



UNIVERSITY OF HYDRABAD

M5 Forecasting - Accuracy

Estimate the Unit Sales of Walmart Retail Goods

by

Akansh Sharma

A thesis submitted in fulfilment for the
Diploma Program

Under the Guidance of

N. Brahma Reddy

(and other Supervisors)

Abstract

Background: Department stores like Walmart have uncountable product and money transactions every day. Because of their rapid transaction rates, keeping a balance between the inventory and customer demand is the most important decision for the managers. Therefore, making an accurate sales prediction for different products becomes an essential need for stores to optimize their profits. Most of the existing sales predictions only depend on extrapolating the statistical trend. The previous studies on market sales prediction require a lot of extra information like customer and product analysis. A more simplified model is needed by the department store to predict the product sales based on only the historical sales record. The new emerging machine learning methods enable us to make more accurate predictions like that. Our data is from the Kaggle M5 Competition.

Objective: This thesis will help to identify the critical features that influence sales and an experiment is performed to find the best suitable algorithm/model for sales forecasting.

Methods: For experimentation machine learning algorithms such as Simple Linear Regression, Decision Tree, Random Forest along with some optimization technique and hyperparameter tuning parameters were considered to achieve best result in this thesis, which they expected to perform well on the issues. Also, we have used performance metric to identify which algorithm will perform better for future unseen data.

Feature Engineering: We have used Lags, Simple moving average and as new features as part of feature engineering.

Conclusions: The Light Gradient Boosting Machine algorithm (LightGBM) performed well after doing all the study when compared with other algorithms. Hence the Light GBM is considered the best suitable algorithm for forecasting the next 28-days period of Walmart sales using the historical sales records, price and calendar information.

Contents

Abstract	ii
1. Introduction	iv
1.1. Aim and Objective	vi
1.2. Source of the Data	vi
1.3. What is the Real Problem?	vi
1.4. What are the Business Constraints?	v
1.5. Why is this an important problem to solve?	vi
1.6. Traditional Solutions Available	vi
1.7. Why ML models better than Classical Methods?	vi
1.8. Performance metric used	vi
2. Related Work	viii
2.1 Existing approaches to the problem	viii
2.2 What improvements can be done?	viii
3. Hypothesis	ix
4. Data Visualization	x
4.1 Dataset Overview	x
4.2 Exploratory Data Analysis	xi
5. Data Preparation	xvii
5.1 Data Downcasting	xvii
5.2 Data Melting and Merging	xvii
6. First Cut Solution	xix
6.1 Without Hyperparameter & Feature Engineering	xix
6. Feature Engineering	xx
6.1 Handling Missing values	xx
6.2 Encoding Categorical data	xx
6.3 Adding Time-Based features	xx
6.4 Basic Feature Engineering for Time Series Problem	xx
6.5 Splitting the dataset into training and test dataset	xxi
7. ALGORITHMS USED	xxii
7.1 Linear Regression	xxii
7.2 Decision Tree Regressor	xxiv
7.3 Random Forest Regressor	xxiv
7.4 LightGBM Regressor	xxv
8. Conclusion	xxv
8.1 Models Comparison	xxvii
8.2 Kaggle Score	xxvii
8.3 Final Conclusion	xxviii
8.4 Future Work	xxviii
9. References	xxix

1. Introduction

Earlier companies used to produce goods without considering the number of sales and demand. For any manufacturer to determine whether to increase or decrease the production of several units, data regarding the demand for products on the market is required. Companies can face losses if they fail to consider these values while competing on the market. Different companies choose specific criteria to determine their demand and sales.

In today's highly competitive environment and ever-changing consumer landscape, accurate and timely forecasting of future revenue, also known as revenue forecasting, or sales forecasting, can offer valuable insight to companies engaged in the manufacture, distribution or retail of goods. Short-term forecasts primarily help with production planning and stock management, while long-term forecasts can deal with business growth and decision-making.

Sales forecasting is particularly important in the industries because of the limited shelf-life of many of the goods, which leads to a loss of income in both shortage and surplus situations. Too many orders lead to a shortage of products and still too few orders lead to a lack of opportunity. Therefore, competition in the food market is continuously fluctuating due to factors such as pricing, advertisement, increasing demand from the customers.

Managers usually make sales predictions randomly. Professional managers, however, become hard to find and not always available (e.g., they can get sick or leave). Sales predictions can be assisted by computer systems that can play the qualified managers' role when they are not available or allow them to make the right decision by providing potential sales predictions. One way of implementing such a method is to try and model the professional managers' skills inside a computer program.

Alternatively, the abundance of sales data and related information can be used through Machine Learning techniques to automatically develop accurate sales predictive models. This approach is much simpler. It is not prejudiced by a single sales manager's particularities and is flexible, which means it can adapt to data changes. It has, however, the potential to overestimate the accuracy of the prediction of a human expert, which is normally incomplete. For example, once companies used to produce the products without taking into consideration the number of sales and demand as they faced several problems. Since they don't know how much to sell, for any manufacturer to decide whether to increase or decrease the number of units, data regarding the consumer demand for products is essential. If companies do not consider these principles when competing in the market, they will face losses. Different companies choose different parameters to determine their market and sales.

