Walmart sales analysis for retail industry

Submitted By

Thamjidha T N

1.INTODUCTION

1.1PROJECT OVERVIEW

Sales forecasting is the process of estimating future sales. Accurate sales forecasts enable companies to make informed business decisions and predict short-term and long-term performance. Companies can base their forecasts on past sales data, industry-wide comparisons, and economic trends. Here, the company is Walmart. Walmart is a renowned retail corporation that operates a chain of hypermarkets. Walmart has provided data by combining the data of 45 stores including store information and monthly sales. Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are 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. The data is provided on weekly basis. We have to find the impact of holidays on the sales of the store. The holidays included are Christmas, Thanksgiving, Super Bowl and Labour Day. We will be using algorithms such as Random Forest, Decision tree, XgBoost and ARIMA. We will train and test the data with these algorithms. Flask integration and IBM deployment will also be done.

1.2PURPOSE

The purpose of this study is to predict the weekly sales for Walmart based on available historical data stores located in different regions around the country. Each store contains a number of departments and the main deliverable is to predict the weekly sales for all such departments. The data has been collected from Kaggle and contains the weekly sales for 45 stores, the size and type of store, department information for each of those stores, the amount of weekly sales, and whether the week is a holiday week or not. There is additional information in the dataset about the factors that might influence the sales of a particular week. Factors like Consumer Price Index (CPI), temperature, fuel price, promotional markdowns for the week, and unemployment rate have been recorded for each week to try and understand if there is a correlation between the sales of each week and their determinant factors

Correlation testing has been performed to understand if there is a correlation between the individual factors and weekly sales and whether such factors have any impact on sales made by Walmart. This

study also includes an extensive exploratory data analysis on the provided Walmart dataset to understand the following:

- Identifying store as well as department-wide sales in Walmart
- Identifying sales based on store size and type
- Identifying how much sales increase during holidays
- Correlation between the different factors that affect sales
- Average sales per year
- Weekly sales as per region temperature, CPI, fuel price, unemployment

Apart from identifying these direct relationships between independent and dependent variables, some interaction effects have also been studied as part of the Multiple Linear Regression model to understand if a certain combination of the factors under study can directly impact the weekly sales for Walmart. After employing different algorithms to predict future sales and correlation between factors for the retail store, a dashboard that tracks the above-mentioned outcomes has been created (in Power BI) and also includes the new predictions to collectively visualize the outcomes of this research and present them to amateur users more effectively.

