WALMART STORE SALES FORECASTING

by

SHRESHTH VATS	18BCE1077
SOHAM DESHPANDE	18BLC1063
MEHUL SANKET	18BLC1075

A project report submitted to

Dr. VETRIVELAN. P

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

CSE3506 – ESSENTIALS OF DATA ANALYTICS

in

B.Tech. COMPUTER SCINCE AND ENGINEERING

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

JUNE 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled "**WALMART STORE SALES FORECASTING"** is a bonafide work of **SHRESHTH VATS – 18BCE1077, SOHAM SARANG DESHPANDE – 18BLC1063 and MEHUL SANKET – 18BLC1075** who carried out the Project work under my supervision and guidance for **CSE3506-Essentials of Data Analytics.**

Dr. VETRIVELAN.P

Associate Professor Senior & HoD - B.Tech (ECE)

School of Electronics Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

Retail companies use Sales Planning by Week to simplify the data capture of volume estimates from regional or store managers, and to get a detailed view of expected sales quantities and revenues by product. When used as part of good business practices in a Financial Planning & Analysis (FP&A) department, a company can improve its sales forecast accuracy which helps to budget for expenses and investments, as well as, reduce the chances that sales revenues become sub-optimized due to poor inventory planning.

Sales forecasting is a common topic in business. One challenge of modelling retail data is the need to make decisions based on limited history. If Christmas comes but once a year, so does the chance to see how strategic decisions impacted the bottom line.

We are provided historical sales data for 45 Walmart stores located in different regions. Each store contains a number of departments, and we are tasked with predicting the department-wide sales for each store.

In addition, Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are the Super Bowl, Labour Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks. We need to model the effects of markdowns on these holiday weeks in the absence of complete/ideal historical data.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Vetrivelan. P,** Associate Professor Senior, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Sivasubramanian. A,** Dean of School of Electronics Engineering, VIT Chennai, for extending the facilities of the School towards our project and for her unstinting support.

We express our thanks to our Head of the Department **Dr. Vetrivelan. P** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

SHRESHTH **SOHAM** **MEHUL**

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Walmart is an American multinational retail corporation that operates a chain of hypermarkets, department stores and grocery stores. As of Jul 2019, Walmart has 11,200 stores in 27 countries with revenues exceeding $500 billion. A challenge facing the retail industry such as Walmart's is to ensure the supply chain and warehouse space usage is optimized to ensure supply meets demand effectively, especially during spikes such as the holiday seasons.

This is where accurate sales forecasting enables companies to make informed business decisions. Companies can base their forecasts on past sales data, industry-wide comparisons and economic trends. However, a forecasting challenge is the need to make decisions based on limited history. If Christmas comes but once a year, so does the chance to see how strategic decisions impacted the bottom line.

1.1 PROBLEM STATEMENT

Historical sales data for 45 Walmart stores located in different regions has been provided. Each store contains many departments, and the sales for each department in each store needs to be projected. Additionally, Walmart runs several promotional markdown events throughout the year. These markdowns precede prominent holidays, the four largest of which are the Super Bowl, Labour Day, Thanksgiving, and Christmas. The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks. These markdowns are known to affect sales, but it is challenging to predict which departments are affected and the extent of the impact.

We need to model the effects of markdowns on these holiday weeks in the absence of complete/ideal historical data.

1.2 OBJECTIVES

This is a **regression analysis** problem. We will analyse the impact of different factors such as holidays, workdays, store location, etc. on the sales. This is because it provides a detailed insight that can be applied further to improve the sales.

The primary objectives of the project are mentioned below.

- analyse the historical sales data for Walmart
- build models to train and test the data
- predict the sales across various departments in each store

- predict the effect of markdowns on the sales during the holiday seasons
- check the accuracy and performance metrics of all models
- finalise a model that best fits the data and gives maximum accuracy

1.3 BENEFITS

Sales forecasting is both a science and an art. Decision makers rely on these forecasts to plan for business expansion and to determine how to fuel the company's growth. So, in many ways, sales forecasting affects everyone in the organization.

The benefits objectives of the project are mentioned below

- A sales forecast helps every business make better business decisions. It helps in overall business planning, budgeting, and risk management.
- Sales forecasting allows companies to efficiently allocate resources for future growth and manage its cash flow.
- Sales forecasts help sales teams achieve their goals by identifying early warning signals in their sales pipeline and course-correct before it's too late
- Sales forecasting also helps businesses to estimate their costs and revenue accurately based on which they are able to predict their short-term and long-term performance.

1.4 DRAWBACKS IN EXISTING SYSTEMS

We read existing papers, mentioned in the 'References' section later and found that none of them tested multiple models to support their accuracy. Hence, the prime detail they lacked was comparison among different models and we tried to solve this in our project.

1.5 CHALLENGES

There were many challenges that we had to face during the course of our project. Some of them were creating a model that would perfectly fit our dataset and also choosing a performance metric that would successfully define the accuracy of our project.

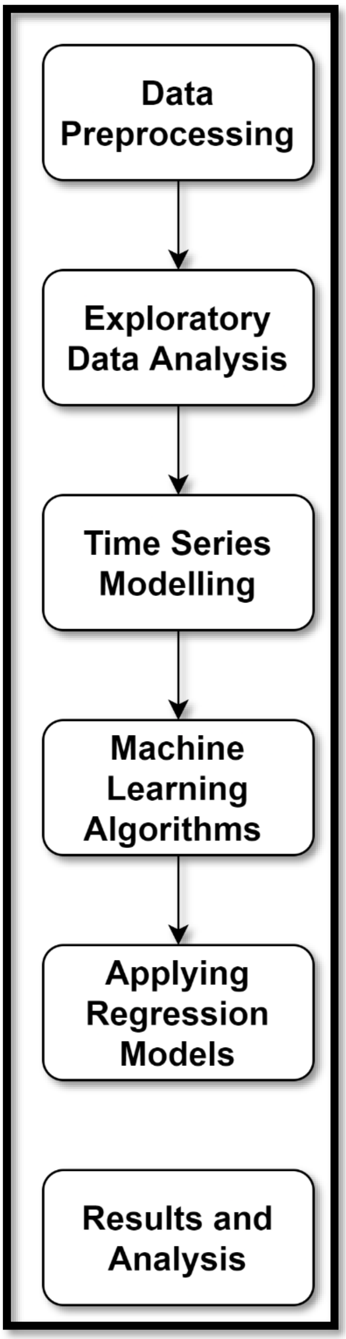
1.5 PROPOSED SOLUTION

We intend to predict the sales of Walmart across its 45 stores in the US and analyses the sales during weekdays, weekends and other important holidays.