There are several ways of forecasting sales in which companies have previously focused on various statistical models such as time series and linear regression, feature engineering and random forest models to obtain future sales and demand prediction. Time series contains data points that are stored over a fixed period and are used to forecast the future. Time series is a collection of data points which are collected in period at sequential, evenly spaced points. The most important components to analyse are patterns, seasonality, irregularity, cyclicity.

1.1 Aim and Objective

The University of Nicosia has organized a Kaggle's competition in which the challenge is to estimate, as precisely as possible, the point forecast of the unit sales of various products sold in the USA by Walmart.

This thesis aims to develop a Machine Learning model that can predict the sales of product from different outlets.

Several Objectives were drawn to attain the goal:

- Converting data into an appropriate form using various pre-processing techniques for the implementation of Machine Learning algorithms.
- Finding critical features that will most influence sales of the product.
- To determine the appropriate Machine Learning algorithm for the product unit sales forecasting in the USA-based Walmart of 30.490 product for the next 28 days.
- Selecting various metrics to compare the performance of the applied Machine Learning algorithms.

1.2 Source of the Data

This dataset was provided by Walmart, the world's largest company by revenue to forecast daily sales for the next 28 days. The data covers 10 stores in three US States (California, Texas and Wisconsin) and includes item level, department, product categories and store details.

Link to the data - <https://www.kaggle.com/c/m5-forecasting-accuracy>

The M5 dataset mainly contains of the following three files:

- Calendar.csv
- Sell_prices.csv
- Sales_train.csv

1.3 What is the Real Problem?

Sales forecasting is nothing but forecasting future sales based on historical data or more specifically a sales forecast is a projection of future revenue within a specific period – typically a week, a month or a quarter.

The historical daily unit sales data, price per product and store data was given by Kaggle.com, our task is to predict item sales at stores in various locations for the following 28-day time periods. The supplemental data was also given, which include the promotions, day of the week, and special events. These datasets can be used to improve the accuracy of prediction and drawing a closer insight on the performance of purchasing behaviours towards different products.

1.4 What are the Business Constraints?

- High Interpretability - as we need to know what the important factors are contributed in sales forecasting
- No strict low latency - as we need to forecast sales daily as compared to minute or hourly basis
- Incorrect forecasting may lead to missed business opportunities or losses

1.5 Why is this an important problem to solve?

Accurate sales forecasting is the foundation for success. It enables smart decision-making, informs budgets and helps detect potential threats before it's too late.

Few of the broader benefits that sales forecasting brings to the table are:

- Estimate future revenues
- Proper resource allocations as per the demand
- Planning of growth strategy i.e., spending, investing and hiring at the right rates at the right time.

Real World impact of solving this problem

An accurate sales forecasting is critical for retail companies to produce the required quantity at the right time and this helps them in doing proper strategic planning like the inventory distribution of the product as per the forecasting and hiring additional staff in areas where the sales are more.

1.6 Traditional Solutions Available

Traditionally, sales forecasting is accomplished by statistical methods. In fact, a lot of statistical methods have been used for sales forecasting, which includes moving average, weighted average, exponential smoothing (used when a trend is present but not linear), exponential smoothing with the trend, double exponential smoothing, Bayesian analysis, and so forth. Statistical time series analysis tools such as ARIMA and SARIMA are widely employed in sales forecasting.

The problem with these models/methods was that they don't take categorical variables into account and tend to work for short predictions only and are unable to provide the degree of interoperability.

1.7 Why ML models better than Classical Methods?

The problem we are solving is a time series problem that we can transform into a supervised learning problem by performing Feature engineering on the raw time-series data.

It is found that Classical method may dominate for one step forecasting

While ML Models work better in case of Multi-Step Forecasting. Here we are working with Multi-step forecasting thus ML Models will be a better choice. We can solve the posing the time series data for forecasting using Machine learning Regression models.

1.8 Performance metric used

A business metric is basically a quantifiable measure that helps a business measure certain aspect of their operations to track, monitor and measure certain aspects of their operations to achieve success.

For this problem, I have used Root Mean Squared Error (RMSE).

- Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors).
- Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are.
- In other words, it tells you how concentrated the data is around the line of best fit. Root mean square error is commonly used in climatology, forecasting, and regression analysis to verify experimental results.
- RMSE does not treat each error the same. It gives more importance to the most significant errors. That means one big error is enough to get a bad RMSE.

$$RMSE = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}}$$

where,

n = number of datapoint

y_i = actual value

\hat{y}_i = predicted value

Code Implementation

```
# Function for : RMSE

def RMSE_calc(pred,actual):
    ...
    This function is used to calculate: Root Mean Squared Error
    ...
    return np.sqrt(((pred-actual)**2).mean())
```

2. Related Work

2.1 Existing approaches to the problem

Previously a lot of sales and demand forecasting work was performed using Machine Learning. The forecasting methods used by the first two winners can be summarized as follows.