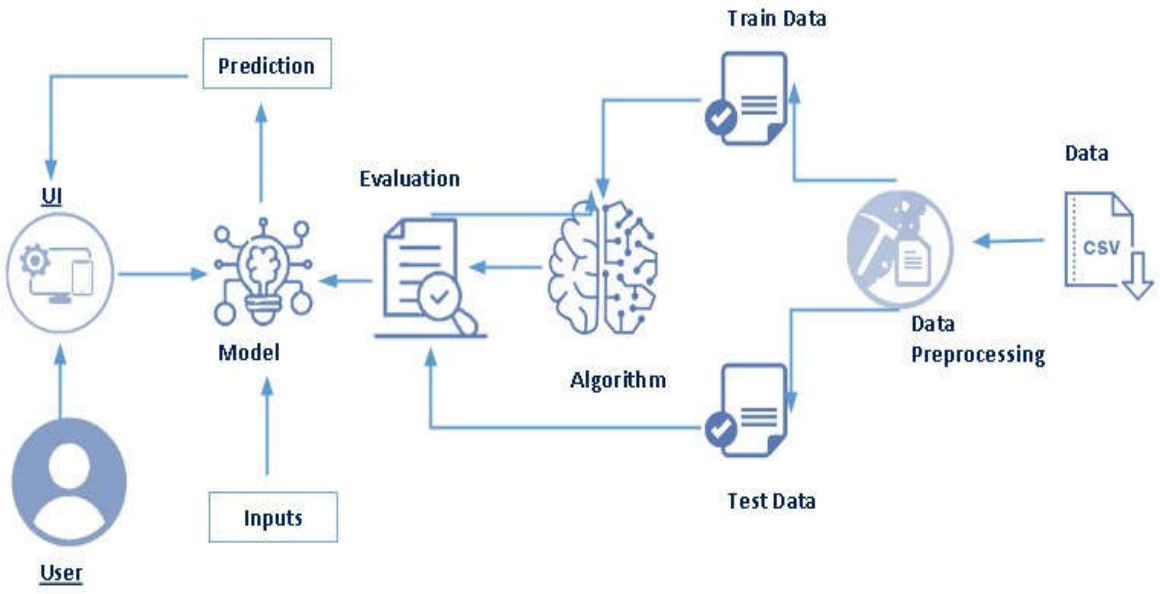
2. LITERATURE SURVEY

PROPOSED SYSTEM

The paper comprises of several different components that explore various aspects of the 45 Walmart stores used in this study. The methodology section is broken down into several sub-sections that follow a 'top-down' approach of the process that is followed in this analysis. This section contains detailed information about the dataset, the exact techniques that have been used in forecasting weekly sales and the last section talks about how this study is significant in predicting the weekly sales for Walmart stores. It will also discuss the success of the applied models in identifying the effect of different factors on such weekly sales.

3. THEORITICAL ANALYSIS

Architecture



Project Flow

The data is loaded and cleaning and feature extraction are performed on the data.

- The model is trained using machine learning models.
- Once the model is trained, it is tested and evaluation is performed.

To accomplish this, we have to complete all the activities listed below,

- Data collection
  - Collect the dataset or create the dataset
- Visualizing and analyzing data
  - Descriptive analysis
- Data pre-processing
  - Checking for null values
  - Filling null values
  - Splitting data into train and test
- Model building

- Import the model-building libraries
- Initializing the model
- Training and testing the model
- Evaluating the performance of the model
- Save the model
-

HARDWARE AND SOFTWARE REQUIREMENTS IN THE PROJECT

For running a machine learning model on the system you need a system with minimum of 16 GBRAM in it and you require a good processor for high performance of the model**.**
In the list of **Software requirements** you must have:
● Jupyter Notebook for programming, which can be installed by Anaconda IDE.
● Python packages.
● A better software for running the html and css files for application building phase e.g. spyder.

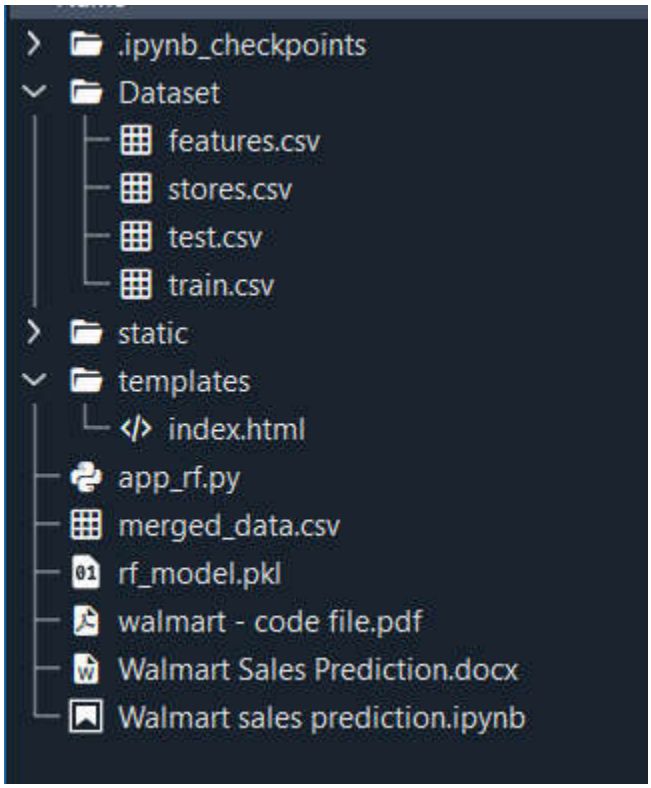
4.EXPERIMENTALI NVESTIGATIONS

Data Pre-Processing

As we have seen and understood the description of the data, let's pre-process the collected data.
The download data set is not suitable for training the machine learning model as it might have so much of randomness so, the dataset has to be cleaned properly in order to fetch good results. This activity includes the following steps.

