Walmart Sale Forecasting

Submitted By

Sayali S. Khedkar

About

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.

- On **July 2, 1962**, Sam Walton opens the first Walmart store in Rogers, Arkansas.
- The Walton family owns 24 stores, ringing up $12.7 million in sales.
- The company officially incorporates as Wal-Mart Stores, Inc.

Content

Chapter 1

[A] Introduction

Walmart Inc. is an American multinational retail corporation that operates a chain of hypermarkets (also called (supercenters), discount department stores, and grocery stores in the United States, headquartered in Bentonville, Arkansas. The company was founded by Sam Walton and James "Bud" Walton in nearby Rogers, Arkansas in 1962 and incorporated under Delaware General Corporation Law on October 31, 1969. It also owns and operates Sam's Club retail warehouses.

- Sales forecasting or predicting the future is very important for every business. It is used for companies to making plans for high revenue, keep costs lower and high efficiency.
- Companies made short-term and long -term future planning as per forecasting data. Based on past data with some assumption which predict future trends and draw their budget accordingly.
- There are many factors like Market changes, Product changes, Economic conditions, season changes, etc; which impact to forecast of sales. Companies can make a plane to meet future demands and make improvements in their sales by keeping in mind these various factors.
- The data collected ranges from 2010 to 2012, where 45 Walmart stores across the country were included in this analysis..
- These data sets contained information about the stores, departments, temperature, unemployment, CPI, isHoliday, and MarkDowns.

[B] Scope & Trend

Walmart is the world's largest company by revenue (approximately four hundred and eighty billion dollars) and the largest private employer in the world with two point three million employees. Walmart is also one of the world's most valuable companies by market value, and is also the largest grocery retailer in the U.S. “One Nation Under Walmart” is a case about how Walmart has taken over the retail business and the effects of their market domination. The case also shows statistics of how much percentage Walmart is of many suppliers’ sales. According to the case Walmart has a 30% market share of all household items. Twenty-eight percent of Dial’s business and twenty-four percent of Del Monte’s business go through Walmart stores. It is also worth noting that Walmart imports ten percent of all United States imports from China.

The case states that Walmart is able to offer cheaper prices because they put so much pressure on their suppliers to lower their prices. The case, “One Nation Under Walmart”, explains the problems that some people have with the massive retailer. One of these problems is how Walmart has forced numerous local businesses to close their doors through their extremely competitive pricing. They are able to purchase bulk goods at such low prices and thus pass the savings onto customers. As a result of these lew costs, rivals are driven out of business which results in a loss of jobs. .

Revenue

Walmart annual/quarterly revenue history and growth rate from 2010 to 2022. Revenue can be defined as the amount of money a company receives from its customers in exchange for the sales of goods or services. Revenue is the top line item on an income statement from which all costs and expenses are subtracted to arrive at net income.

- Walmart revenue for the quarter ending October 31, 2022 was **$152.813B**, a **8.74% increase** year-over-year.
- Walmart revenue for the twelve months ending October 31, 2022 was **$600.112B**, a **4.92% increase** year-over-year.
- Walmart annual revenue for 2022 was **$572.754B**, a **2.43% increase** from 2021.
- Walmart annual revenue for 2021 was **$559.151B**, a **6.72% increase** from 2020.
- Walmart annual revenue for 2020 was **$523.964B**, a **1.86% increase** from 2019.

[C] Literature of Existing System

Walmart is multinational retail hypermarket in USA. It delivers quality merchandise for the lowest prices possible. Besides building a reputation, it always maintains a high position in terms of strong logistics, supply chain management, and efficiency.

Walmart is well known for its slogan – "Always Low Prices," making it one of the key customer-centric approaches to be a market leader consistently.

Historically, Walmart is known for its two main business keys:

1.One-stop shopping

2. Everyday Low Cost (EDLC) or Everyday Low Price (**EDLP**)

Advantages:

- Walmart offers the lowest prices to attract customers and focuses more in making bulk sales rather than a few. It doesn't believe in overpricing the products.
- Diversification makes sure that their complimentary products drive sales of a product even if another is under-promoted.
- Outstanding procurement strategies allow Walmart to negotiate with even affordable suppliers in order to keep their prices low.

Disadvantages:

- Strict Price Requirements Walmart is committed to offering consumers the lowest price possible for products on its marketplace. While brands and sellers choose the price for their products, it must not interfere with Walmart's Price Parity Rule and Price Leadership Rule.

➢ Lower Profit Margins Brands and sellers may need to reduce their product prices on Walmart's marketplace compared to Amazon.com or their own website due to Walmart's EDLP guarantee.

[A] Objective

- Predict the sales for each department using historical markdown data from the Walmart dataset containing data of 45 Walmart stores.
- The purpose of this project is to develop a predictive model and find out the sales of each product at a given Walmart store.
- Predict which departments are affected with the holiday markdown events and the extent of impact.
- Perform dimensionality reduction to improve prediction error by shrinkage in order to reduce overfitting.

[B] Abstract

- These data sets contained information about the stores, departments, temperature, unemployment, CPI, isHoliday, and MarkDowns.
- 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, Labor Day, Thanksgiving, and Christmas.
- The weeks including these holidays are weighted five times higher in the evaluation than non-holiday weeks. Part of the challenge presented by this competition is modeling the effects of markdowns on these holiday weeks in the absence of complete/ideal historical data.
- This file contains anonymized information about the 45 stores, indicating the type and size of store.
- Sales forecasting or predicting the future is very important for every business. It is used for companies to making plans for high revenue, keep costs lower and high efficiency. Companies made short-term and long-term future planning as per forecasting data.
- Based on past data with some assumption which predict future trends and draw their budget accordingly.

[C] Methodology

Objective and Methodology of the Project:

- Python Programming
- Python Libraries
- Importing Libraries
- Machine Learning Implementation
- Machine Learning Algorithms

Python Programming

- Machine learning and AI, as a unit, are still developing but are rapidly growing in usage due to the need for automation.
- Artificial Intelligence makes it possible to create innovative solutions to common problems, such as Business Improving Strategy, Pricing, etc.
- The demand for smart solutions to real-world problems necessitates the need to develop AI further in order to automate tasks that are tedious to program without AI.
- Python programming language is considered the best algorithm to help automate such tasks, and it offers greater simplicity and consistency than other programming languages.
- Further, the presence of an engaging python community makes it easy for developers to discuss projects and contribute ideas on how to enhance their code.
- Python includes a modular machine learning library known as Sci-Kit Learn, which provides easy-to-use algorithms for use in machine learning tasks. The best and most reliable coding solutions require a proper structure and tested environment, which is available in the Python frameworks and libraries.

Python Libraries

- Numpy = Use for high computation numerical operation. It consists of inbuilt mathematical functions for easy computations.

- Pandas = It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.

- matplotlib = This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc.