- **First Placed Solution (YJ_STU; Yeon Jun In):** [Link to - 1st Placed Solution](#)
 - The winner of the competition used an equal weighted combination of various LightGBM models, which were trained to produce forecasts for the product-store series using data pooled per store (10 models), store-category (30 models), and store-department (70 models).
 - In total, 220 models were built, and each series was forecast using the average of 6 models which includes two variations of each type of model, recursive and non-recursive.
 - The models considered various identifiers, calendar-related information, special events, promotions, prices, and unit sales data, both in a recursive and a non-recursive format.
 - The method was fine-tuned using the last four 28-day-long windows of available data for CV and by measuring both the mean and the standard deviation of the errors produced by the individual models and their combinations.
 - The final solution was chosen because it was providing both accurate and robust forecasts.
- **Second Place Solution (Matthias):** [Link to - 2nd Placed Solution](#)
 - This method also used equally weighted LightGBM models, however, was externally adjusted through multipliers according to the forecasts produced by N-BEATS.
 - Total 50 models were trained in which first trained per store 10 models and then five different multipliers were used to adjust their forecast.
 - LightGBM models were trained using only some basic features about calendar effects and prices (past unit sales were not considered), while the N-BEATS model was based solely on historical unit sales.

2.2 What improvements can be done?

The solution discussed on the paper were used by first and second-place winners.

Their solutions are very good, but the main issue was it's too complicated to understand and implement when you have very low-end devices. As we already discussed, the dataset has a very huge size.

We'll try to implement a simple solution with some good scores.

3. Hypothesis

3.1 Single/Multiple Hypothesis

Based on the data given some of the factors that may affect sales are:

- **Day:** Customers shopping time and spending mostly depends on the weekend. Many customers may like to shop only at weekends.
- **Special Events/Holidays:** Depending on the events and holidays customers purchasing behaviour may change. For holidays like Easter, food sales may go up and for sporting events like Superbowl finals Household item sales may go up.
- **Product Price:** The sales are affected the most by the product price. Most customers will check the price tag before making the final purchase.
- **Product Category:** The type of product greatly affects sales. For instance, products in the household like TV will have fewer sales when compared with sales of food products.
- **Location:** The location also plays an important role in sales. In states like California, the customers might buy products they want irrespective of price, and customers in another region may be price sensitive.
- **Trend:** Our hypothesis is also based on the assumption that customer's demand and purchasing habits are also in some pattern along the time.

4. Data Visualization

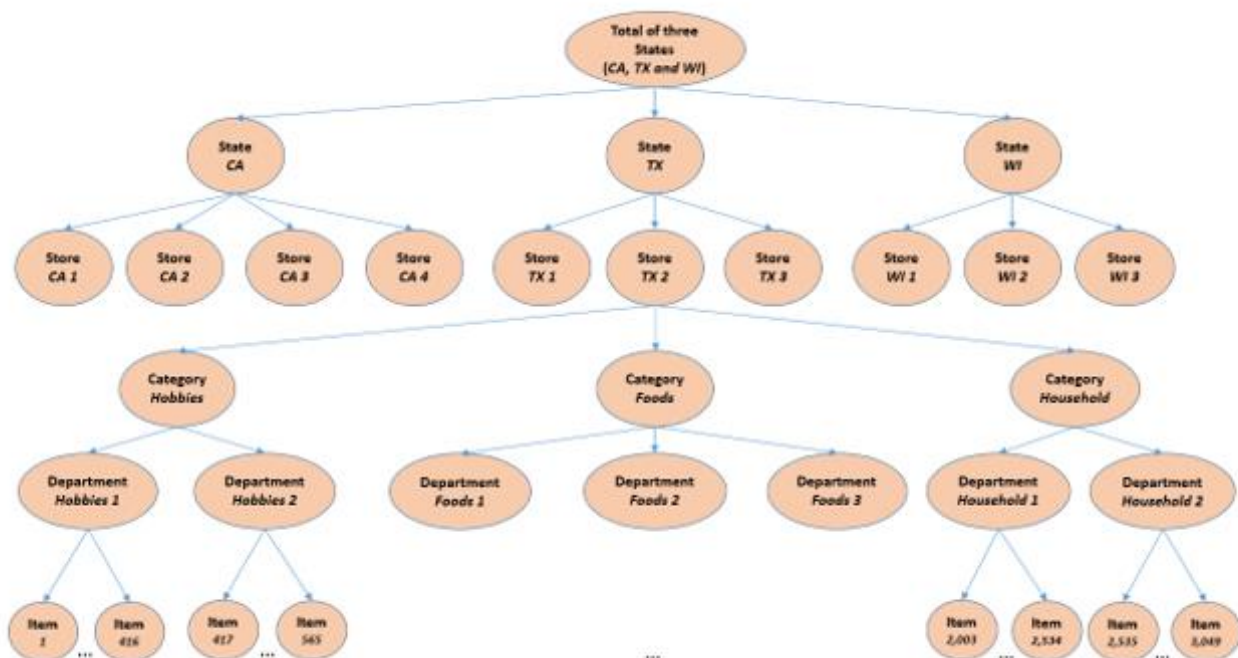
4.1 Dataset Overview

The data was provided by Kaggle website: M5 Forecasting - Accuracy, Estimate the unit sales of Walmart retail goods. The data covers stores in three US States (California, Texas, and Wisconsin) and includes item level, department, product categories, and store details.