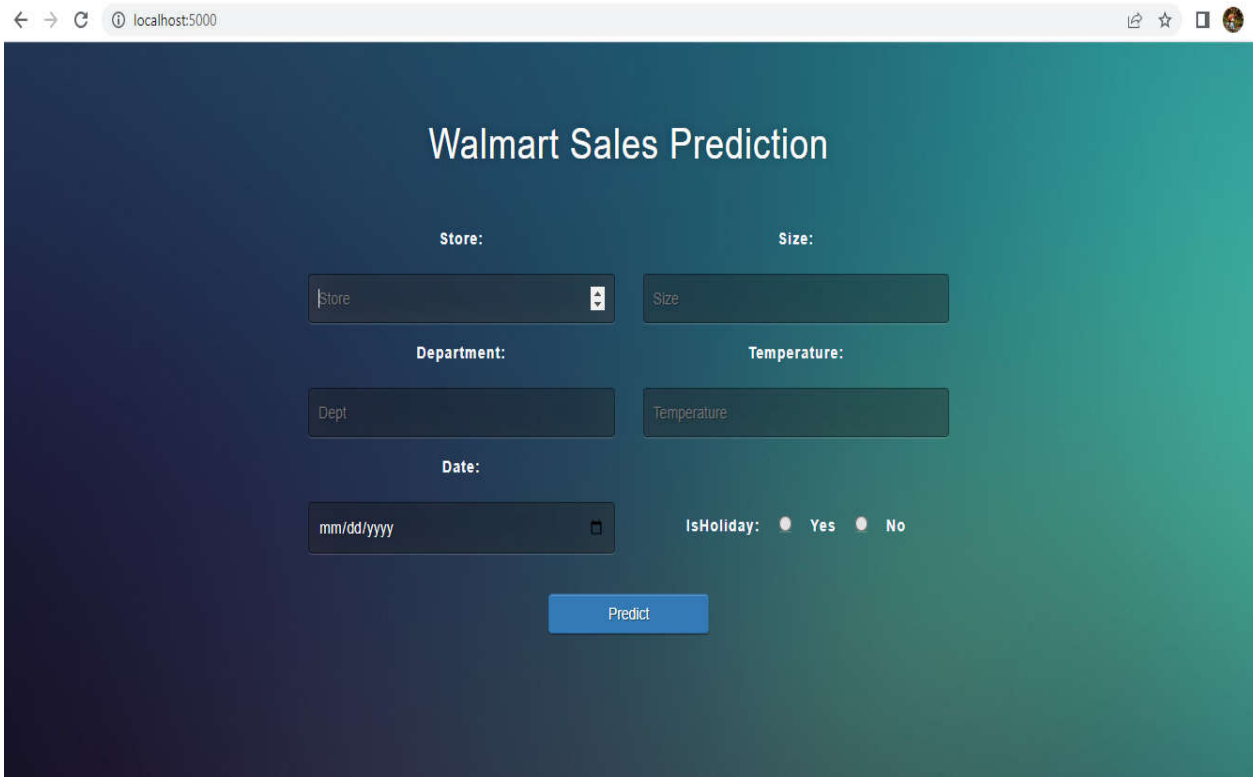
- Handling missing values
- Handling categorical data
- Handling outliers
- Scaling Techniques
- Splitting dataset into training and test set

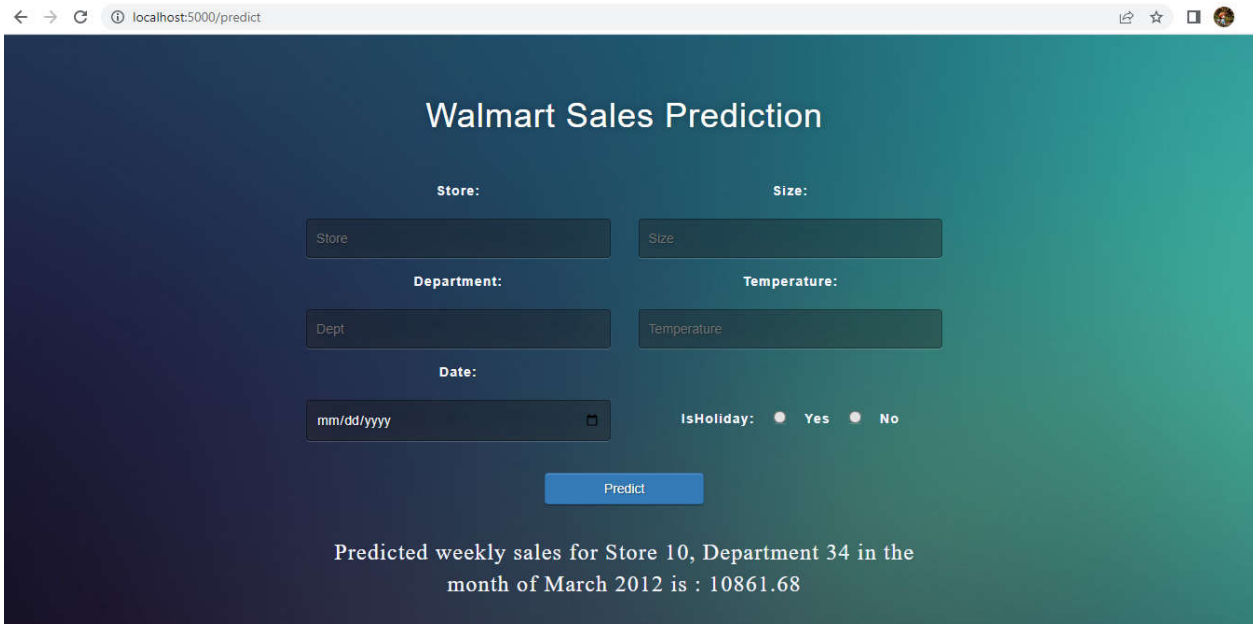
5.FLOWCHART



6. RESULTS

Final output of the project:


localhost:5000
Walmart Sales Prediction
Store:
Size:
Store
Size
Department:
Temperature:
Dept
Temperature
Date:
mm/dd/yyyy
IsHoliday:
Yes
No
Predict


localhost:5000/predict
Walmart Sales Prediction
Store:
Size:
Store
Size
Department:
Temperature:
Dept
Temperature
Date:
mm/dd/yyyy
IsHoliday:
Yes
No
Predict
Predicted weekly sales for Store 10, Department 34 in the month of March 2012 is : 10861.68

7. ADVANTAGES

• Identifying store as well as department-wide sales in Walmart

• Identifying sales based on store size and type

• Identifying how much sales increase during holidays

• Correlation between the different factors that affect sales

•Average sales per year

• Weekly sales as per region temperature, CPI, fuel price, unemployment

DISADVANTAGES

A huge constraint of this study is the lack of sales history data available for analysis. Because of this limited past history data, models cannot be trained as efficiently to give accurate results and predictions. Because of this lack of availability, it is harder to train and tune models as an over-constrained model might reduce the accuracy of the model. An appropriate amount of training data is required to efficiently train the model and draw useful insights. Additionally, the models created have been developed based on certain preset assumptions and business conditions. it is harder to predict the effects of certain economic, political, or social policies on the sales recorded by the organization. Also, it is tough to predict how the consumer buying behavior changes over the years or how the policies laid down by the management might affect the company's revenue. these factors can have a direct impact on Walmart sales and it is necessary to constantly study the market trends and compare them with existing performance to create better policies and techniques for increased profits.

8 APPLICATIONS

The main objective of this study is to predict weekly sales for Walmart stores and create a Power BI dashboard that tracks the final predicted sales until 2013 through interactive and immersive visualizations. The conclusion section highlights the findings from the exploratory data analysis as well as from the models implemented as part of this study. The dashboard created compares the findings from the Exploratory Data Analysis with the findings from the dashboard. The user of the dashboard can filter data based on stores, departments, store type, store size, week of the year, holiday versus nonholiday sales, etc.

9.CONCLUSION

The main purpose of this study was to predictWalmart's sales based on the available historic data and identify whether factors like temperature, unemployment, fuel prices, etc affect the weekly sales of particular stores under study. This study also aims to understand whether sales are relatively higher during holidays like Christmas and Thanksgiving than normal days so that stores can work on creating promotional offers that increase sales and generate higher revenue.

Interaction effects were studied as part of the linear regression model to identify if a combination of different factors could influence the weekly sales for Walmart. This was necessary because of the presence of a high number of predictor variables in the dataset. While the interaction effects were tested on a combination of significant variables, a statistically significant relationship was only observed between the independent variables of temperature, CPI and unemployment, and weekly sales (predictor variable). However, this is not definite because of the limitation of training data.

10.FUTURE SCOPE

With growing technology and increasing consumer demand,Walmart can shift its focus on the e-commerce aspects of the business. Taking inspiration from Amazon's business model, Walmart can grow its online retail business massively and gather huge profits. With already established stores and warehouses, it is easier for the organization to create a nationwide reach, limiting the presence of physical stores and helping their customers save on fuel costs by delivering goods at their doorstep. It also makes it a lot easier to identify consumer buying patterns. An important aspect of this study is also to try and understand customer buying behavior based on regional and departmental sales. This customer segmentation can help the organization in creating and communicating targeted messages for customers belonging to a particular region, establishing better customer relationships, focusing 57 on profitable regions, and identifying ways to improve products and services in specific regions or for specific customers. Another aspect that would be worth exploring with this study is identifying trends with sales for each of the stores and predicting future trends based on the available sales data. Time series forecasting can be utilized (ARMA and ARIMA modeling) to predict future sales for each of the stores and their respective departments

11.BIBILOGRAPHY

Bakshi, C. (2020). Random forest regression. https : / / levelup . gitconnected . com / random-forest-regression-209c0f354c84