- Seaborn = Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

- Scikit Learn = It supports variously supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

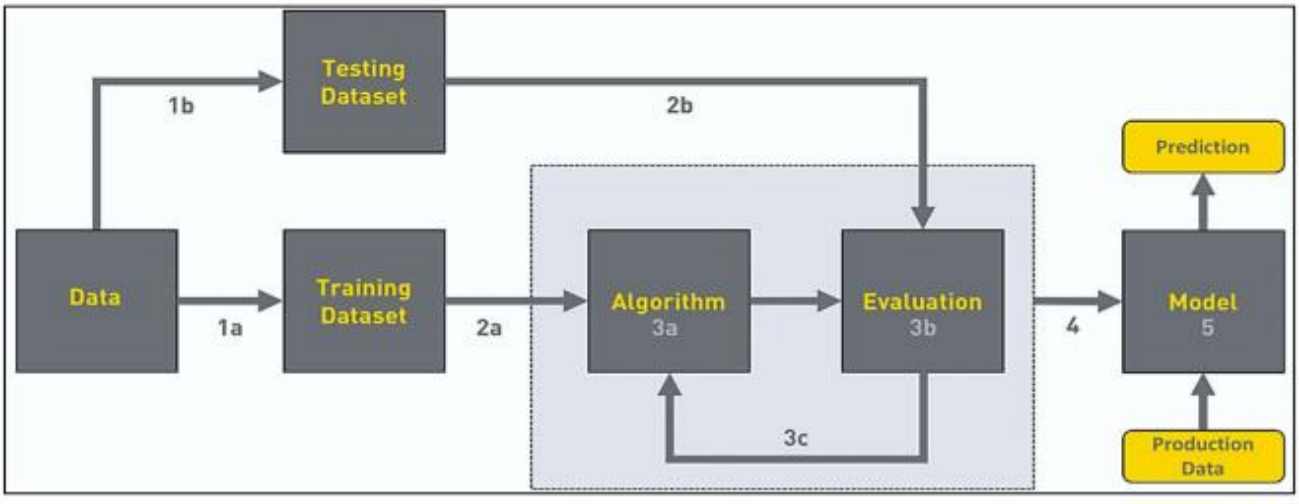
- Warnings.filterwarnings = The warnings filter is initialized by (-w) options passed to the Python interpreter command line and the python warnings environment variable. The interpreter saves the arguments for all supplied entries without interpretation system warning option; the warnings module parses these when it is first imported (invalid options are ignored, after printing a message to system error).

Machine Learning Implementation

- The Algorithms we used are Linear Regression, Decision Tree Regressor, Random Forest Regressor and XGBoost Regressor by using Kaggle repository dataset. For implementation of Python programming Anaconda (Jupyter notebook) is best tool, which have many type of library that make the work more accurate and precise.

Use of Regressor in the Machine Learning Models:

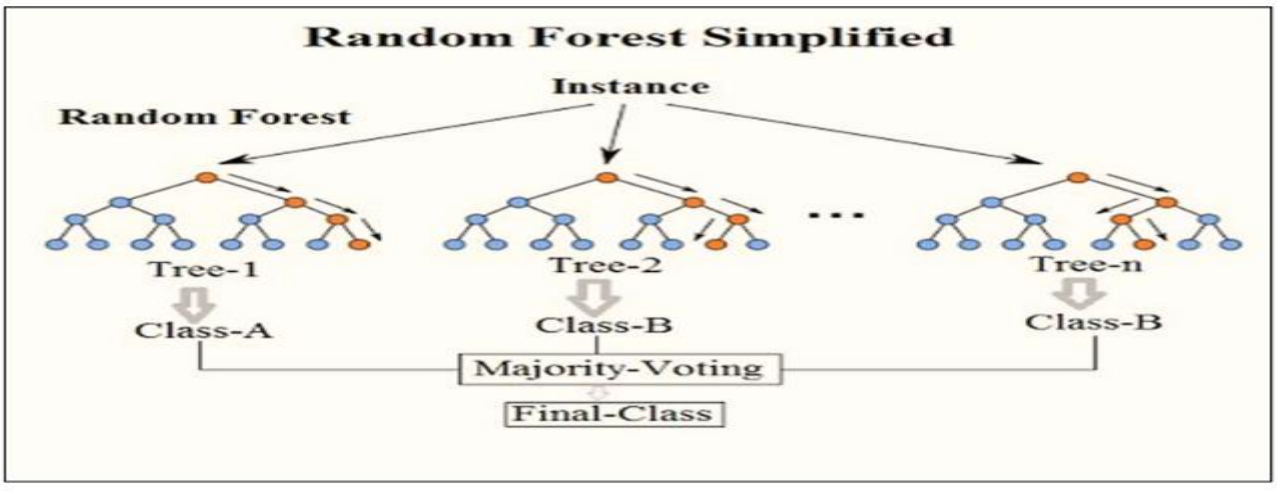
- Regression is a supervised machine learning technique which is used to predict Continuous values. The ultimate goal of the regression algorithm is to plot a best-fit line or a curve between the data.
- Machine Learning Regression is a technique for investigating the relationship between independent variables or features and a dependent variable or outcome. It's used as a method for predictive modelling in machine learning, in which an algorithm is used to predict continuous outcomes.
- There is no categorical Values in the dataset so, will not use the Classifiers in the Machine Learning Models.
- That's why we use Regressor Model in this Dataset.



Machine Learning Algorithms

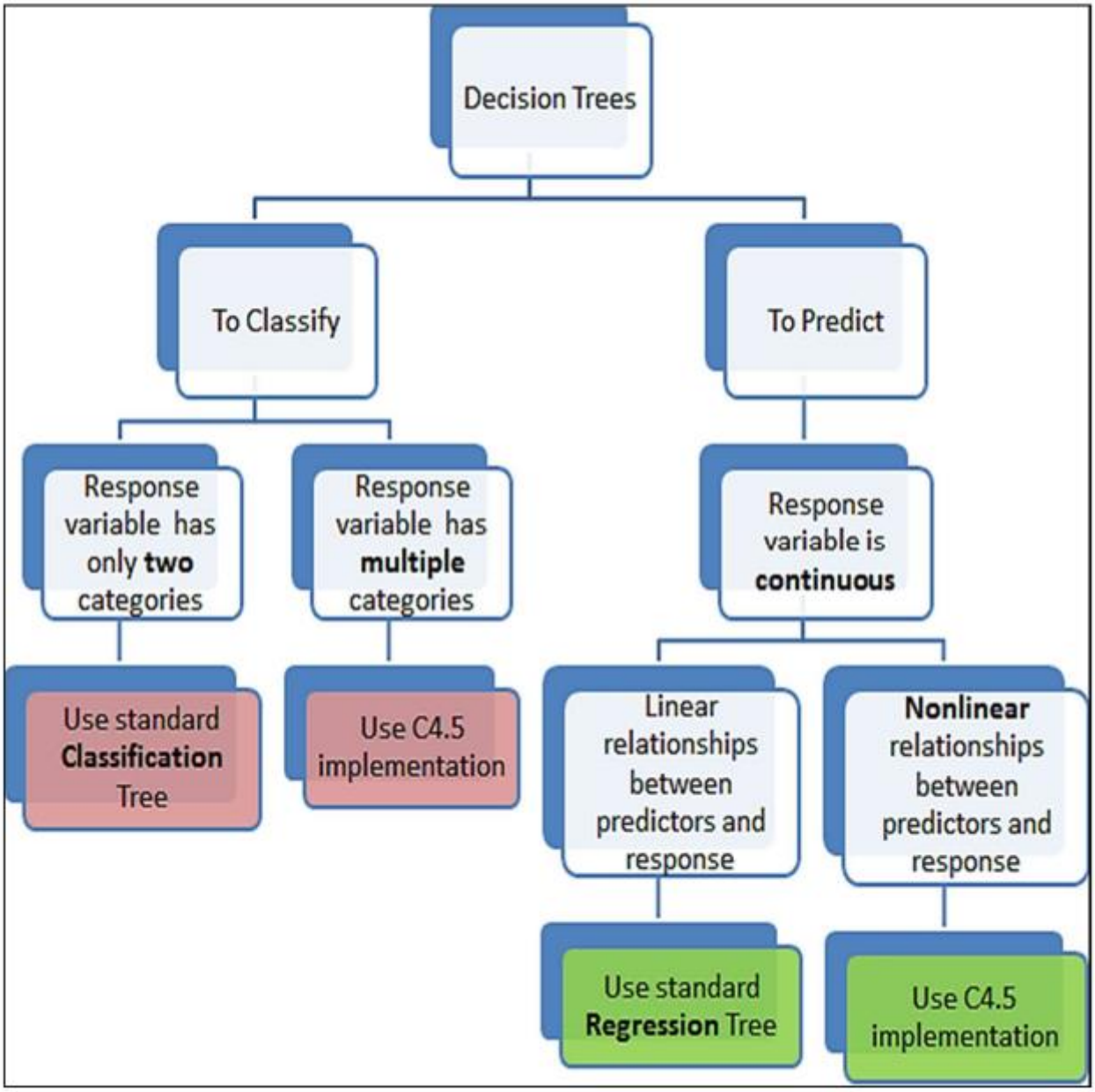
[1] Random Forest:

Random Forest is the most famous and it is considered as the best algorithm for machine learning. It is a supervised learning algorithm. To achieve more accurate and consistent prediction, random forest creates several decision trees and combines them together. The major benefit of using it is its ability to solve both regression and classification issues. When building each individual tree, it employs bagging and feature randomness in order to produce an uncorrelated tree forest whose collective forecast has much better accuracy than any individual tree's prediction. Bagging enhances accuracy of machine learning methods by grouping them together. In this algorithm, during the splitting of nodes it takes only random subset of nodes into an account. When splitting a node, it looks for the best feature from a random group of features rather than the most significant feature. This results into getting better accuracy. It efficiently deals with the huge datasets. It also solves the issue of overfitting in datasets. It works as follows: First, it'll select random samples from the provided dataset. Next, for every selected sample it'll create a decision tree and it'll receive a forecasted result from every created decision tree. Then for each result which was predicted, it'll perform voting and through voting it will select the best predicted result



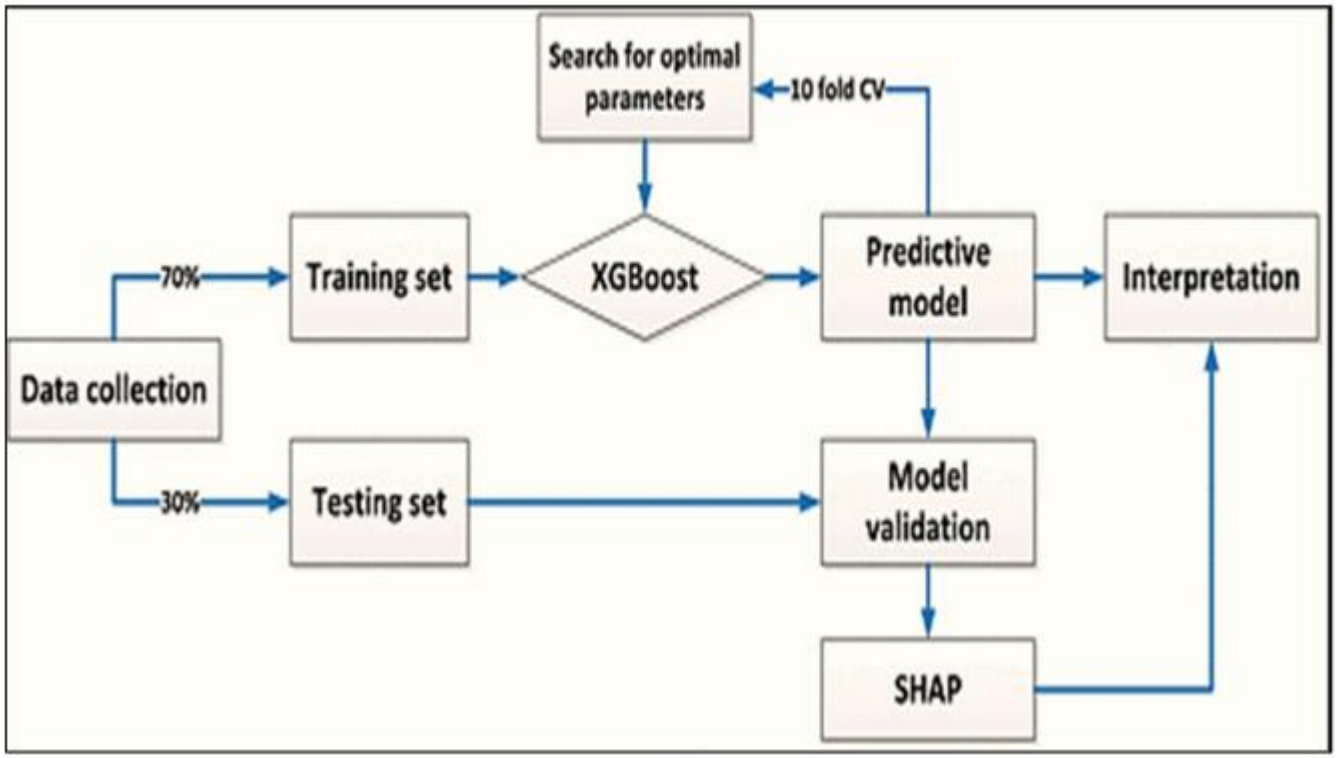
[2]Decision Tree Regressor:

The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data (training data). In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.



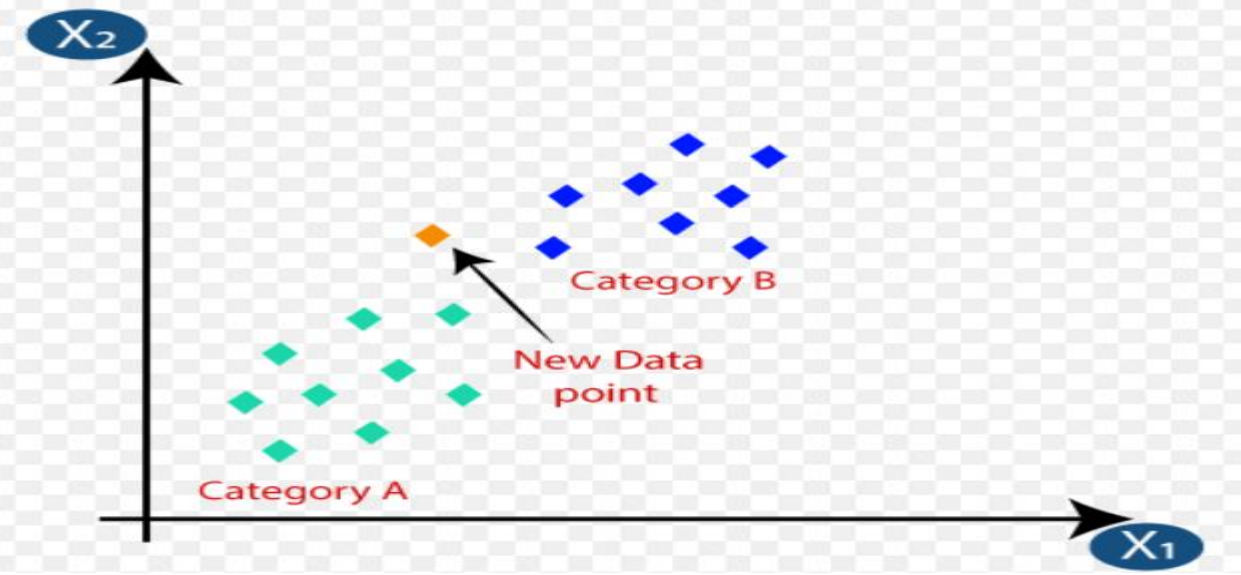
[3] XGBoost Regressor:

XGBoost is a powerful approach for building supervised regression models. The validity of this statement can be inferred by knowing about its (XGBoost) objective function and base learners. The objective function contains loss function and a regularization term. It tells about the difference between actual values and predicted values, i.e. how far the model results are from the real values. The most common loss functions in XGBoost for regression problems is reg: linear, and that for binary classification is reg: logistics. Ensemble learning involves training and combining individual models (known as base learners) to get a single prediction, and XGBoost is one of the ensemble learning methods. XGBoost expects to have the base learners which are uniformly bad at the remainder so that when all the predictions are combined, bad predictions cancels out and better one sums up to form final good predictions.



[4] K Neighbors Regressor Model:

Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.