In simple, the dataset involves the unit sales of 3,049 products, classified into 3 product categories (Hobbies, Foods, and households) and 7 product departments, and products are sold across 10 stores, located in three states (CA, TX, and WI).

The historical data range from 2011-01-29 to 2016-06-19. Thus, the products have a (maximum) selling history of 1,941 days. Besides the time series data, calendar dataset and prices dataset have also included explanatory variables such as price, promotions, day of the week, and special events that affect sales which together are used to improve forecasting accuracy.

Structure of the data



(Source : [M5 Competition Guide](#))

Information about Files:

Calendar Data: Contains information about the dates on which the products are sold along with some self-explanatory features like day, week & month.

It also contains information about special events names and type of event if happened on that day. Also features that indicate if snap purchases were allowed on that day.

About SNAP Program:

In the United States, SNAP - Supplemental Nutrition Assistance Program, formerly known as the Food Stamp Program, is a federal program that provides food-purchasing assistance for low and no-income people.

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1	event_name_2	event_type_2	snap_CA	snap_TX	snap_WI
1964	15-06-2016	11620	Wednesday	5	6	2016	d_1965	NaN	NaN	NaN	NaN	0	1	1
1965	16-06-2016	11620	Thursday	6	6	2016	d_1966	NaN	NaN	NaN	NaN	0	0	0
1966	17-06-2016	11620	Friday	7	6	2016	d_1967	NaN	NaN	NaN	NaN	0	0	0
1967	18-06-2016	11621	Saturday	1	6	2016	d_1968	NaN	NaN	NaN	NaN	0	0	0
1968	19-06-2016	11621	Sunday	2	6	2016	d_1969	NBAFinalsEnd	Sporting	Father's day	Cultural	0	0	0

Sell Price Data: Contains information about the price of the products sold per store and date. The sell price of each product is marked on a weekly basis.

	store_id	item_id	wm_yr_wk	sell_price
0	CA_1	HOBBIES_1_001	11325	9.58
1	CA_1	HOBBIES_1_001	11326	9.58
2	CA_1	HOBBIES_1_001	11327	8.26

Sales Data: This file has combination of id, item id, dept id, cat id, store id, state id and historical daily sales unit data from d_1 to d_1941 i.e., 1941 days of data which contains units sold for that id on that day.

	id	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	...	d_1932	d_1933	d_1934	d_1935	d_1936
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	2	4	0	0	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	1	2	1	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	0	2	0	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	1	0	4	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	0	2	0
5	HOBBIES_1_006_CA_1_evaluation	HOBBIES_1_006	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	2	1	0	0	0
6	HOBBIES_1_007_CA_1_evaluation	HOBBIES_1_007	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	1	0	0	0

4.2 Exploratory Data Analysis

To solve the problem through machine learning, first, we have to understand about the data which is an important approach is known as Exploratory data analysis.

Using EDA, we have to find two important things:

- whether Seasonality exists in feature
- whether Trend exists in feature

Basic Info about the data

Products sold in the USA, organized in the form of grouped time series. More specifically, the dataset involves the unit sales of 3,049 products, classified into 3 product categories (Hobbies, Foods, and Household) and 7 product departments. The products are sold across ten stores, located in three states (CA, TX, and WI).

- **Analysis on State division:**

In CA state nearly 70 percent sales are happening on food category and very few sales on hobbies.

In all states, higher sales on food category and very less sales on households and each state almost falling same distribution of sales for each category.

- **Analysis on Store division:**

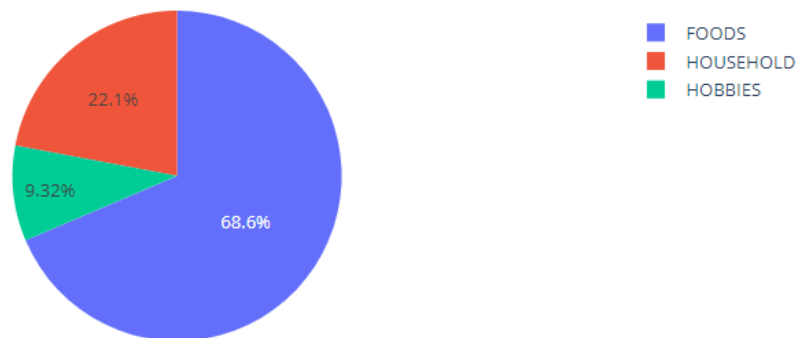
In California: CA_3 has sold maximum items while CA_4 has sold least number of items