Bari, A., Chaouchi, M., & Jung, T. (n.d.). How to utilize linear regressions in predictive analytics. https://www.dummies.com/programming/big-data/data-science/ how-to-utilize-linear-regressions-in-predictive-analytics/

Baum, D. (2011). How higher gas prices affect consumer behavior. https : / / www . sciencedaily.com/releases/2011/05/110512132426.htm

Brownlee, J. (2016). Feature importance and feature selection with xgboost in python. https : / / machinelearningmastery . com / feature - importance - and - feature - selection-with-xgboost-in-python/

Chouksey, P., & Chauhan, A. S. (2017). A review of weather data analytics using big data. International Journal of Advanced Research in Computer and Communication Engineering, 6. https://doi.org/https://ijarcce.com/upload/2017/january17/IJARCCE%2072.pdf

APPENDIX

app.py

```
import numpy as np

from numpy.core.fromnumeric import size

import pandas as pd

from sklearn.model_selection import train_test_split

from flask import Flask, render_template, request

import pickle

import datetime as dt

import calendar

import os


app = Flask(__name__)
```

```
loaded_model = pickle.load(open('rf_model.pkl','rb'))
fet = pd.read_csv('merged_data.csv')



@app.route('/')
def home():
    return render_template('index.html')



@app.route('/predict', methods=['POST'])
def predict():
    store = request.form.get('store')
    dept = request.form.get('dept')
    size=request.form.get('size')
    temp=request.form.get('temp')
    isHoliday = request.form['isHolidayRadio']
    date = request.form.get('date')
    d=dt.datetime.strptime(date, '%Y-%m-%d')
    month = d.month
    year = (d.year)
    month_name=calendar.month_name[month]
    print("year = ", type(year))
    print("year val = ", year, type(year), month)
    X_test = pd.DataFrame({'Store': [store], 'Dept': [dept],'Size':[size],
                           'Temperature':[temp], 'CPI':[212],' IsHoliday':[isHoliday],
                           'Type_A':[0], 'Type_B':[0], 'Type_C':[1],
                           'month':[month], 'year':[year]})
```

```
    print("X_test = ", X_test.head())
    print("type of X_test = ", type(X_test))
    print("predict = ", store, dept, date, isHoliday)

    y_pred = loaded_model.predict(X_test)
    output=round(y_pred[0],2)
    print("predicted = ", output)
    return render_template('index.html', output=output, store=store, dept=dept,
                           month_name=month_name, year=year)

if __name__ == "__main__":
    app.run(debug=False)
```

CSS

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn {
  display: inline-block;
  *display: inline;
  *zoom: 1;
  padding: 4px 10px 4px;
  margin-bottom: 0;
  font-size: 13px;
  line-height: 18px;
  color: #333333;
  text-align: center;
  text-shadow: 0 1px 1px rgba(255, 255, 255, 0.75);
  vertical-align: middle;
```

```
background-color: #f5f5f5;
background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6);
background-image: -ms-linear-gradient(top, #ffffff, #e6e6e6);
background-image: -webkit-gradient(
  linear,
  0 0,
  0 100%,
  from(#ffffff),
  to(#e6e6e6)
);
background-image: -webkit-linear-gradient(top, #ffffff, #e6e6e6);
background-image: -o-linear-gradient(top, #ffffff, #e6e6e6);
background-image: linear-gradient(top, #ffffff, #e6e6e6);
background-repeat: repeat-x;
filter: progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff, endColorstr=#e6e6e6, GradientType=0);
border-color: #e6e6e6 #e6e6e6 #e6e6e6;
border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.25);
border: 1px solid #e6e6e6;
-webkit-border-radius: 4px;
-moz-border-radius: 4px;
border-radius: 4px;
-webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2),
  0 1px 2px rgba(0, 0, 0, 0.05);
-moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2),
  0 1px 2px rgba(0, 0, 0, 0.05);
box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2),
  0 1px 2px rgba(0, 0, 0, 0.05);
cursor: pointer;
```

```
  *margin-left: 0.3em;
}
.btn:hover,
.btn:active,
.btn.active,
.btn.disabled,
.btn[disabled] {
  background-color: #e6e6e6;
}
.btn-large {
  padding: 9px 14px;
  font-size: 15px;
  line-height: normal;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
  border-radius: 5px;
}
.btn:hover {
  color: #333333;
  text-decoration: none;
  background-color: #e6e6e6;
  background-position: 0 -15px;
  -webkit-transition: background-position 0.1s linear;
  -moz-transition: background-position 0.1s linear;
  -ms-transition: background-position 0.1s linear;
  -o-transition: background-position 0.1s linear;
  transition: background-position 0.1s linear;
}
```