Implementation on Dataset

As we already discussed in the methodology section about some of the implementation details. So, the language used in this project is Python programming. We're running python code in anaconda navigator's Jupyter notebook. Jupyter notebook is much faster than Python IDE tools like PyCharm or Visual studio for implementing ML algorithms. The advantage of Jupyter notebook is that while writing code, it's really helpful for Data visualization and plotting some graphs like histogram and heatmap of correlated matrices. Let's revise implementation steps.

a) Dataset collection.

b) Importing Libraries: Numpy, Pandas, Scikit-learn, Matplotlib and Seaborn libraries were used.

c) Exploratory data analysis: For getting more insights about data.

d) Data cleaning and preprocessing: Checked for null and junk values using isnull() and isna().sum() functions of python .In Preprocessing phase, we did feature engineering on our dataset. As we converted categorical variables into numerical variables using function of Pandas library. Both our datasets contains some categorical variables

e) Feature Scaling: In this step, we normalize our data by applying Standardization by using StandardScalar() and fit_transform() functions of scikit-learn library.

f) Model selection : We first separated X's from y's. X's are features or input variables of our datasets and y's are dependent or target variables which are crucial for predicting disease. Then using by the importing model_selection function of the sklearn library, we splitted our X's and y's into train and test split using train_test_split() function of sklearn. We splitted 70% of our data for training and 30% for testing.

g) Applied ML models and created a confusion matrix of all models. h) Deployment of the model which gave the best accuracy.

❖ Importing Libraries

➢ Python:

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

➢ Machine Learning:

- from sklearn.model_selection import train_test_split
- from sklearn.tree import DecisionTreeRegressor
- from sklearn.ensemble import RandomForestRegressor
- import xgboost as xgb
- from sklearn.neighbors import KNeighborsRegressor

Chapter 3

[A] Analysis of Dataset

1) Data Wrangling

In this we have to read the raw csv file with the help of pandas library , which gives us access to get the information of feature and record. So after analysing the data we can modifies and clean the dataset.

- features=pd.read_csv("features1.csv")
- store=pd.read_csv("stores1.csv")
- train=pd.read_csv("train1.csv")

❖ train.csv - CSV Data file containing following attributes

- Store
- Dept
- Date
- Weekly_Sales
- IsHoliday
- 115064 Data rows

❖ stores.csv - CSV Data File containing following attributes

- Store
- Type
- Size
- 45 Data rows

❖ features.csv - CSV Data file containing following attributes

- Date
- Temperature
- Fuel_Price
- MarkDown1
- MarkDown2
- MarkDown3
- MarkDown4
- MarkDown5
- CPI
- Unemployment
- IsHoliday
- 8190 Data rows

Processing and merging dataset

Merging all dataset and Processing to one dataset. Checking all dataset and then merging into one single dataset.

```
In [3]: features.shape
Out[3]: (8190, 12)
```

Observation=This dataset consist of records 8190 and Features 12

```
In [4]: features.head()
```

Out[4]:

	Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	Mari
0	1	05-02-2010	42.31	2.572	NaN	NaN	NaN	NaN	
1	1	12-02-2010	38.51	2.548	NaN	NaN	NaN	NaN	
2	1	19-02-2010	39.93	2.514	NaN	NaN	NaN	NaN	
3	1	26-02-2010	46.63	2.561	NaN	NaN	NaN	NaN	
4	1	05-03-2010	46.50	2.625	NaN	NaN	NaN	NaN	

Observation= Checking top five values

Training Dataset

```
n [7]: train.shape
ut[7]: (421570, 5)
```

Observation=This dataset consist of records 421570 and Features 5

```
n [8]: train.head()
```

ut[8]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
0	1	1	05-02-2010	24924.50	False
1	1	1	12-02-2010	46039.49	True
2	1	1	19-02-2010	41595.55	False
3	1	1	26-02-2010	19403.54	False
4	1	1	05-03-2010	21827.90	False

```
n [9]: train.tail()
```

ut[9]:

	Store	Dept	Date	Weekly_Sales	IsHoliday
421565	45	98	28-09-2012	508.37	False
421566	45	98	05-10-2012	628.10	False
421567	45	98	12-10-2012	1061.02	False
421568	45	98	19-10-2012	760.01	False
421569	45	98	26-10-2012	1076.80	False

Dataset containing data of Stores

In [11]:
```
store.shape
```

Out[11]: (45, 3)

In [12]:
```
store.head()
```

Out[12]:

	Store	Type	Size
0	1	A	151315
1	2	A	202307
2	3	B	37392
3	4	A	205863
4	5	B	34875

In [13]:
```
store.tail()
```

Out[13]:

	Store	Type	Size
40	41	A	196321
41	42	C	39690
42	43	C	41062
43	44	C	39910
44	45	B	118221

Merging Dataset

In [15]:
```
#reseting train
train=train.groupby(['Store','Dept','Date'])['Weekly_Sales'].sum()
train=train.reset_index()
train.head(10)
```

Out[15]:

	Store	Dept	Date	Weekly_Sales
0	1	1	01-04-2011	20398.09
1	1	1	01-06-2012	16065.49
2	1	1	01-07-2011	15363.50
3	1	1	01-10-2010	20094.19
4	1	1	02-03-2012	20113.03
5	1	1	02-04-2010	57258.43
6	1	1	02-07-2010	16333.14
7	1	1	02-09-2011	15277.27
8	1	1	02-12-2011	25293.49
9	1	1	03-02-2012	23510.49

In [16]:
```
#merging train and feature
data=pd.merge(train,features,on=['Store','Date'],how='inner')
data.head(10)
```

Out[16]:

	Store	Dept	Date	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3
0	1	1	01-04-2011	20398.09	59.17	3.524	NaN	NaN	NaN
1	1	2	01-04-2011	46991.58	59.17	3.524	NaN	NaN	NaN
2	1	3	01-04-2011	8734.19	59.17	3.524	NaN	NaN	NaN
3	1	4	01-04-2011	34451.90	59.17	3.524	NaN	NaN	NaN
4	1	5	01-04-2011	23598.55	59.17	3.524	NaN	NaN	NaN
5	1	6	01-04-2011	3249.27	59.17	3.524	NaN	NaN	NaN
6	1	7	01-04-2011	20144.71	59.17	3.524	NaN	NaN	NaN
7	1	8	01-04-2011	35319.05	59.17	3.524	NaN	NaN	NaN
8	1	9	01-04-2011	28388.25	59.17	3.524	NaN	NaN	NaN
9	1	10	01-04-2011	30863.93	59.17	3.524	NaN	NaN	NaN

In [17]:
```
data.shape
```

Out[17]: (421570, 14)

In [18]:
```
#merging store with data
data=pd.merge(data,store,on=['Store'],how='inner')
data
```

Out[18]:

	Store	Dept	Date	Weekly_Sales	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkD
0	1	1	01-04-2011	20398.09	59.17	3.524	NaN	NaN	
1	1	2	01-04-2011	46991.58	59.17	3.524	NaN	NaN	
2	1	3	01-04-2011	8734.19	59.17	3.524	NaN	NaN	
3	1	4	01-04-2011	34451.90	59.17	3.524	NaN	NaN	
4	1	5	01-04-2011	23598.55	59.17	3.524	NaN	NaN	
...	...	...	...	...	...	...	...	...	...
421565	45	93	31-12-2010	2072.46	29.67	3.179	NaN	NaN	
421566	45	94	31-12-2010	3966.80	29.67	3.179	NaN	NaN	
421567	45	95	31-12-2010	43149.88	29.67	3.179	NaN	NaN	
421568	45	97	31-12-2010	5881.22	29.67	3.179	NaN	NaN	
421569	45	98	31-12-2010	74.55	29.67	3.179	NaN	NaN	

In [20]:
```
data.shape
```

Out[20]:
```
(421570, 16)
```

In [21]:
```
data.columns
```

Out[21]:
```
Index(['Store', 'Dept', 'Date', 'Weekly_Sales', 'Temperature', 'Fuel_Price',
       'MarkDown1', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5', 'CPI',
       'Unemployment', 'IsHoliday', 'Type', 'Size'],
      dtype='object')
```