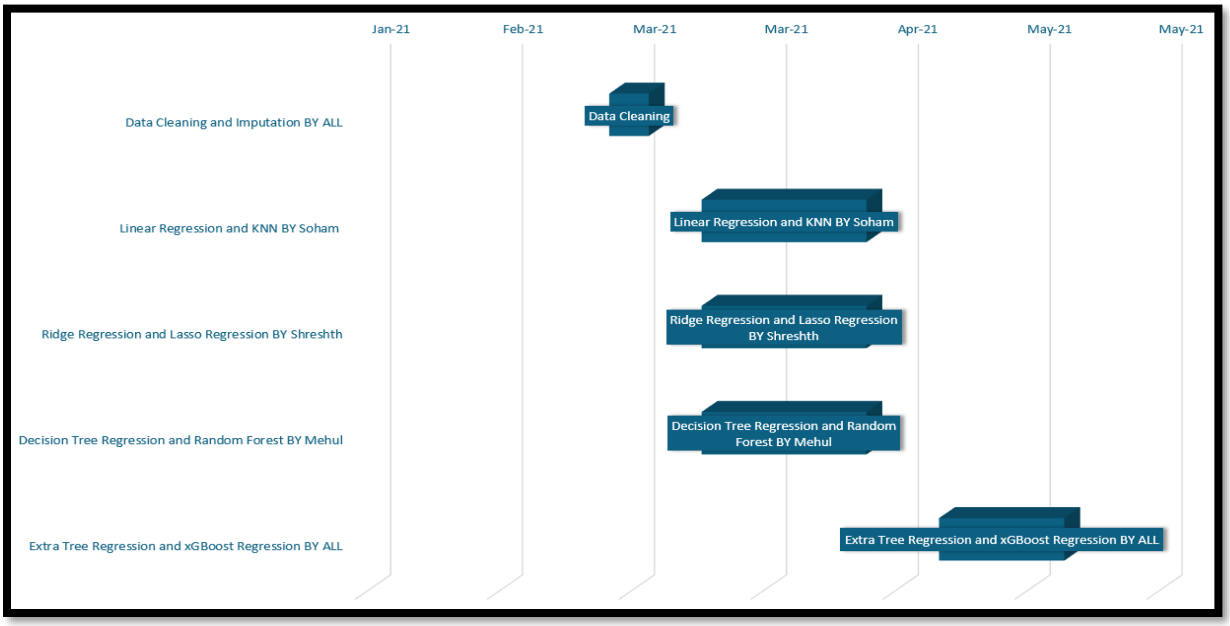
CHAPTER 2

SYSTEM DESIGN

2.1 ARCHITECTURE DIAGRAM



2.2 GANTT CHART



2.3 SOFTWARE SPECIFICATIONS

We use Google Colab because it provides us an easy-to-use platform for Python coding, the language we used for the project. Also, we can easily upload large datasets with ease and share the notebooks with the team members to keep track of the work.

2.4 DATASET DESCRIPTION

Link to data set: https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/data

Data Field contains a total of 4 datasets:

stores.csv **-** This file contains anonymized information about the 45 stores, indicating the type and size of store.

test.csv **-** This file is identical to train.csv, except we have withheld the weekly sales. You must predict the sales for each triplet of store, department, and date in this file.

train.csv **-** This is the historical training data, which covers to 2010–02–05 to 2012–11–01. Within this file you will find the following fields:

Store — the store number

Dept — the department number

Date — the week

Weekly_Sales — sales for the given department in the given store

IsHoliday — whether the week is a special holiday week

features.csv **-** This file contains additional data related to the store, department, and regional activity for the given dates. It contains the following fields:

Store — the store number
Date — the week
Temperature — average temperature in the region
Fuel_Price — cost of fuel in the region
MarkDown1–5 — anonymized data related to promotional markdowns that Walmart is running. MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA.
CPI — the consumer price index
Unemployment — the unemployment rate
IsHoliday — whether the week is a special holiday week

2.5 PERFORMANCE METRICS

Model prediction for this problem can be evaluated in several ways. However, since Kaggle's evaluation is based on weighted mean absolute error (WMAE), same will be leveraged here:

$$\text{WMAE} = \frac{1}{\sum w_i} \sum_{i=1}^{n} w_i |y_i - \hat{y}_i|$$

where

- n is the number of rows
- $\hat{y}_i$ is the predicted sales
- y_i is the actual sales
- w_i are weights. w = 5 if the week is a holiday week, 1 otherwise

This regression metric seems to be a good candidate here as more weightage is being given to predicting the sales on Holiday Weeks (as compared to other weeks) to ensure the spike in demand is predicted appropriately.

CHAPTER 3
SYSTEM IMPLEMENTATION AND ANALYSIS

3.1 SYSTEM IMPLEMENTATION

3.1.1 Algorithms and Techniques

Since the data file contains a date field, the given data set is a time series data set where according to each date the weekly sales with respect to stores and departments have been provided.

Hence, we will apply some of the popular time series forecasting models namely:

1. Auto ARIMA Model
2. Holt-Winters Model

3.2.2 Regression Models

Now we will move to conventional regression algorithms to predict the sales values.

Here we will use below regression models for Weekly Sales Prediction.

1.Linear Regression
2. K Nearest Neighbors Regression
3. Ridge Regression
4. Lasso Regression
5. Decision Tree Regression
6. Random Forest Regression
7. ExtraTrees Regression
8. XGBoost Regression

In every model below steps will be performed.

1) Define the parameters that each library takes.
2) Fit the model on training data.
3) Hyper parameter-tune the parameters using simple for loops.
4) Retrain the model on tuned parameters.
5) Get the Weighted Mean Absolute Error (WMAE) score.

3.2.3 Outputs and Analysis

Importing libraries

```python
# importing all the libraries needed
import numpy as np
import pandas as pd
import scipy.stats as stats
import matplotlib.pyplot as plt
import sklearn
import seaborn as sns
sns.set_style("whitegrid")
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import warnings
warnings.filterwarnings('ignore')
from sklearn.metrics import mean_squared_error, mean_absolute_error

#For date time functions
from datetime import datetime
from datetime import timedelta
import math

# Importing the most popular regression libraries.
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression, ridge_regression, Lasso, SGDRegressor, Ridge
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor
from xgboost import XGBRegressor
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

Loading dataset:

```python
#Loading the data from csv files.
train=pd.read_csv('train.csv')
features=pd.read_csv('features.csv')
stores = pd.read_csv('stores.csv')
```

```python
train.head()
```

	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

```python
features.head()
```

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	IsHoliday
0	1	2010-02-05	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	False
1	1	2010-02-12	38.51	2.548	NaN	NaN	NaN	NaN	NaN	211.242170	8.106	True
2	1	2010-02-19	39.93	2.514	NaN	NaN	NaN	NaN	NaN	211.289143	8.106	False
3	1	2010-02-26	46.63	2.561	NaN	NaN	NaN	NaN	NaN	211.319643	8.106	False
4	1	2010-03-05	46.50	2.625	NaN	NaN	NaN	NaN	NaN	211.350143	8.106	False

```python
stores.head()
```

	Store	Type	Size
0	1	A	151315
1	2	A	202307

Data Pre-processing:

```
# First we check what happens when we replace NaN's with 0.
data.fillna(0).head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Type	Size
0	1	1	2010-02-05	24924.50	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106	A	151315
1	1	2	2010-02-05	50605.27	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106	A	151315
2	1	3	2010-02-05	13740.12	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106	A	151315
3	1	4	2010-02-05	39954.04	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106	A	151315
4	1	5	2010-02-05	32229.38	False	42.31	2.572	0.0	0.0	0.0	0.0	0.0	211.096358	8.106	A	151315