```
.btn-primary,
.btn-primary:hover {
  text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
  color: #ffffff;
}
.btn-primary.active {
  color: rgba(255, 255, 255, 0.75);
}
.btn-primary {
  background-color: #4a77d4;
  background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4);
  background-image: -ms-linear-gradient(top, #6eb6de, #4a77d4);
  background-image: -webkit-gradient(
    linear,
    0 0,
    0 100%,
    from(#6eb6de),
    to(#4a77d4)
  );
  background-image: -webkit-linear-gradient(top, #6eb6de, #4a77d4);
  background-image: -o-linear-gradient(top, #6eb6de, #4a77d4);
  background-image: linear-gradient(top, #6eb6de, #4a77d4);
  background-repeat: repeat-x;
  filter: progid:dximagetransform.microsoft.gradient(startColorstr=#6eb6de, endColorstr=#4a77d4,
GradientType=0);
  border: 1px solid #3762bc;
  text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.4);
  box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2),
```

```
    0 1px 2px rgba(0, 0, 0, 0.5);
}
.btn-primary:hover,
.btn-primary:active,
.btn-primary.active,
.btn-primary.disabled,
.btn-primary[disabled] {
  filter: none;
  background-color: #4a77d4;
}
.btn-block {
  width: 25%;
  display: block;
  margin: auto;
}

* {
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  -ms-box-sizing: border-box;
  -o-box-sizing: border-box;
  box-sizing: border-box;
}

html {
  width: 100%;
  height: 100%;
  overflow: hidden;
```

```
}

body {
  width: 100%;
  height: 100%;
  font-family: "Open Sans", sans-serif;
  color: #fff;
  font-size: 18px;
  text-align: center;
  letter-spacing: 1.2px;
  background-image: url(../images/BackgroundImg.jpg);
}
.login {
  position: absolute;
  top: 30%;
  left: 35%;
  margin: -150px 0 0 -150px;
  width: 50%;
  height: 50%;
  color: ghostwhite;
}

.login h1 {
  color: #fff;
  text-shadow: 0 0 10px rgba(0, 0, 0, 0.3);
  letter-spacing: 1px;
  text-align: center;
}
```

```
input {
  width: 100%;
  margin-bottom: 10px;
  background: rgba(0, 0, 0, 0.3);
  border: none;
  outline: none;
  padding: 10px;
  font-size: 13px;
  color: #fff;
  text-shadow: 1px 1px 1px rgba(0, 0, 0, 0.3);
  border: 1px solid rgba(0, 0, 0, 0.3);
  border-radius: 4px;
  box-shadow: inset 0 -5px 45px rgba(100, 100, 100, 0.2),
    0 1px 1px rgba(255, 255, 255, 0.2);
  -webkit-transition: box-shadow 0.5s ease;
  -moz-transition: box-shadow 0.5s ease;
  -o-transition: box-shadow 0.5s ease;
  -ms-transition: box-shadow 0.5s ease;
  transition: box-shadow 0.5s ease;
}
input:focus {
  box-shadow: inset 0 -5px 45px rgba(100, 100, 100, 0.4),
    0 1px 1px rgba(255, 255, 255, 0.2);
}
```

Html

```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Walmart Sales Prediction</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
<link rel="icon" type="image/png" href="./static/images/walmartIcon.png">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>



</head>

<body>
<div>
 <div class="login" style="top:30%">
            <h1 style="margin-bottom:35px">Walmart Sales Prediction</h1>

    <form method="post" action="{{url_for('predict')}}">

      <div class="row">
      <div class="col-sm-6">
```

```
    <label for="store" style="margin-top:15px">Store:</label>

        <input type="number" name="store" placeholder="Store" style="margin-top:15px">


    <label for="dept" style="margin-top:15px">Department:</label>

    <input type="number" name="dept" placeholder="Dept" style="margin-top:15px">


    <label for="date" style="margin-top:15px">Date:</label>

    <input type="date" id="date" name="date" style="margin-top:15px">


  </div>


  <div class="col-sm-6">


    <label for="size" style="margin-top:15px">Size:</label>

    <input type="number" id="size" name="size" placeholder="Size" style="margin-top:15px">


    <label for="temp" style="margin-top:15px">Temperature:</label>

    <input type="number" id="temp" name="temp" placeholder="Temperature" style="margin-top:15px">


    <label for="isholiday" style="margin-top: 20%;">IsHoliday:</label>

  <input class="form-check-input" type="radio" value="1" name="isHolidayRadio" id="yes" style="width:auto;
margin-left:15px">