In [22]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 421570 entries, 0 to 421569
Data columns (total 16 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Store         421570 non-null  int64
 1   Dept          421570 non-null  int64
 2   Date          421570 non-null  object
 3   Weekly_Sales  421570 non-null  float64
 4   Temperature   421570 non-null  float64
 5   Fuel_Price    421570 non-null  float64
 6   MarkDown1     150681 non-null  float64
 7   MarkDown2     111248 non-null  float64
 8   MarkDown3     137091 non-null  float64
 9   MarkDown4     134967 non-null  float64
 10  MarkDown5     151432 non-null  float64
 11  CPI           421570 non-null  float64
 12  Unemployment  421570 non-null  float64
 13  IsHoliday     421570 non-null  bool
 14  Type          421570 non-null  object
 15  Size          421570 non-null  int64
dtypes: bool(1), float64(10), int64(3), object(2)
memory usage: 51.9+ MB
```

```
In [26]: data.isnull().sum().sum()

Out[26]: 1422431
```

❖ Observation:

- Dataset consist of 3 datasets Features, store and train.
- Firstly, we check all datasets their shape, information, and first 5 observations.
- Merging train and feature dataset into one dataset and then merging store dataset.
- We get one dataset i.e data and then we check all information of that dataset i.e shape, columns, info and describe and data types.
- Then we check null values in dataset.
- We have 1422431 null values in dataset.
- From observation most of null values in markdowns.
- The info () method prints information about the Data Frame. The information contains the number of columns, column labels, and column data types.
- The describe () = This method computes and displays summary statistics for a Python data Frame.

2) Data Cleaning/Preparation:

- As the dataset from kaggle was not very suitable for data analysis, we had to change the format of some data in the dataset. We also had to do separate data preparation for exploratory analysis and machine learning.
- Some of our data preparation were:
    - Convert price of listings from strings to floats and also remove the '$' sign.
    - Change NaN values to the integer 0.
    - Cleaning the textual data into a form that would be suitable for Python's Jupyter Notebook.
    - Encoding the categorical variables so that it can be fit into the regression models later.
    - Separating the data into Predicted and Target variables.
    - Separating the data into training and testing sets (Training Sets: Testing Sets = 70% : 30%).

```
In [27]: data=data.drop(['MarkDown1','MarkDown2','MarkDown3','MarkDown4','MarkDown5'],axi
         data.head(10)
```

Out[27]:

	Store	Dept	Date	Weekly_Sales	Temperature	Fuel_Price	CPI	Unemployment	IsH
0	1	1	01-04-2011	20398.09	59.17	3.524	214.837166	7.682	
148053	16	34	01-04-2011	4535.17	35.75	3.461	192.269171	6.339	
148054	16	35	01-04-2011	845.00	35.75	3.461	192.269171	6.339	
148055	16	36	01-04-2011	1172.50	35.75	3.461	192.269171	6.339	
148056	16	38	01-04-2011	57659.83	35.75	3.461	192.269171	6.339	
148057	16	40	01-04-2011	33428.25	35.75	3.461	192.269171	6.339	
148058	16	41	01-04-2011	2555.26	35.75	3.461	192.269171	6.339	
148059	16	42	01-04-2011	2998.87	35.75	3.461	192.269171	6.339	
148060	16	44	01-04-2011	3894.04	35.75	3.461	192.269171	6.339	
148061	16	45	01-04-2011	11.47	35.75	3.461	192.269171	6.339	