In Texas: TX_2 has sold max items while TX_1 has sold least number of items

In Wisconsin: WI_2 has sold max number of items while WI_1 has sold least number of items

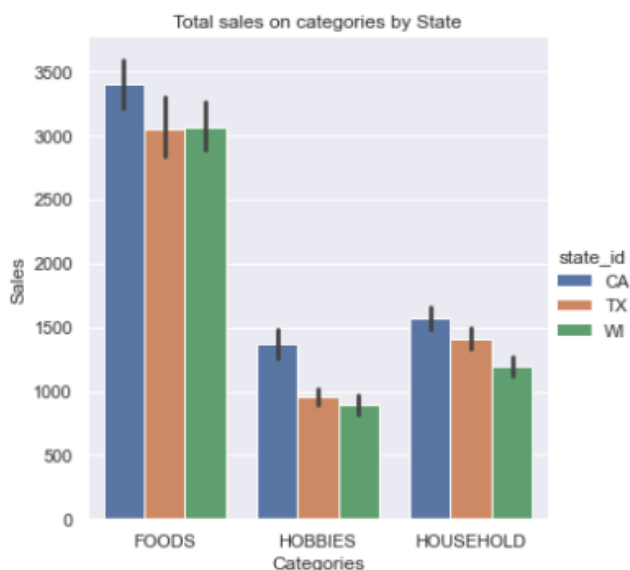
- **Total Sales Based on Product Categories Across All Three-State:**

Sales proportion- category wise



Foods category has highest item sales proportion, 68.61%, Household has second highest, approx. 22.1% and Hobbies has least, 9.32%.

- **Sales based on a product category and in every state:**



For CA, TX and WI - Food has highest sales proportion, hobbies have lowest

▪ Time Series of Total sales on product category among all years



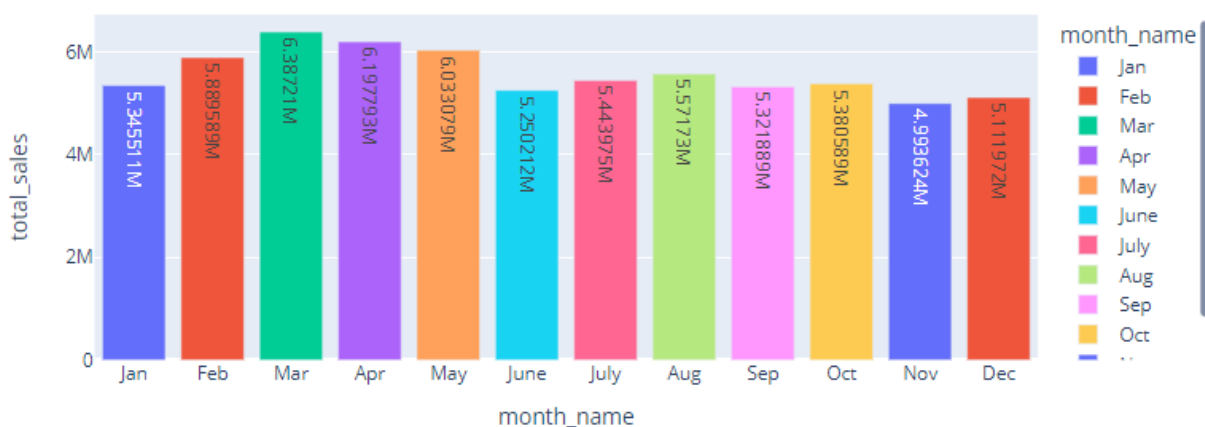
First thing we observe is the pattern, for each year it is showing some pattern in sales
 There are many days in a year where sales go to zero drastically, this could be due to some holiday when stores are closed for that day

Sales are increasing year by year

- For Foods and Household: sales increasing over the years
- For Hobbies: sales remain almost constant over the years

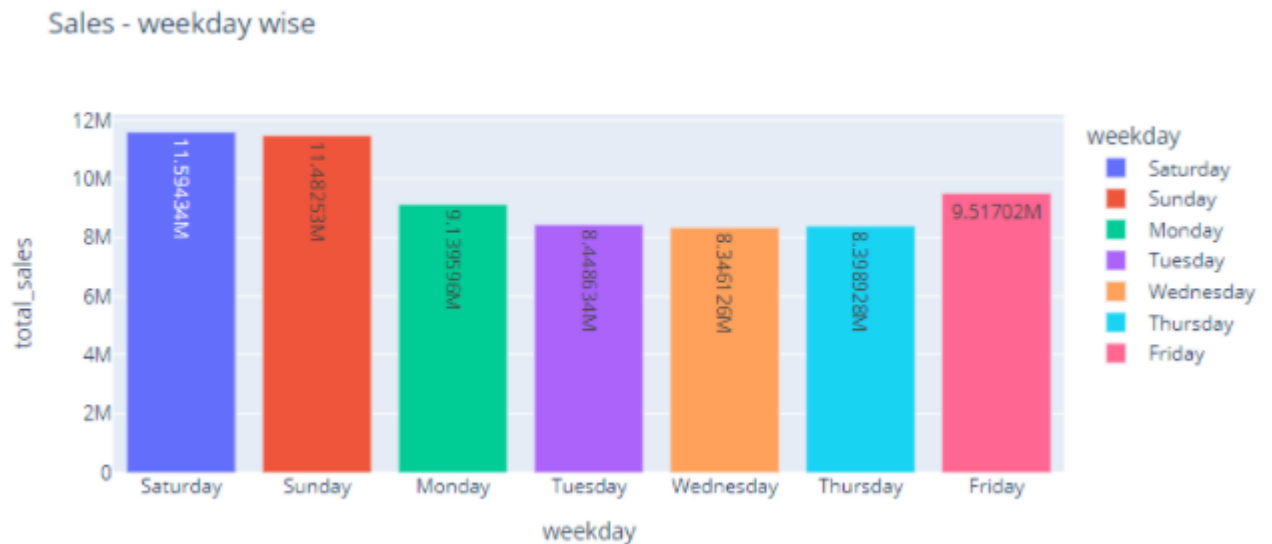
▪ Total Sales Based on Months among all years

