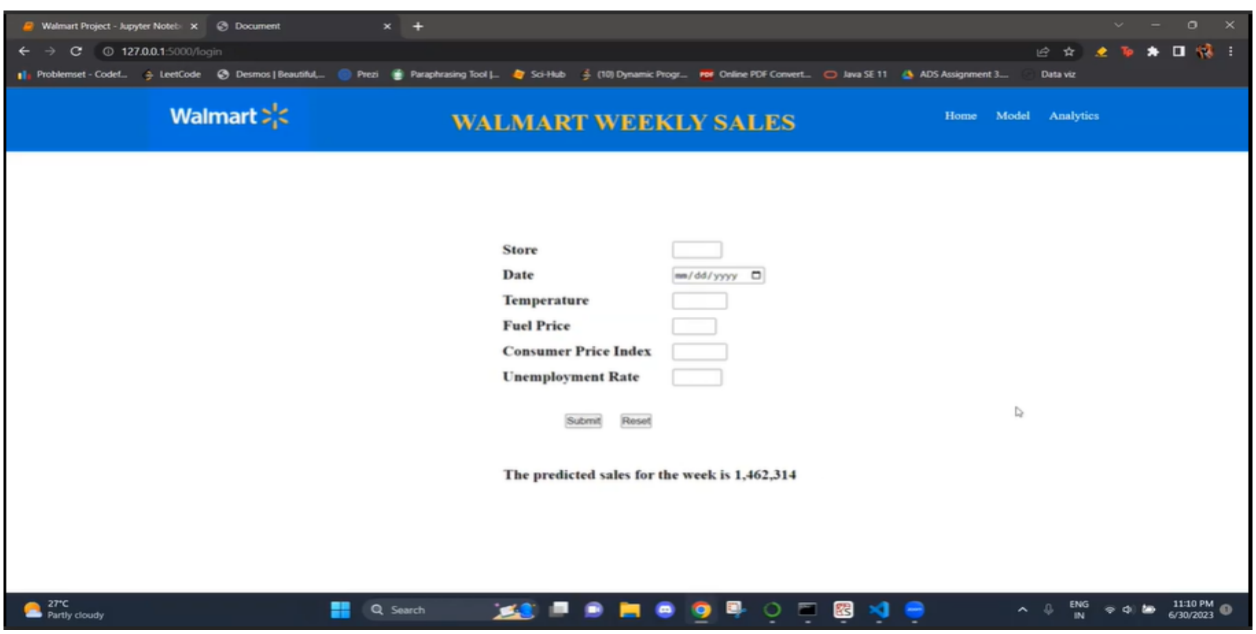
```
import pickle
pickle.dump(rand_forest,open("model.pkl","wb"))
pickle.dump(scaler,open("scaler.pkl", "wb"))
```

FINAL RESULT:






Walmart Project - Jupyter Notebook
Document
127.0.0.1:5000/analytics
Walmart
WALMART WEEKLY SALES
Home
Model
Analytics
ABOUT THE DATASET
This is the historical data that covers sales from 2010-02-05 to 2012-11-01.
A. Store - the store number
B. Date - the week of sales
C. Weekly_Sales - sales for the given store
D. Holiday_Flag - whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week
E. Temperature - Temperature on the day of sale
F. Fuel_Price - Cost of fuel in the region
G. CPI – Prevailing consumer price index
H. Unemployment - Prevailing unemployment rate
DATASET ANALYSIS
WEEKLY SALES
600
500
400
300
200
Count
27°C
Partly cloudy
Search
11:09 PM
6/30/2023


Walmart Project - Jupyter Notebook
Document
127.0.0.1:5000/login
Walmart
WALMART WEEKLY SALES
Home
Model
Analytics
Store
Date
mm/dd/yyyy
Temperature
Fuel Price
Consumer Price Index
Unemployment Rate
Submit
Reset
The predicted sales for the week is 1,462,314
27°C
Partly cloudy
Search
11:10 PM
6/30/2023