```
In [28]: data.drop(['Date'],axis=1,inplace=True)
```

Observation:

- In order to make data clean and more readable to drop markdowns
- We also drop date as it was not relevant in EDA.
- We have 1422431 null values in dataset.
- We drop in order to make dataset clean.

observation:

- MarkDown1 270889
- MarkDown2 310322
- MarkDown3 284479
- MarkDown4 286603
- MarkDown5 270138
- So these are null value which we drop to make data clean.

3) Feature Engineering:

In the Feature Engineering we change the data type of feature for cleaning and better understanding of data so we can analyze and implement the algorithm on it.

Converting IsHoliday in Holiday which is integer and 1 for holiday and 0 otherwise.

In [33]:
```
data['Holiday']=[int(i) for i in list(data.IsHoliday)]
data.head(10)
```

Out[33]:

	Store	Dept	Weekly_Sales	Temperature	Fuel_Price	CPI	Unemployment	IsHoliday
0	1	1	20398.09	59.17	3.524	214.837166	7.682	False
148053	16	34	4535.17	35.75	3.461	192.269171	6.339	False
148054	16	35	845.00	35.75	3.461	192.269171	6.339	False
148055	16	36	1172.50	35.75	3.461	192.269171	6.339	False
148056	16	38	57659.83	35.75	3.461	192.269171	6.339	False
148057	16	40	33428.25	35.75	3.461	192.269171	6.339	False
148058	16	41	2555.26	35.75	3.461	192.269171	6.339	False
148059	16	42	2998.87	35.75	3.461	192.269171	6.339	False
148060	16	44	3894.04	35.75	3.461	192.269171	6.339	False
148061	16	45	11.47	35.75	3.461	192.269171	6.339	False

In [46]:
```
data.Type.replace({'A':0,'B':1,'C':2},inplace=True)
```

Observation:

- Converting IsHoliday into Holiday into 1 for holiday and o otherwise.
- Replace Type of store A, B and C into 0, 1 and 2 for machine learning models.

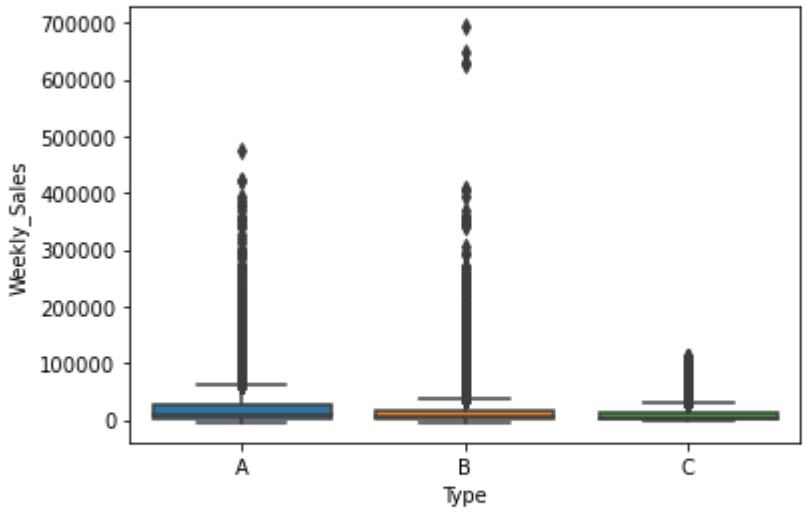
4) Exploratory Data Analysis & Problem Formulation on Walmart Dataset:

➢ Problem Statement

- Historical sales data for 45 walmart stores located in different regions is taken. As sales play most important roles in any business.
- Predicting output values i.e sales is a regression problem and machine learning can be applied for this problem.
- Goal is to predict sales.
- Dataset contains a few non-numeric columns, which would need to be converted to numerical.