total sales - month wise



Sales in the month of February, March, April and May is high as compare to other months among all years.

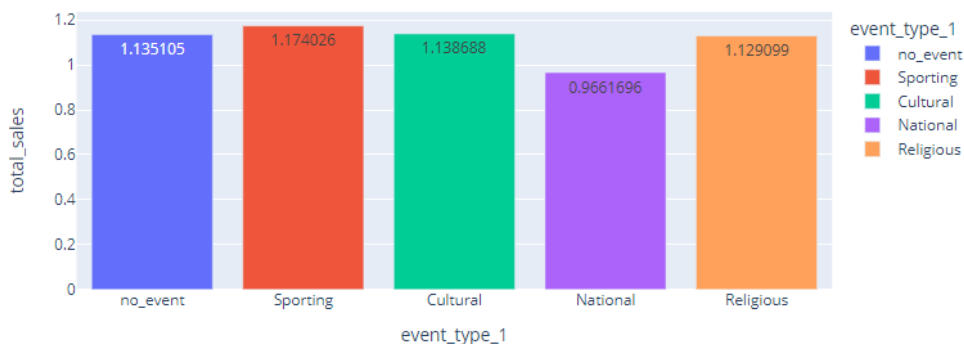
- **Total Sales Based on Weekdays among all years**



Sales are high on weekends including Friday and rest of the days sales are normal