Removing null values from the dataset.

```
data.isnull().head()
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Type	Size
0	False	False	False	False	False	False	False	True	True	True	True	True	False	False	False	False
1	False	False	False	False	False	False	False	True	True	True	True	True	False	False	False	False
2	False	False	False	False	False	False	False	True	True	True	True	True	False	False	False	False
3	False	False	False	False	False	False	False	True	True	True	True	True	False	False	False	False
4	False	False	False	False	False	False	False	True	True	True	True	True	False	False	False	False

```
# Removing rows with null values in all columns
data.dropna(axis=0, how="all", inplace=True)
# Removing all rows with null values in all rows
data.dropna(axis=1, how="all", inplace=True)
```

Exploratory Data Analysis:

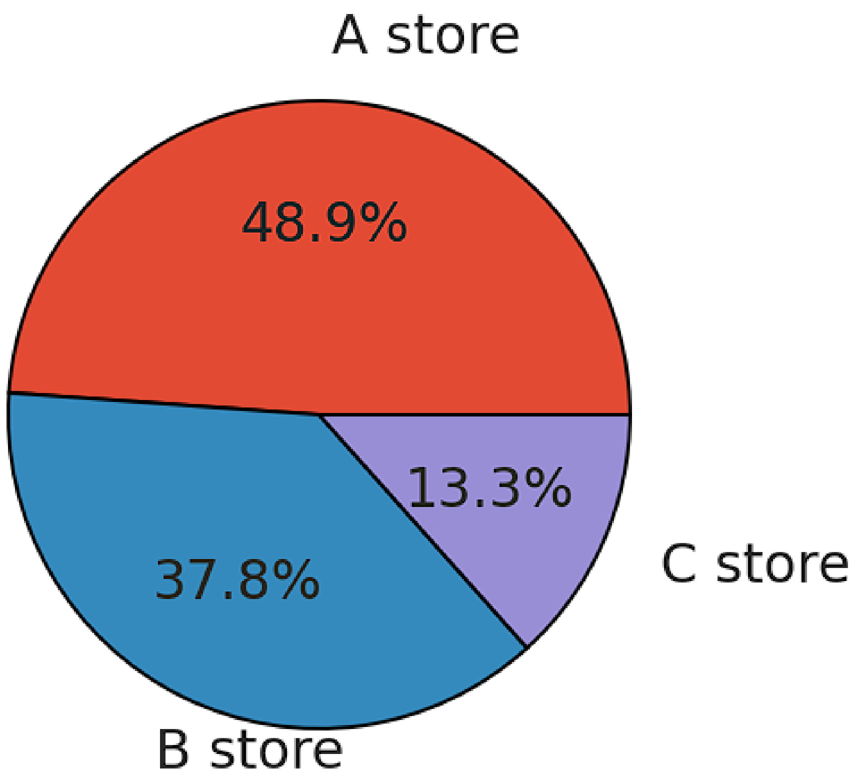
```
print("the shape of stores data set is", stores.shape)
print('='*50)
print("the unique value of store is", stores['Store'].unique())
print('='*110)
print("the unique value of Type is", stores['Type'].unique())
```

```
the shape of stores data set is (45, 3)
==================================================
the unique value of store is [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45]
==============================================================================================================
the unique value of Type is ['A' 'B' 'C']
```

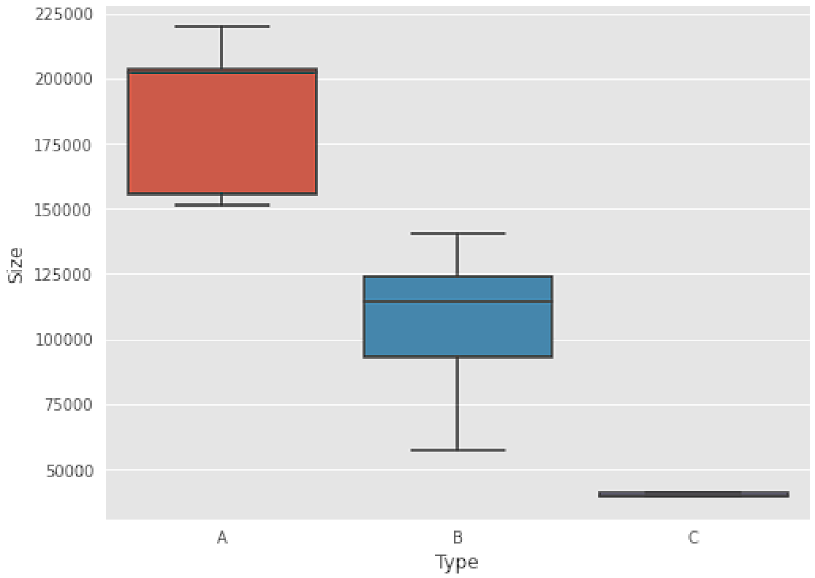
```
sorted_type = stores.groupby('Type')
print(sorted_type.describe()['Size'].round(2))
```

```
      count       mean       std  ...       50%       75%       max
Type                              ...
A      22.0  177247.73  49392.62  ...  202406.0  203819.0  219622.0
B      17.0  101190.71  32371.14  ...  114533.0  123737.0  140167.0
C       6.0   40541.67   1304.15  ...   39910.0   40774.0   42988.0

[3 rows x 8 columns]
```

```
#Make Pie chart for Stores including Weekly Sales.
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=(10,10))

wprops={'edgecolor':'black',
        'linewidth':2}

tprops = {'fontsize':30}

axes.pie(sizes,
         labels=labels,
         explode=(0.0,0,0),
         autopct='%1.1f%%',
         pctdistance=0.6,
         labeldistance=1.2,
         wedgeprops=wprops,
         textprops=tprops,
         radius=0.8,
         center=(0.5,0.5))
plt.show()
```



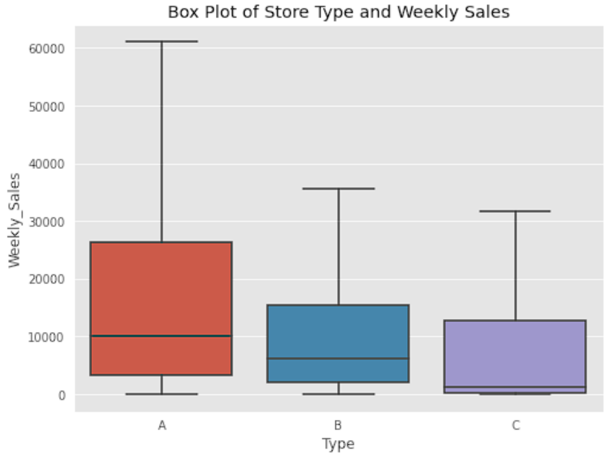
```
#Box Plot of Store Type and Store Size.
type_size = pd.concat([stores['Type'], stores['Size']], axis=1)
plt.figure(figsize=(8,6))
fig = sns.boxplot(x='Type', y='Size', data=type_size, showfliers=False)
```



6. Weekly Sales for Store Type:

```
[ ] # There are negative values present in Weekly sales which are absurd because sales cannot be negative.
    train_stores= train_stores[train_stores['Weekly_Sales'] > 0]
```

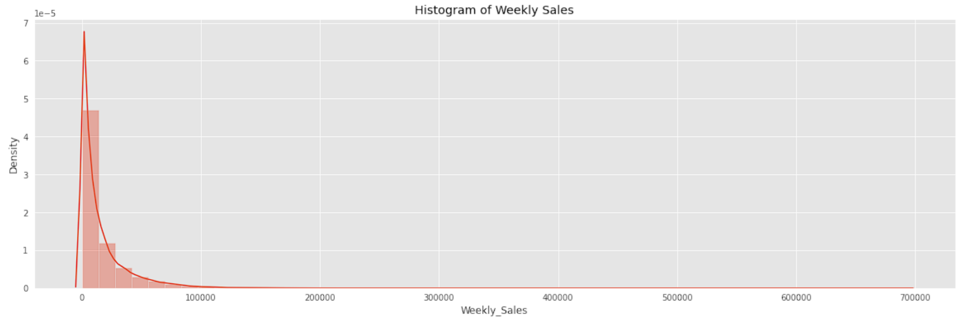
```
[ ] #Plot of Store Type and Weekly Sales
    type_sales = pd.concat([train_stores['Type'], train_stores['Weekly_Sales']], axis=1)
    plt.figure(figsize=(8,6))
    plt.title('Box Plot of Store Type and Weekly Sales')
    fig = sns.boxplot(x='Type', y='Weekly_Sales', data=type_sales, showfliers=False)
```



18. Histogram of Weekly Sales:

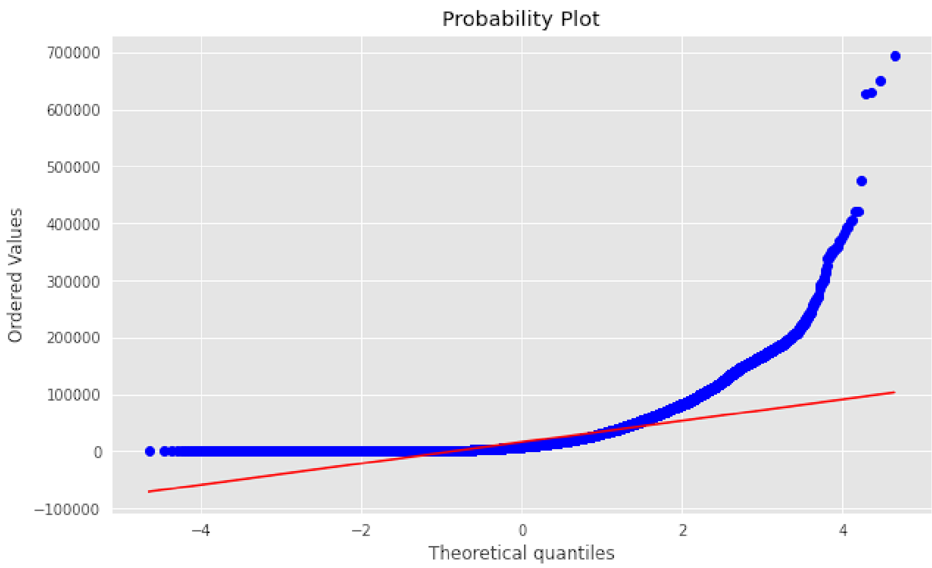
```
[ ]  plt.figure(figsize=(20,6))
     plt.title('Histogram of Weekly Sales')
     fig = sns.distplot(train_stores['Weekly_Sales'].dropna()) #Taking only valid weekly sales values.
     print('Minimum sales:', train_stores['Weekly_Sales'].min())
```

```
Minimum sales: 0.01
```



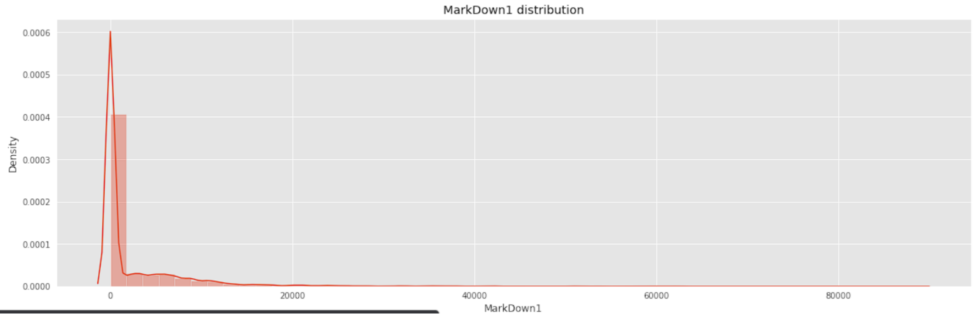
19. Probability Plot of Weekly Sales:

```
[ ]  plt.figure(figsize=(10,6))
     fig = stats.probplot(train_stores['Weekly_Sales'], plot=plt)
```



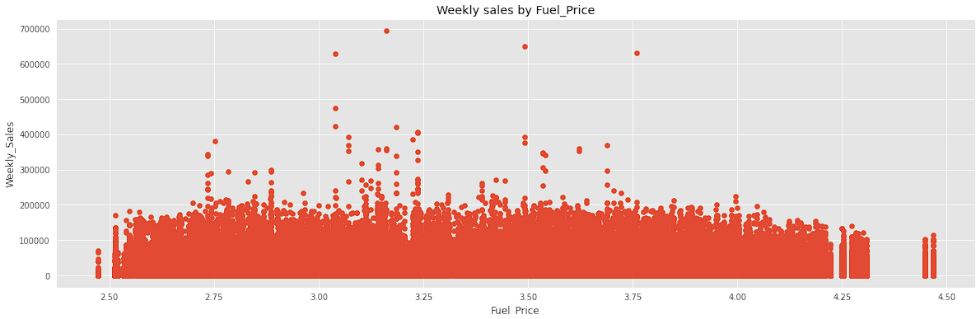
20. Distributions of MarkDown 1, MarkDown 2, MarkDown 3, MarkDown 4, MarkDown 5:

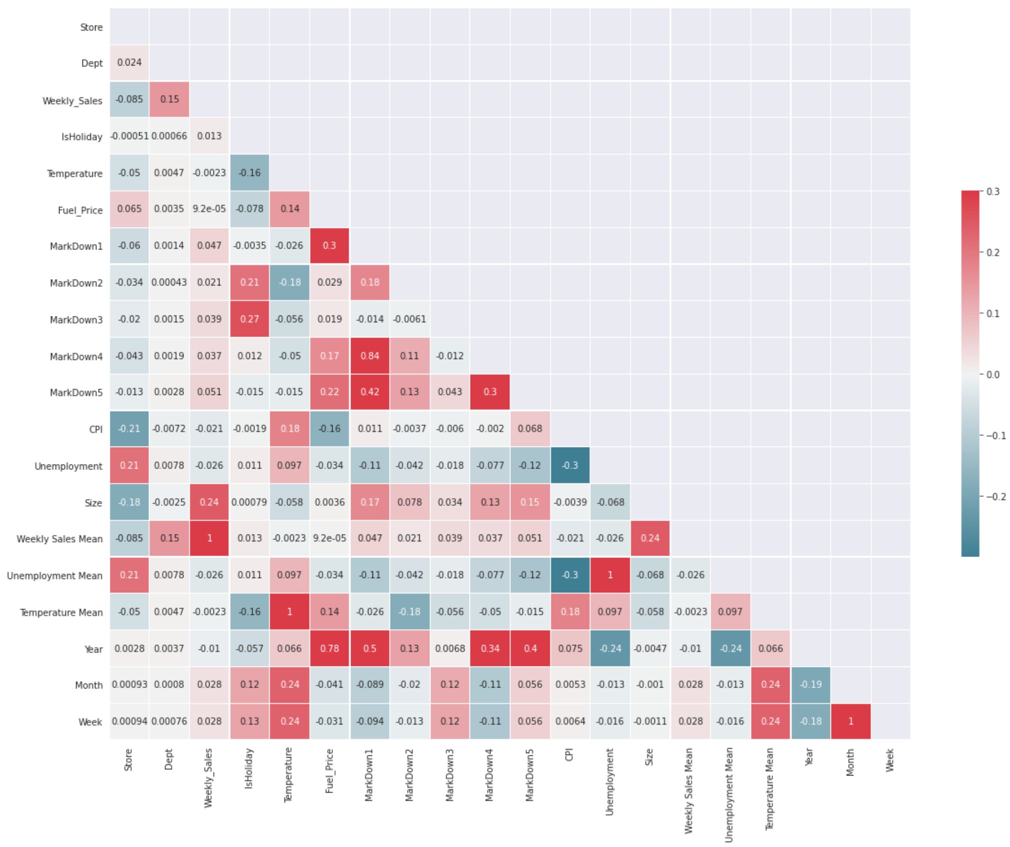
```
[ ]
# Histograms of MarkDowns.
def markdowns(data, column):
  plt.figure(figsize=(20,6))
  sns.distplot(data[column], kde=True)
  plt.title(str(column)+' distribution')
  plt.xlabel(column)