- Once the data has been pre-processed, multiple regression model would be evaluated and finally the best model for this problem would be chosen.

- Sales according to Store Types

In [39]:
```
sns.boxplot(x='Type',y='Weekly_Sales',data=data)
plt.show()
```



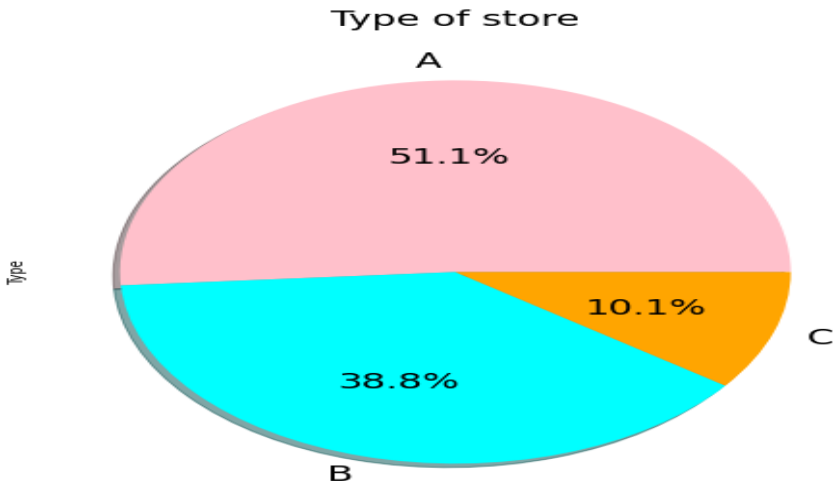
Obsevation= There is trace of outliers which can be removed in data processing

Problem Statement

Store distribution according to Type

Pie Chart

In [40]:
```
plt.figure(figsize=(10,8))
plt.title('Type of store',fontsize=20)
data['Type'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True,colors=['pink
plt.show()
```



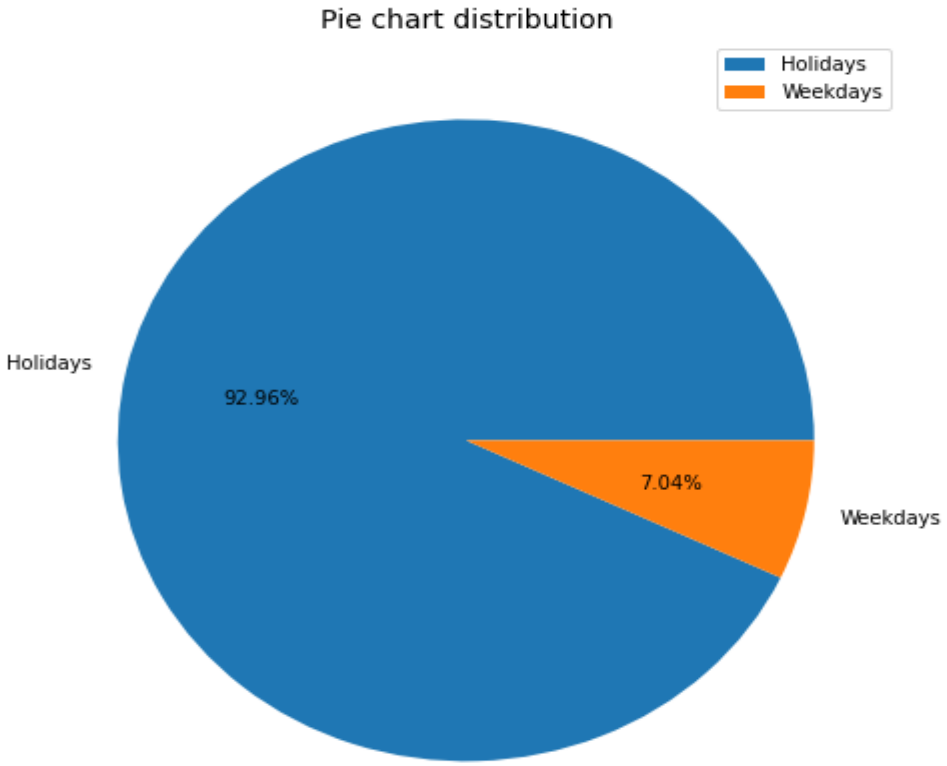
Observation=Store wise distribution of weekly sales

- store type A has highest sale among other two store type

Problem Statement

Sales based on weekdays and Holidays

In [41]:
```
plt.figure(figsize=(8,8))
plt.pie(data['IsHoliday'].value_counts(),labels=['Holidays','Weekdays'],autopct=
plt.title("Pie chart distribution",fontsize=14)
plt.legend()
plt.show()
```

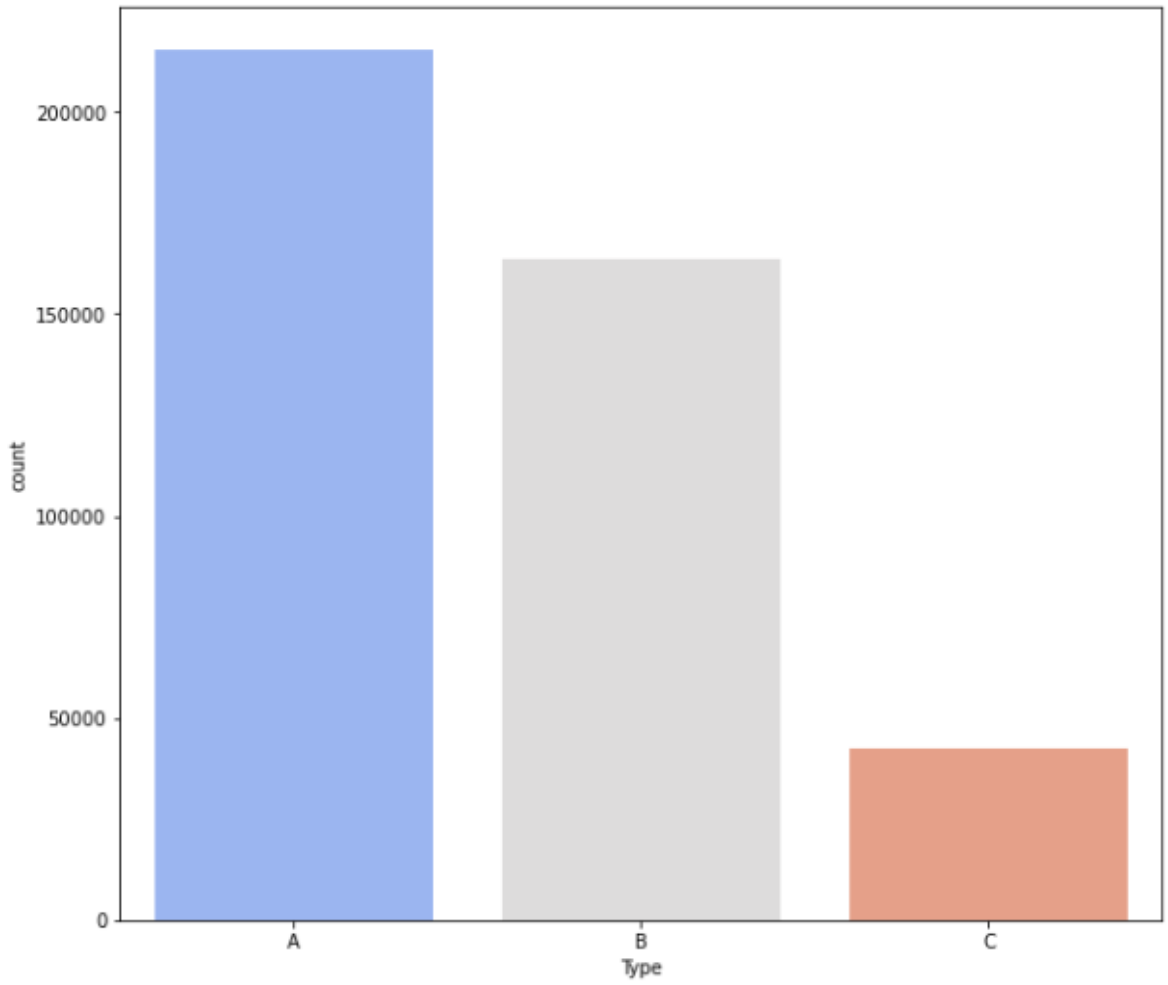


Observation:

- Maximum sales are on holidays as compared to weekdays

Count Plot

In [42]:
```
plt.figure(figsize=(10,9))
sns.countplot(x="Type", data=data,palette='coolwarm')
plt.show()
```

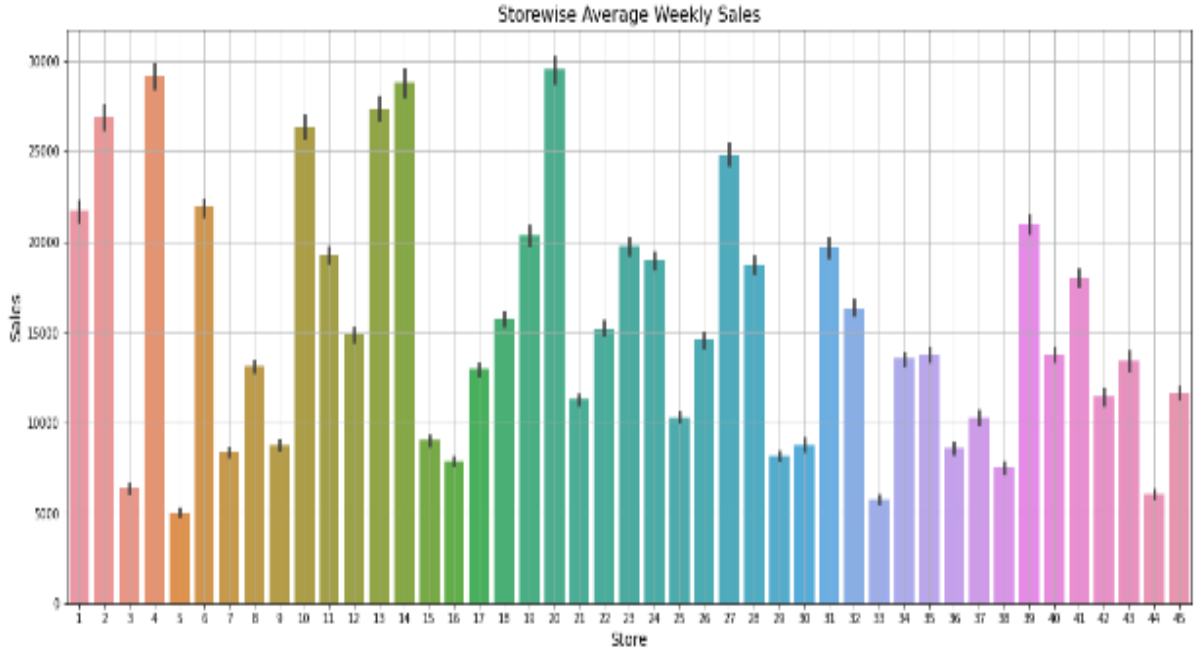


Observation:

- In Countplot store type A has highest sale among two other store type.

Barplot

In [43]:
```
plt.figure(figsize=(20,8))
sns.barplot(x='Store',y='Weekly_Sales',data=data)
plt.ylabel('Sales',fontsize=14)
plt.xlabel('Store',fontsize=14)
plt.title('Storewise Average Weekly Sales',fontsize=16)
plt.grid()
plt.show()
```

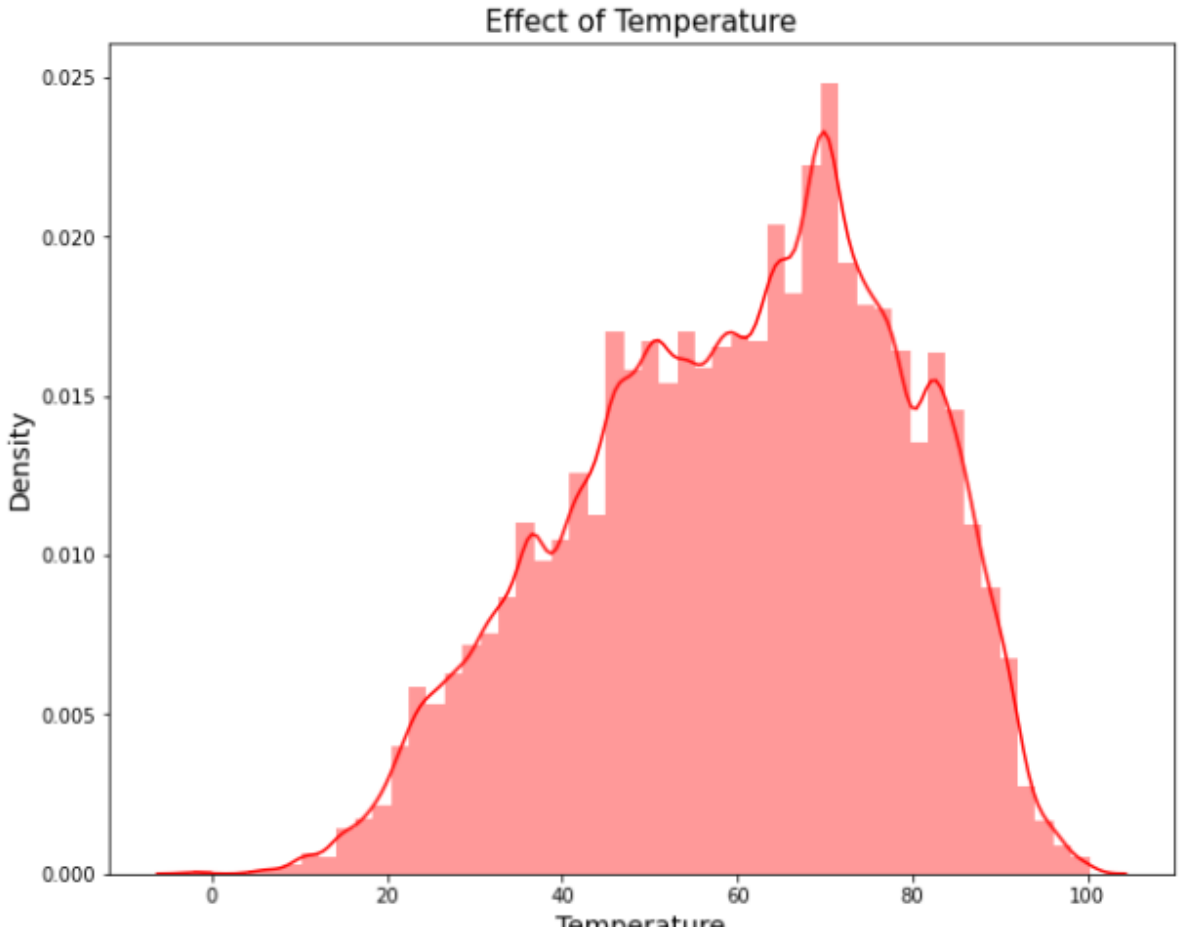


Observation:

- Store no 20 has highest Sales according to barplot

Dist Plot

In [44]:
```
plt.figure(figsize=(10,8))
sns.distplot(data['Temperature'],color='red')
plt.title('Effect of Temperature',fontsize=15)
plt.xlabel('Temperature',fontsize=14)
plt.ylabel('Density',fontsize=14)
plt.show()
```



Observation:

- Effect of Temperature on sales.

Machine Learning Model Implementation

(1) Convert Values