- **Total Sales Based on event type and event name among all years**

Sales - event type 1 wise

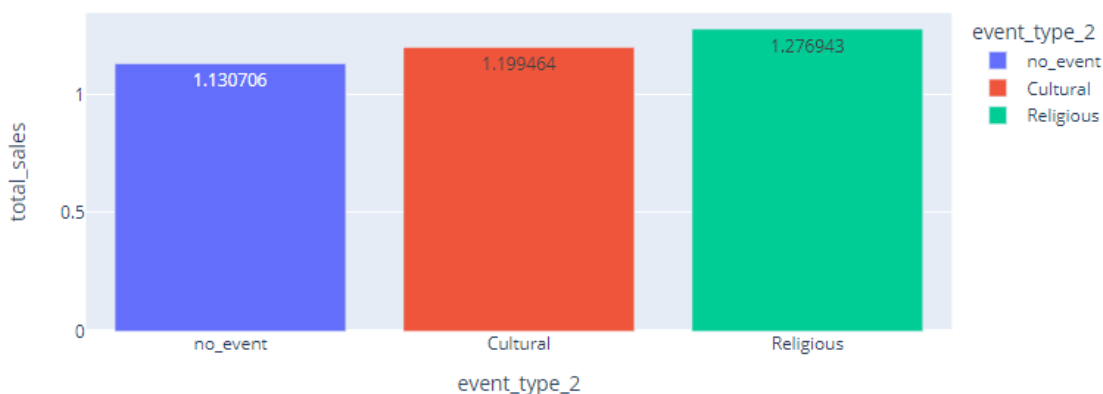


Sales - Event name 1 wise

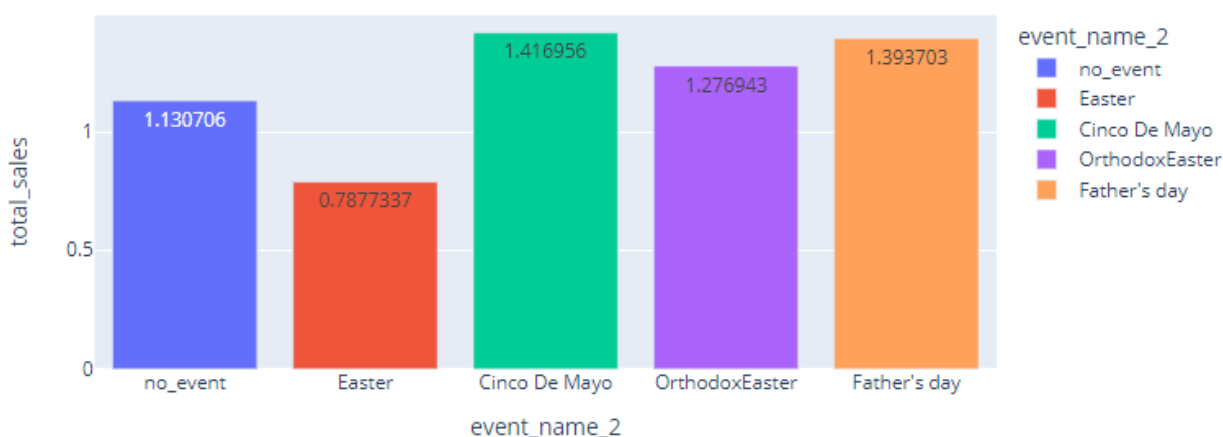


For event type 1 mean sales are high on events as compared to no events specially if there is an event related to sports. Also, for days having National event mean sales are lower comparatively.

Sales - event type 2 wise



Sales - event name 2 wise



Overall, we can say sales get impacted according to event type. Average sales are maximum when we have events like Religious and Cultural type.

Even on no event day, sales appear almost has event day so we can't get an exact conclusion that sales may occur more on event days.

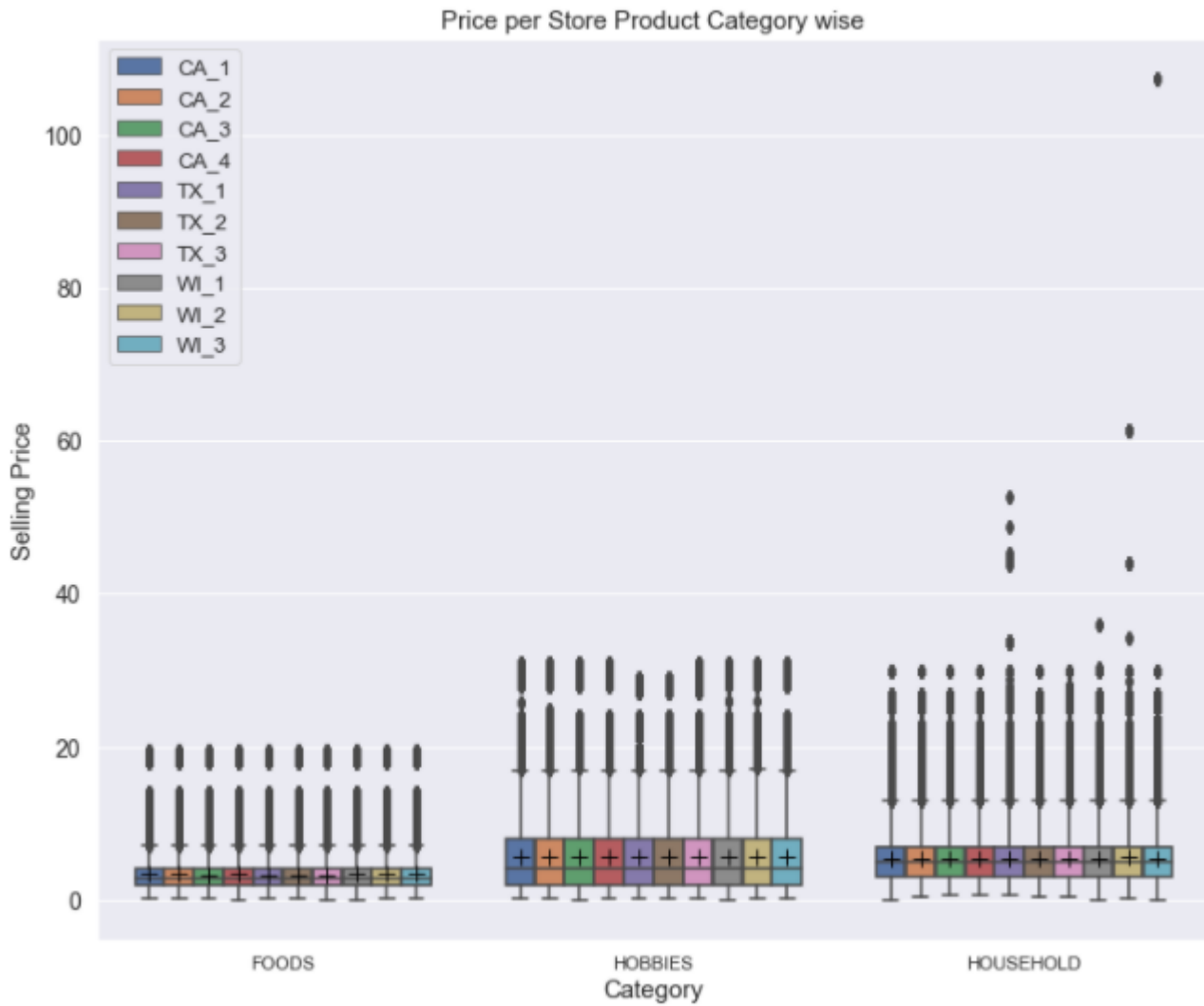
▪ Total Sales Based on SNAP days among all years

snap_CA		total_sales	snap_TX		total_sales	snap_WI		total_sales
0	0	19069726.0	0	0	12416557.0	0	0	11569147.0
1	1	10126991.0	1	1	6811848.0	1	1	6932904.0

Here 0 indicates non-Snap days and 1 indicates Snap days.