markdowns(data, 'MarkDown1')
markdowns(data, 'MarkDown2')
markdowns(data, 'MarkDown3')
markdowns(data, 'MarkDown4')
markdowns(data, 'MarkDown5')
```



21. Fuel, Temparature and CPI effects:

```
[ ]
scatter(data, 'Fuel_Price') # Fuel
scatter(data, 'Temperature') #Temparature
scatter(data, 'CPI') #CPI
```





Machine Learning Algorithms:

Machine Learning Algorithms

```
[ ]
#Joining the train data with store and features data using inner join.
train = train.merge(features, on=['Store', 'Date'], how='inner').merge(stores, on=['Store'], how='inner')
print(train.shape)
```

```
(421570, 17)
```

```
[ ]
# Make one IsHoliday column instead of two.
train = train.drop(['IsHoliday_y'], axis=1)
train = train.rename(columns={'IsHoliday_x':'IsHoliday'})
print('Train columns:\n', train.columns)
```

```
Train columns:
 Index(['Store', 'Dept', 'Date', 'Weekly_Sales', 'IsHoliday', 'Temperature',
       'Fuel_Price', 'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4',
       'MarkDown5', 'CPI', 'Unemployment', 'Type', 'Size'],
      dtype='object')
```

```
[ ]
# Converting Date to datetime
train['Date'] = pd.to_datetime(train['Date'])

# Extract date features
train['Date_dayofweek'] =train['Date'].dt.dayofweek
train['Date_month'] =train['Date'].dt.month
train['Date_year'] =train['Date'].dt.year
train['Date_day'] =train['Date'].dt.day
```

Remove Unnecessary Columns

```
# Not so important features.
features_drop=['Unemployment','CPI','MarkDown5']
train=train.drop(features_drop, axis=1)

print('Final train shape:', train.shape)
train.head(2)
```

```
Final train shape: (420285, 17)
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	Type	Size
0	1	1	2010-02-05	24924.50	0	42.31	2.572	0.0	0.0	0.0	0.0	1	151315
1	1	2	2010-02-05	50605.27	0	42.31	2.572	0.0	0.0	0.0	0.0	1	151315

Train-Test Splitting

```
train = train.sort_values(by='Date', ascending=True) # Sorting the data in increasing order of Date and then splitting.
y = train['Weekly_Sales']
X = train.drop(['Weekly_Sales'], axis=1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3) # Train:Test = 70:30 splitting.
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.3) #Train:CV = 70:30 splitting.
```

```
# Remove Date column as it does not allow the models to fit on the data.
X_train = X_train.drop(['Date'], axis=1)
X_cv = X_cv.drop(['Date'], axis=1)
X_test = X_test.drop(['Date'], axis=1)
```

```
# Final shapes.
print('Train:', X_train.shape, y_train.shape)
print('CV:', X_cv.shape, y_cv.shape)
print('Test', X_test.shape, y_test.shape)
```

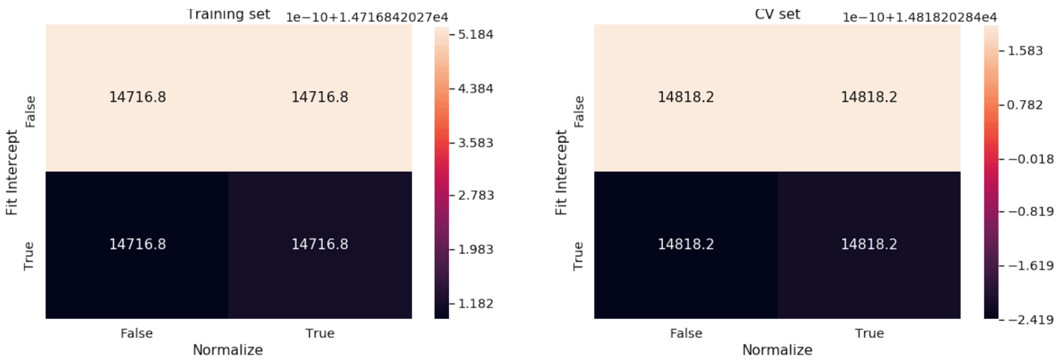
Regression Models:

Linear Regression:

```
# Define list of empty train error and cv error.
error_cv_lr = []
error_train_lr = []
fit_intercept = [True,False]
normalize = [True,False]
lr_hyperparams = []

"""Calculating train and CV errors for Fit Intercept and Normalize parameters."""

for i in fit_intercept:
    for j in normalize:
        lr = LinearRegression(fit_intercept=i, normalize=j) # Apply Linear Regression.
        lr.fit(X_train, y_train) # Fit the model.
        y_pred_cv_lr = lr.predict(X_cv) # Predict CV data.
        y_pred_train_lr = lr.predict(X_train) # Predict Train data.
        error_cv_lr.append(wmae_cv(y_cv, y_pred_cv_lr)) # Get CV error.
        error_train_lr.append(wmae_train(y_train, y_pred_train_lr)) # Get Train error.
        lr_hyperparams.append({'Fit Intercept':i, 'Normalize':j}) # Hyperparameters.
```

```python
"""Making dataframe containing values of hyper parameters with train and cv

lr_dataframe = pd.DataFrame(lr_hyperparams)
lr_dataframe['Train Error'] = error_train_lr
lr_dataframe['CV Error'] = error_cv_lr
lr_dataframe.sort_values(by=['CV Error'], ascending=True)
lr_dataframe
```