  <label class="form-check-label" for="yes" style="margin-left:15px"> Yes </label>

  <input class="form-check-input" type="radio" value="0" name="isHolidayRadio" id="no" style="width:auto;
margin-left:15px">

  <label class="form-check-label" for="no" style="margin-left:15px"> No </label>


  </div>
```

```
    </div>

      <button type="submit" class="btn btn-primary btn-block btn-large" style="width: 25%; margin: auto; margin-top: 5%;">Predict</button>


    </form>


    {% if output %}

   <div style="font-size: x-large; color: ghostwhite; font-family: serif; margin-top: 5%;">

      <p id="text">Predicted weekly sales for Store {{store}}, Department {{dept}} in the month of {{month_name}} {{year}} is : {{ output }}</p>

   </div>

    {% endif %}

  </div>


 </div>


 </body>

</html>
```

.ipynb

```
import numpy as np

import pandas as pd

import scipy.stats as stats

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.metrics import mean_squared_error, mean_absolute_error
```

```
from datetime import datetime
import math
train=pd.read_csv(r'train.csv')
features=pd.read_csv(r'features.csv')
stores = pd.read_csv(r'stores.csv')
train.head()
features.head()
stores.head()
train.describe()
features.describe()
stores.describe()
train.info()
features.info()
stores.info()
train.isnull().sum()
features.isnull().sum()
stores.isnull().sum()
data = train.merge(features, on=['Store', 'Date', 'IsHoliday'],
                   how='inner').merge(stores, on=['Store'], how='inner')
print(data.shape)
data.describe()
data.info()
data = data[data['Weekly_Sales'] >= 0]
data.describe()
sorted_type = stores.groupby('Type')
plt.style.use('ggplot')
```

```
labels=['A store','B store','C store']
sizes=sorted_type.describe()['Size'].round(1)
sizes=[(22/(17+6+22))*100,(17/(17+6+22))*100,(6/(17+6+22))*100] # convert to the proportion
fig, axes = plt.subplots(1,1, figsize=(7,7))

axes.pie(sizes,
        labels=labels,
        explode=(0.0,0,0),
        autopct='%1.1f%%',
        pctdistance=0.6,
        labeldistance=1.2,
        radius=0.8,
        center=(0.5,0.5))
plt.show()
# Weekly Sales on Holidays
holiday = train['Weekly_Sales'].loc[train['IsHoliday']== True]

#Weekly Sales on Non-holidays.
non_holiday = train['Weekly_Sales'].loc[train['IsHoliday']== False]
sns.barplot(x='IsHoliday', y='Weekly_Sales', data=train)

plt.figure(figsize=(28,14))
plt.xticks( fontsize=20)
plt.yticks( fontsize=20)

sns.heatmap(data.corr(), cmap='Reds', annot=True, annot_kws={'size':12})
```

```
plt.title('Correlation Matrix', fontsize=30)

data['MarkDown1'] = data['MarkDown1'].replace(np.nan, 0)
data['MarkDown2'] = data['MarkDown2'].replace(np.nan, 0)
data['MarkDown3'] = data['MarkDown3'].replace(np.nan, 0)
data['MarkDown4'] = data['MarkDown4'].replace(np.nan, 0)
data['MarkDown5'] = data['MarkDown5'].replace(np.nan, 0)
data=pd.get_dummies(data,columns=['Type'])
data['Date']= pd.to_datetime(data['Date'])
data['month'] = data['Date'].dt.month
data['Year'] = data['Date'].dt.year
data[['Date','month', 'Year']].head()

data['dayofweek_name'] = data['Date'].dt.day_name()
data[['Date','dayofweek_name']].head()

data['is_weekend'] = np.where(data['dayofweek_name'].isin(['Sunday','Saturday']),1,0)
data[['Date','is_weekend']].head()

data["IsHoliday"] = data["IsHoliday"].astype(int)
del data['dayofweek_name']
# del df['Date']

data.to_csv('merged_data.csv', index=False)

X = df.loc[:, df.columns != 'Weekly_Sales']
```

```
y = df.loc[:, df.columns == 'Weekly_Sales']


X = X[["Store", "Dept", "Size", "IsHoliday", "CPI",
"Temperature","Type_A","Type_B","Type_C","month","Year" ]]

y = y.values.reshape(-1, 1)

print(X.head())