```
In [46]: data.Type.replace({'A':0,'B':1,'C':2},inplace=True)
```

```
In [45]: from sklearn import metrics
         from sklearn.model_selection import train_test_split
         from sklearn.linear_model import LinearRegression
         from sklearn.ensemble import RandomForestRegressor
         from sklearn.neighbors import KNeighborsRegressor
```

```
In [46]: data.Type.replace({'A':0,'B':1,'C':2},inplace=True)
```

```
In [47]: X = data.drop(['Weekly_Sales'],axis=1)
         y = data.Weekly_Sales
```

```
In [48]: ain,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20, random_state=1)
```

```
In [49]: data.columns
```

```
Out[49]: Index(['Store', 'Dept', 'Weekly_Sales', 'Temperature', 'Fuel_Price', 'CPI',
                'Unemployment', 'IsHoliday', 'Type', 'Size', 'Holiday'],
               dtype='object')
```

➢ Observation:

- In dataset we have feature name Type in which we convert into integer.
- Then we split data into dependent and Independent.
- Where dependent is (y) i.e weekly_sales.
- Whereas independent is (X).
- That includes other feature except weekly_sales.

Decision Tree Regressor Model

```
In [50]: from sklearn.tree import DecisionTreeRegressor

In [51]: model_DT = DecisionTreeRegressor()

In [52]: model_DT

Out[52]: DecisionTreeRegressor()

In [53]: model_DT.fit(X_train, y_train)

Out[53]: DecisionTreeRegressor()

In [54]: BDT=model_DT.score(X_test, y_test)*100
         BDT

Out[54]: 89.35276487632198
```

Decision Tree Regressor gives us 89% of Accuracy.

Observation:

- Decision Tree Regressor gives us 89% of Accuracy.

Random Forest Regressor Model

```
In [55]: from sklearn.ensemble import RandomForestRegressor

In [56]: model_RFR = RandomForestRegressor()

In [57]: model_RFR

Out[57]: RandomForestRegressor()

In [58]: model_RFR.fit(X_train,y_train)

Out[58]: RandomForestRegressor()

In [59]: RFR =model_RFR.score(X_test, y_test)*100
         RFR

Out[59]: 94.34931683764293
```

Random Forest Regressor gives us 94% of Accuracy

Observation:

Random Forest Regressor gives us 94% of Accuracy.

XGBoost Model

In [60]:
```
import xgboost as xgb
```

In [61]:
```
model_xgb = xgb.XGBRegressor()
```

In [62]:
```
model_xgb
```

Out[62]:
```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

In [63]:
```
model_xgb.fit(X_train, y_train)
```

Out[63]:
```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, feature_types=None, gamma=0, gpu_id=-1,
             grow_policy='depthwise', importance_type=None,
             interaction_constraints='', learning_rate=0.300000012, max_bin=256,
             max_cat_threshold=64, max_cat_to_onehot=4, max_delta_step=0,
             max_depth=6, max_leaves=0, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=0,
             num_parallel_tree=1, predictor='auto', random_state=0, ...)
```

In [64]:
```
xgb = model_xgb.score(X_test, y_test)*100
```

In [65]:
```
xgb
```

Out[65]:
```
91.68118105991944
```

XGBoost Regressor gives us 91% of Accuracy.

Observation:

XGBoost Regressor gives us 91% of Accuracy.

K Neighbors Regressor Model

```
In [66]: from sklearn.neighbors import KNeighborsClassifier

In [67]: knn = KNeighborsRegressor(weights = 'uniform')

In [68]: model_knn=knn.fit(X_train,y_train)

In [69]: knn_acc = knn.score(X_test, y_test)*100
         print("KNeigbhbors Regressor Accuracy - ",knn_acc)

         KNeigbhbors Regressor Accuracy -  59.24557905010239
```

KNeigbhbors Regressor gives Accuracy 59%

Observation:

K Neighbors Regressor gives Accuracy 59%.

Machine Learning Model Comparison

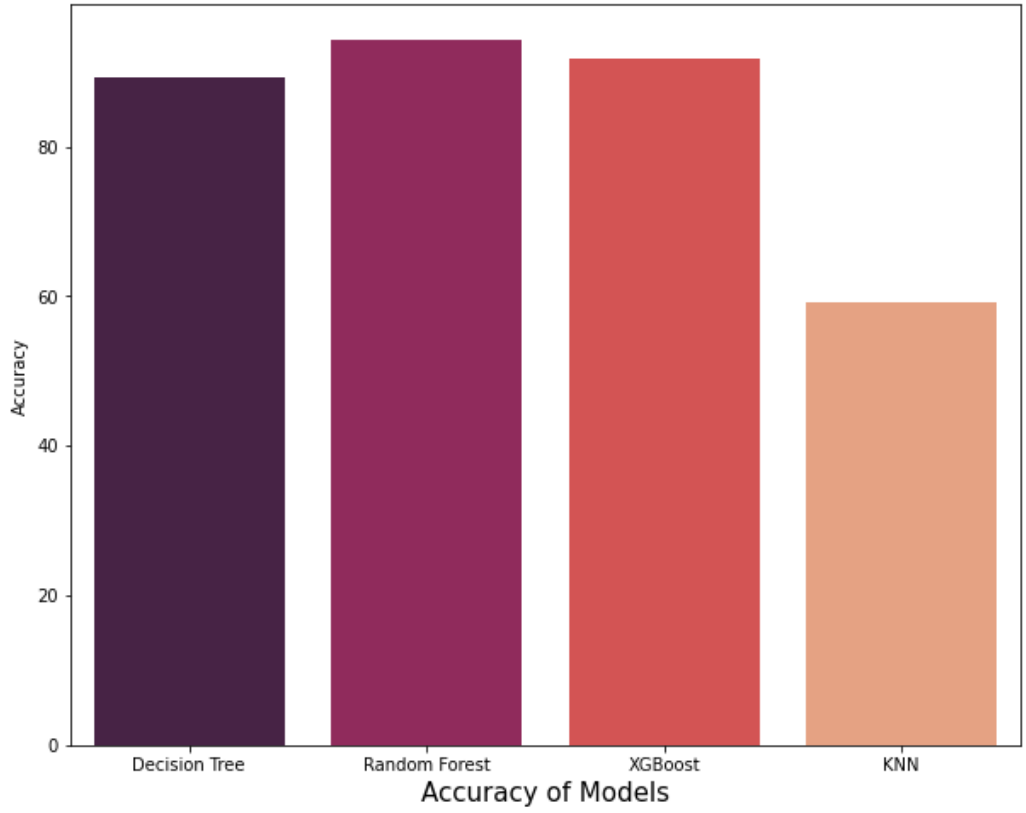
In [70]:
```
Models = pd.DataFrame({'Model': [ 'Decision Tree',
 'Random Forest','XGBoost','KNN'],
 'Accuracy': [BDT, RFR, xgb , knn_acc]})
```

In [71]:
```
Models
```

Out[71]:

	Model	Accuracy
0	Decision Tree	89.352765
1	Random Forest	94.349317
2	XGBoost	91.681181
3	KNN	59.245579

In [72]:
```
plt.figure(figsize=(10,8))
sns.barplot(x='Model',y='Accuracy',data=Models,palette='rocket')
plt.xlabel('Accuracy of Models', fontsize =15)
plt.show()
```



Observation:

Decision Tree, Random Forest and XGBoost has accuracy greater than 85%

Conclusion

- Following inferences and conclusions can be drawn from the analysis of the data:
- Type 'A' stores are more popular than 'B' and 'C' types.
- Type 'A' stores outclass the 'B' and 'C' types in terms of size and the average weekly sales.
- Weekly Sales are affected by the week of year. Holiday weeks witnessed more sales than the non-holiday weeks. Notables are Thanksgiving and Christmas weeks.
- Size of the store is a major contributing factor in the weekly sales.
- Sales are also dependent on the department of the store as different departments showed different levels of weekly sales.
- Among the trained models for predicting the future sales, Random Forest Machine with tuned hyperparameters performs the best.

References

1.J.R. Quinlan (1986). "Induction of Decision Trees". Machine Learning. 1: 81– 106. doi:10.1007/BF00116251.

2. Nick Littlestone (1988). "Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm"

3. "General Python FAQ — Python 3.9.2 documentation". docs.python.org. Archived from the original on 24 October 2012. Retrieved 28 March 2021.

4. The Python Library Reference by Guido van Rossum Fred L. Drake, Jr., editor.

5. Dataset from Kaggle:

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