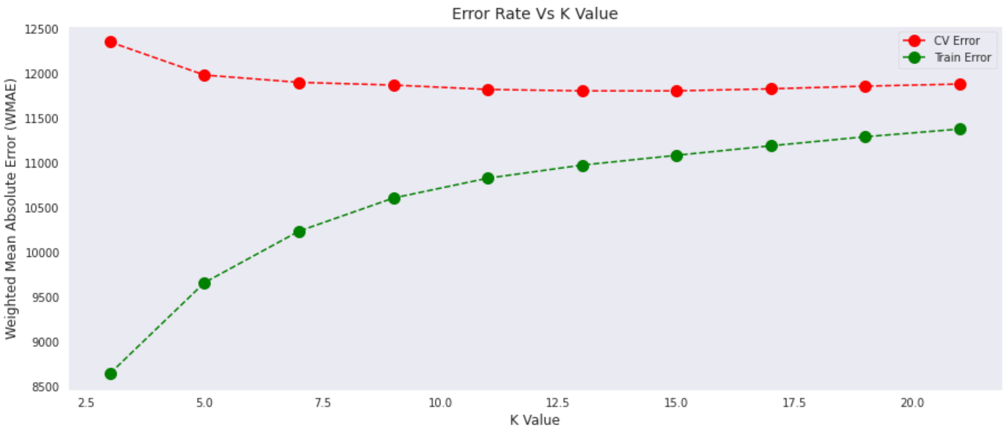
	Fit Intercept	Normalize	Train Error	CV Error
0	True	True	14716.842027	14818.20284
1	True	False	14716.842027	14818.20284
2	False	True	14716.842027	14818.20284
3	False	False	14716.842027	14818.20284



KNN Regression

```python
# Define empty train and CV error lists and list of hyper parameter values.
error_cv = []
error_train = []

for i in range(3,22,2): # Loop over number of nearest neighbors (K) values [3,5,7,9,11,13,15,17,19,21].
    knn = KNeighborsRegressor(n_neighbors=i) # Apply KNN Regressor.
    knn.fit(X_train, y_train) # Fit the model.
    y_pred_cv= knn.predict(X_cv) # Predict CV data.
    y_pred_train= knn.predict(X_train) # Predict Train data.
    error_cv.append(wmae_cv(y_cv, y_pred_cv)) # Get CV error.
    error_train.append(wmae_train(y_train, y_pred_train)) # Get Train error.
```

```python
"""Plot Train error and CV error against number of nearest neighbors (K) values to select best hyper parameter."""

plt.figure(figsize=(15, 6))
plt.plot(range(3,22,2), error_cv, color='red', linestyle='dashed', marker='o', markerfacecolor='red', markersize=10, label='CV Error')
plt.plot(range(3,22,2), error_train, color='green', linestyle='dashed', marker='o',markerfacecolor='green', markersize=10, label='Train Error')
plt.legend(loc='best')
plt.title('Error Rate Vs K Value', fontsize=14)
plt.xlabel('K Value',fontsize=12)
plt.ylabel('Weighted Mean Absolute Error (WMAE)', fontsize=12)
plt.show()
```



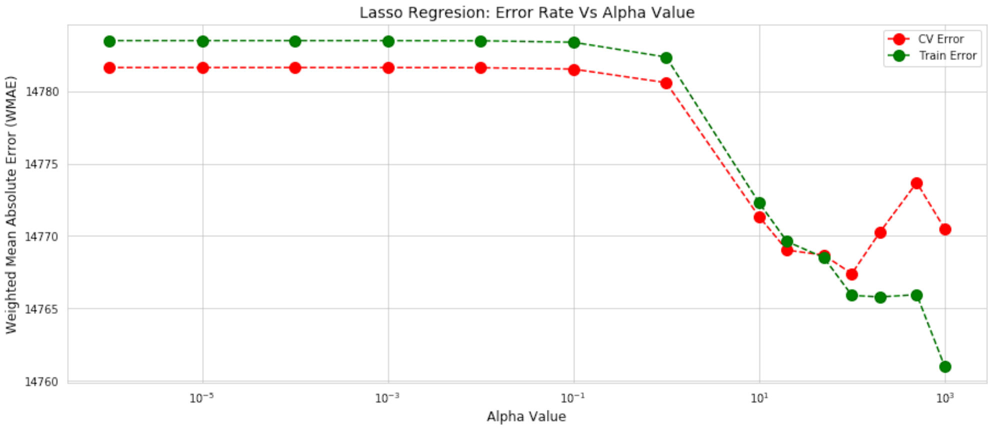
Ridge Regression

```
[ ]
# Define the empty train and CV error lists and list of hyper parameter values.
error_cv_ridge = []
error_train_ridge = []
alpha = [0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10,20,50,100,200,500,1000]

"""Calculating train and cv errors for alpha values."""

for i in alpha:# Loop over alpha.
  ridge = Ridge(alpha=i) # Apply Ridge Regressor.
  ridge.fit(X_train, y_train) # Fit the mdoel.
  y_pred_cv_ridge = ridge.predict(X_cv) # Predict CV data.
  y_pred_train_ridge = ridge.predict(X_train) # Predict Train data.
  error_cv_ridge.append(wmae_cv(y_cv, y_pred_cv_ridge)) # Calculate CV error.
  error_train_ridge.append(wmae_train(y_train, y_pred_train_ridge)) # Calculate Train error.
```

```
[ ]
"""Plot Train error and CV error against alpha values to select best hyper parameter."""

plt.figure(figsize=(15, 6))
plt.plot(alpha, error_cv_ridge, color='red', linestyle='dashed', marker='o', markerfacecolor='red', markersize=10, label='CV Error')
plt.plot(alpha, error_train_ridge, color='green', linestyle='dashed', marker='o',markerfacecolor='green', markersize=10, label='Train Error')
plt.legend(loc='best')
plt.title('Ridge Regresion: Error Rate Vs Alpha Value', fontsize=14)
plt.xlabel('Alpha Value',fontsize=12)
plt.ylabel('Weighted Mean Absolute Error (WMAE)', fontsize=12)
plt.show()
```

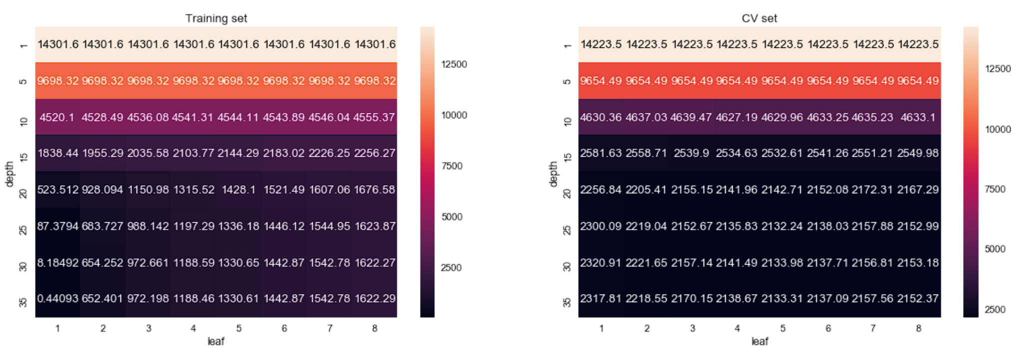
Lasso Regression:

```python
# Define empty train and cv error list and list of hyper parameter values.
error_cv_lasso = []
error_train_lasso = []
alpha = [0.000001,0.00001,0.0001,0.001,0.01,0.1,1,10,20,50,100,200,500,1000]

"""Calculating train and cv errors for alpha values."""

for i in alpha: # Loop over alpha
    lasso = Lasso(alpha=i) # Apply Lasso Regresor.
    lasso.fit(X_train, y_train) # Fit the model.
    y_pred_cv_lasso = lasso.predict(X_cv) # Predict CV data.
    y_pred_train_lasso = lasso.predict(X_train) # Predict Train data.
    error_cv_lasso.append(wmae_cv(y_cv, y_pred_cv_lasso)) # Get CV error.
    error_train_lasso.append(wmae_train(y_train, y_pred_train_lasso)) # Get Train error.
```