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


from sklearn.ensemble import RandomForestRegressor


rf = RandomForestRegressor(n_estimators=50, max_depth=20, min_samples_split=3,

                           min_samples_leaf=1)

rf.fit(X_train, y_train.ravel())

print('Accuracy:',rf.score(X_test, y_test.ravel())*100,'%')


y_pred = rf.predict(X_test)


from sklearn.metrics import mean_squared_error

from sklearn.metrics import mean_absolute_error


rms = mean_squared_error(y_test, y_pred, squared=False)

print('RMSE:',rms)


print('MAE:',mean_absolute_error(y_test, y_pred))


print('Training Accuracy:',rf.score(X_train, y_train.ravel())*100,'%')
```

```
import xgboost as xgb
import warnings

xg_reg = xgb.XGBRegressor(objective='reg:squarederror', nthread= 4,
                          n_estimators= 500, max_depth= 4, learning_rate= 0.5)
xg_reg.fit(X_train, y_train)

pred=xg_reg.predict(X_train)
y_pred=xg_reg.predict(X_test)

print('Accuracy:',xg_reg.score(X_test, y_test)*100,'%')

rms = mean_squared_error(y_test, y_pred, squared=False)
print('RMSE:',rms)

print('MAE:',mean_absolute_error(y_test, y_pred))

print('Training Accuracy:',xg_reg.score(X_train, y_train)*100,'%')

!pip install pmdarima
import pmdarima
from pmdarima.arima import auto_arima
df =  pd.read_csv("merged_data.csv",keep_default_na=False, na_values=[""])
print(df.columns)

df.Date = pd.to_datetime(df.Date,format='%Y-%m-%d')
```

```
df.index = df.Date
df = df.drop('Date', axis=1)
df = df.resample('MS').mean()
# Resmapling the time series data with month starting first.
# Train-Test splitting of time series data
train_data = df[:int(0.7*(len(df)))]
test_data = df[int(0.7*(len(df))):]

train_data = train_data['Weekly_Sales']
test_data = test_data['Weekly_Sales']

# Plot of Weekly_Sales with respect to years in train and test.
train_data.plot(figsize=(20,8), title= 'Weekly_Sales', fontsize=14)
test_data.plot(figsize=(20,8), title= 'Weekly_Sales', fontsize=14)
plt.show()

model_auto_arima = auto_arima(train_data, trace=True, error_action='ignore',
                              suppress_warnings=True)
model_auto_arima = auto_arima(train_data, trace=True,start_p=0, start_q=0, start_P=0,
                              start_Q=0, max_p=10, max_q=10, max_P=10, max_Q=10,
                              seasonal=True,stepwise=False, suppress_warnings=True,
                              D=1, max_D=10,error_action='ignore',approximation = False)
model_auto_arima.fit(train_data)

# Predicting the test values using predict function.
forecast = model_auto_arima.predict(n_periods=len(test_data))
```

```
forecast = pd.DataFrame(forecast,index = test_data.index,columns=['Prediction'])

plt.figure(figsize=(20,6))

plt.title('Prediction of Weekly Sales using Auto ARIMA model', fontsize=20)


plt.plot(train_data, label='Train')

plt.plot(test_data, label='Test')

plt.plot(forecast, label='Prediction using ARIMA Model')


plt.legend(loc='best')

plt.xlabel('Date', fontsize=14)

plt.ylabel('Weekly Sales', fontsize=14)

plt.show()


# Performance metric for ARIMA model

print('Mean Squared Error (MSE) of ARIMA: ', mean_squared_error(test_data, forecast))

print('Root Mean Squared Error (RMSE) of ARIMA: ', math.sqrt(mean_squared_error(test_data,
forecast)))

print('Mean Absolute Deviation (MAD) of ARIMA: ', mean_absolute_error(test_data, forecast))


from prettytable import PrettyTable


tb = PrettyTable()

tb.field_names = ["Model" ,"Training Accuracy","Testing Accuracy","RMSE","MAE"]

tb.add_row(["Random Forest", 99.05, 96.6, 4202.24, 1665.78])

tb.add_row(["XgBoost", 94.17, 93.96, 5609.57, 3090.79])

tb.add_row(["Arima","-", "-", 685.54, "446.99 (MAD)" ])
```

```
print(tb)


from sklearn.model_selection import cross_val_score


rf = RandomForestRegressor(n_estimators=58, max_depth=27, min_samples_split=3,
                           min_samples_leaf=1)
rf.fit(X_train, y_train.ravel())
y_pred = rf.predict(X_test)


cv2=cross_val_score(rf,X,y.ravel(),cv=5)


np.mean(cv2)
import pickle
pickle.dump(rf, open('rf_model.pkl', 'wb'))
```