Decision Tree Regression:

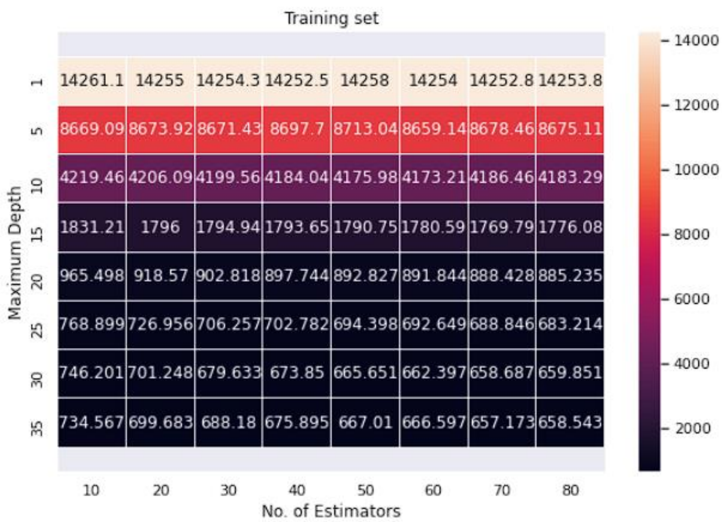
```python
# Define the list of errors and list of hyper parameters.
error_cv_dt = []
error_train_dt = []
max_depth = [1,5,10,15,20,25,30,35]
min_samples_leaf = [1,2,3,4,5,6,7,8]
dt_hyperparams = []

"""Calculating train and CV errors for maximum depth and minimum samples leaf parameters."""

for i in max_depth: # Loop over max_depth.
    for j in min_samples_leaf: # Loop over min_samples_leaf.
        dt = DecisionTreeRegressor(max_depth=i, min_samples_leaf=j) # Apply Decision Tree Regressor.
        dt.fit(X_train, y_train) # Fit the model.
        y_pred_cv_dt = dt.predict(X_cv) # Predict CV data.
        y_pred_train_dt = dt.predict(X_train) # Predict Train data.
        error_cv_dt.append(wmae_cv(y_cv, y_pred_cv_dt)) # Calculate CV error.
        error_train_dt.append(wmae_train(y_train, y_pred_train_dt)) # Calculate Train error.
        dt_hyperparams.append({'depth':i, 'leaf':j}) # Get the list of hyper parameters.
```

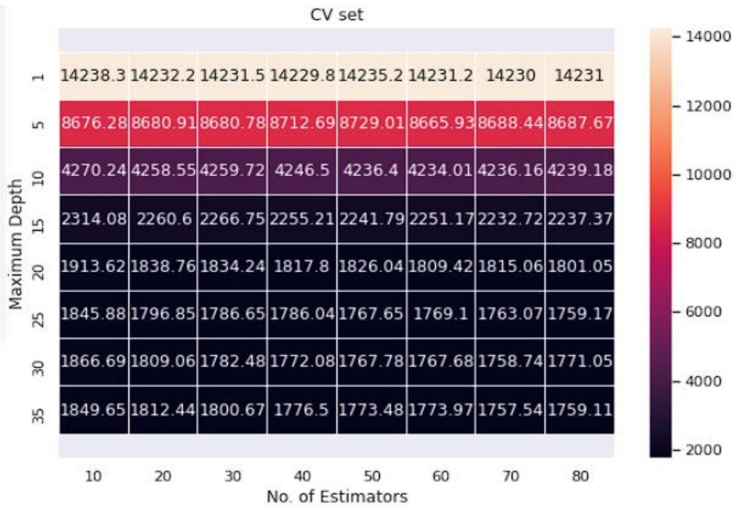


Random Forest Regression

```python
# Define the list of errors and list of hyper parameters.
error_cv_rf = []
error_train_rf = []
max_depth = [1,5,10,15,20,25,30,35]
n_estimators = [10,20,30,40,50,60,70,80]
rf_hyperparams = []

"""Calculating train and CV errors for maximum depth and number of estimators parameters."""

for i in max_depth: # Loop over max_depth.
    for j in n_estimators: # Loop over n_estimators.
        rf = RandomForestRegressor(max_depth=i, n_estimators=j) # Apply Random Forest Regressor.
        rf.fit(X_train, y_train) # Fit the model.
        y_pred_cv_rf = rf.predict(X_cv) # Predict CV data.
        y_pred_train_rf = rf.predict(X_train) # Predict Train data.
        error_cv_rf.append(wmae_cv(y_cv, y_pred_cv_rf)) # Get CV error.
        error_train_rf.append(wmae_train(y_train, y_pred_train_rf)) # Get Train error.
        rf_hyperparams.append({'Maximum Depth':i, 'No. of Estimators':j}) # Get list of hyper parameter values.
```

```python
"""Creating heatmaps for Train loss and CV loss."""

sns.set(font_scale=1.0)
train_rf = pd.pivot_table(rf_dataframe,'Train Error','Maximum Depth','No. of Estimators') # Pivot table of Train data.
cv_rf = pd.pivot_table(rf_dataframe, 'CV Error','Maximum Depth','No. of Estimators') # Pivot table of CV data.
fig, ax = plt.subplots(1,2, figsize=(20,6))
ax_train = sns.heatmap(train_rf, annot=True, fmt='2g', ax=ax[0], linewidths=0.01)
ax_cv = sns.heatmap(cv_rf, annot=True, fmt='4g', ax=ax[1], linewidths=0.01)

bottom_train, top_train = ax_train.get_ylim()
ax_train.set_ylim(bottom_train + 0.5, top_train - 0.5)

bottom_cv, top_cv = ax_cv.get_ylim()
ax_cv.set_ylim(bottom_cv + 0.5, top_cv - 0.5)

ax[0].set_title('Training set')
ax[1].set_title('CV set')
plt.show()
```





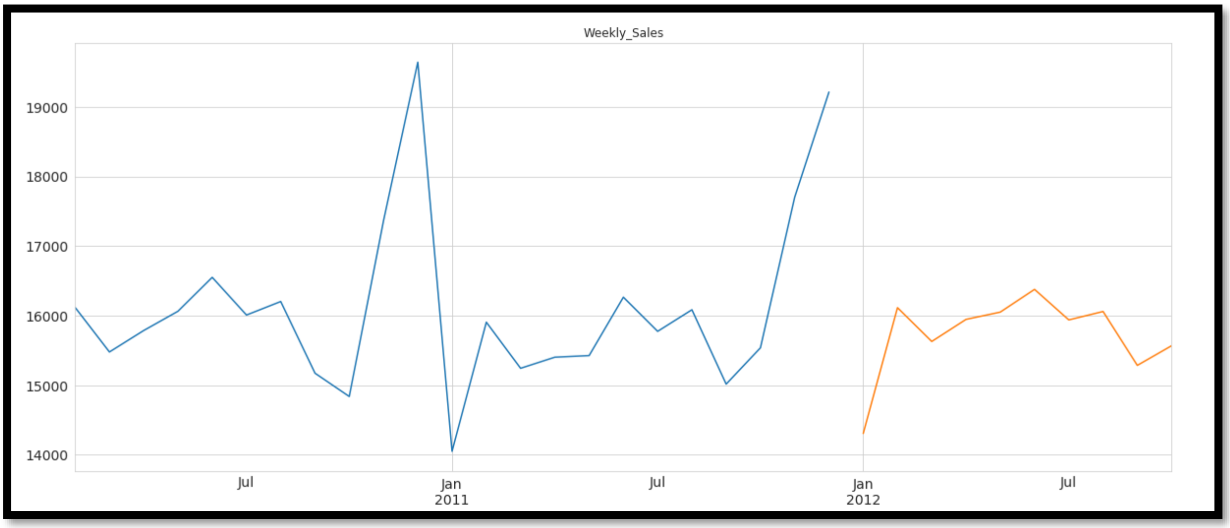
3.2 RESULTS AND INFERENCES

We applied 8 regression models and we got the WMAE scores as given below:

	Model Name	WMAE Score
1	Linear Regression	14904.66
2	KNN Regression	11887.99
3	Ridge Regression	14824.52
4	Lasso Regression	14810.89
5	Decision Tree Regression	2134.17
6	Random Forest Regression	1785.20
7	ExtraTrees Regression	1986.29
8	XGBoost Regession	2765.22

The lowest Weighted mean absolute error (WMAE) score is observed for Random Forest Regression i.e., 1785.20.
Hence, the best model out of all 8 for the correct time series forecasting of the Walmart Sales Data is Random Forest due to a better WMAE score even though its execution time was quite high.

ARIMA plot can also be given as:


Weekly_Sales
19000
18000
17000
16000
15000
14000
Jul
Jan
2011
Jul
Jan
2012
Jul

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 CONCLUSION

In this project we have performed Walmart store sales forecasting for 45 different stores using 8 different regression techniques. The task involved predicting the sales on any given day at any store. In order to familiarize ourselves with the task we have studied previous work in the domain including Time Series Algorithm. A lot of analysis was performed on the data to identify patterns and outliers which would boost or impede the prediction algorithm.

The features used ranged from store information to department information as well as socio-geographical information. Machine learning methods like Linear Regression, Random Forest Regression and more were implemented and the results compared. Random Forest was observed to perform the best at prediction.

Hence, we can conclude that taking averages of top n models helps in reducing loss. As here available data is less, so loss difference is not extraordinary. But in large datasets of sizes in Gigabytes and Terabytes, this trick of simple averaging may reduce the loss to a great extent.

With efficiency being the way forward in most industries today, we aim to expand our solution to help stores improve productivity and increase revenue by taking advantage of Data Analysis.

4.2 FUTURE WORK

We could move froward with our analysis that the faster processing of Random Forest model would thus help us in better results along with its increased accuracy.

- Modifying date feature into days, month, weeks.
- The dataset includes special occasions i.e. Christmas, pre-Christmas, black Friday, Labour Day, etc. On these days people tend to shop more than usual days. So, adding these as a feature to data will also improve accuracy to a great extent.
- Also, there are a missing value gap between training data and test data with 2 features i.e., CPI and Unemployment. If that gap is reduced then also performance can be improved.

APPENDIX

SOURCE CODES

https://colab.research.google.com/drive/1pCPxKhqKBhTJrggmMhCznIXcS-smJ5cz?usp=sharing

VIDEO LINK

https://drive.google.com/file/d/1y2q9yPFyzpDya-EcyP9XulDD1E_mId5u/view

PREDICTED OUTPUT

https://docs.google.com/spreadsheets/d/1a4atRHVfnhNQwDAoydWB2jYeJ5wO7Qm499yY3ynxJ2g/edit?usp=sharing

	A	B
1	Id	Weekly_Sales
2	1_1_2012-11-02	32992.28856
3	1_1_2012-11-09	27440.72242
4	1_1_2012-11-16	19463.83795
5	1_1_2012-11-23	16473.65406
6	1_1_2012-11-30	22482.16718
7	1_1_2012-12-07	34064.67382
8	1_1_2012-12-14	43008.74
9	1_1_2012-12-21	43026.89952
10	1_1_2012-12-28	34294.63308
11	1_1_2013-01-04	22404.18174
12	1_1_2013-01-11	13839.89038
13	1_1_2013-01-18	12294.12718
14	1_1_2013-01-25	17747.6194
15	1_1_2013-02-01	27197.86353
16	1_1_2013-02-08	35778.41914
17	1_1_2013-02-15	38630.91315
18	1_1_2013-02-22	33830.94503
19	1_1_2013-03-01	24496.19486
20	1_1_2013-03-08	17326.42914
21	1_1_2013-03-15	17645.38246
22	1_1_2013-03-22	24994.51279
23	1_1_2013-03-29	33574.51928
24	1_1_2013-04-05	37302.50015
25	1_1_2013-04-12	34633.20507

CHAPTER 5

REFERENCES

[1] Beheshti-Kashi, S., Karimi, H.R., Thoben, K.D., Lütjen, M. and Teucke, M., 2015. A survey on retail sales forecasting and prediction in fashion markets. *Systems Science & Control Engineering*, *3*(1), pp.154-161.

[2] Pavlyshenko, B.M., 2019. Machine-learning models for sales time series forecasting. *Data*, *4*(1), p.15.

[3] Loureiro, A.L., Miguéis, V.L. and da Silva, L.F., 2018. Exploring the use of deep neural networks for sales forecasting in fashion retail. *Decision Support Systems*, *114*, pp.81-93.

[4] Thiesing, F.M. and Vornberger, O., 1997, June. Sales forecasting using neural networks. In *Proceedings of International Conference on Neural Networks (ICNN'97)* (Vol. 4, pp. 2125-2128). IEEE.

[5] Catal, C., Kaan, E.C.E., Arslan, B. and Akbulut, A., 2019. Benchmarking of regression algorithms and time series analysis techniques for sales forecasting. *Balkan Journal of Electrical and Computer Engineering*, *7*(1), pp.20-26.

[6] Lasek, A., Cercone, N. and Saunders, J., 2016. Restaurant sales and customer demand forecasting: Literature survey and categorization of methods. *Smart City 360°*, pp.479-491.

BIODATA

Name: Shreshth Vats

Mobile Number: 9384685285

E-mail: shreshth.vats2018@vitstudent.ac.in

Permanent Address: Apex Athena, Sector 75, Noida

Name: Soham Deshpande

Mobile Number: 9384695477

E-mail: soham.deshpande2000@gmail.com

Permanent Address: Daisy appt lalit estate baner pune

Name: Mehul Sanket

Mobile Number: 9973921477

E-mail: mehul.sanket2018@vitstudent.ac.in

Permanent Address: Brindavan, Jagdeopath, Patna-800014, Bihar