So, sales in all the three states during the Non-Snap days are higher than sales in Snap days.

- **Analysis based on Sell Price:**



For the Foods category, sell price is almost same for every store and lowest among all categories. For Hobbies category sell price is same for every store but higher than foods category and for household category sell price is highest among all categories and sometimes sell price goes very high (more than 100)

5. Data Preparation

In machine learning algorithm, data can't be used in its normal form as it is the as the way it is obtained, so the data needs to be devised before employing it in machine learning models

5.1 Data Downcasting

Downcasting means type-casting of the data and to reduce the amount of storage used by them. As pandas automatically create int32, int64, float32, or float64 columns for numeric ones, we can convert them into int8/int16 and float8/float16. Also, Pandas stores categorical columns as objects which take more storage as compare to category datatype, so convert object datatype to category datatype.

After downcasting it is observed the size of all dataframes approximately reduced to one-fourth of the size of the original dataframes and this also reduced chances of 'RAM crashed' error.

5.2 Data Melting and Merging

▪ Data Melting

To make analysis of data in table easier we can reshape the data into a more computer-friendly form using pandas in Python. First, we have used pandas.melt() function to univerts a Sales DataFrame from wide format to long format where 'd' represents days and 'quantity sold' represents demand of the product on that day. This would be necessary for us to pose our problem into exact regression problem.

Also, conversion of wide format to long format increases the size of the file drastically e.g., for sales dataframe previously memory consumption was 96.3 MB but after melting it increased to 1.4GB

```
%%time
# https://www.geeksforgeeks.org/python-pandas-melt/#:~:text=melt()%20function%20is%20useful,identifier%20columns%2C%20variable%2C
# https://pandas.pydata.org/docs/reference/api/pandas.melt.html

sales_melt = pd.melt(sales_df, id_vars =['id', 'item_id','dept_id','cat_id','store_id','state_id'],
                    var_name = 'd', value_name='quantity sold').dropna()
```

Before melt:

	id	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	...	d_1932	d_1933	d_1934	d_1935	d_1936
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	2	4	0	0	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	1	2	1	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	0	2	0	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	1	0	4	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	0	2	0
5	HOBBIES_1_006_CA_1_evaluation	HOBBIES_1_006	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	2	1	0	0	0
6	HOBBIES_1_007_CA_1_evaluation	HOBBIES_1_007	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	1	0	0	0

After melt:

	id	item_id	dept_id	cat_id	store_id	state_id	d	quantity sold
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
5	HOBBIES_1_006_CA_1_evaluation	HOBBIES_1_006	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
6	HOBBIES_1_007_CA_1_evaluation	HOBBIES_1_007	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0
7	HOBBIES_1_008_CA_1_evaluation	HOBBIES_1_008	HOBBIES_1	HOBBIES	CA_1	CA	d_1	12
8	HOBBIES_1_009_CA_1_evaluation	HOBBIES_1_009	HOBBIES_1	HOBBIES	CA_1	CA	d_1	2
9	HOBBIES_1_010_CA_1_evaluation	HOBBIES_1_010	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0

▪ Data Merging

We have given 3 files calendar.csv, sale_train_evaluation.csv, and sell_price.csv and we need to create a single Data frame by combining all 3 of them. We have merged it with other 2 dataframes using merge function.

```
%%time
# merging calendar dataframe with sales_data
ddf = dd.merge(sales_data, calendar_df, on = 'd', how = 'left')

# merging sell_price dataframe within our final_df
final_ddf = dd.merge(ddf, prices_df, on=['store_id','item_id','wm_yr_wk'], how='left')
```

6. First Cut Solution

6.1 Without Feature Engineering & Hyperparameter Tuning

We have decided to solve the problem without any feature engineering as well as without hyperparameter tuning. We have converted all the categorical features available into numerical form using Label Encoder which convert them into the machine-readable form.

StandardScaler transforms the data in such a manner that it has mean as 0 and standard deviation as 1. In short, it standardizes the data. Standardization is useful for data which has negative values and arranges the data in a standard normal distribution. Generally, it is more useful in classification than regression. But since we were just experimenting, so we used it here also to see if it improves our RMSE score.

Technique Info	Model	RMSE on Test Data	RMSE: After Standardization
Without FE & No Hyperparameter Tuning			
1. Using Date Based Features Only			
1.1	Simple Linear Regresion	3.5804	3.5803
1.2	LR with L1 Regularization	3.5848	3.5928
1.3	LR with L2 Regularization	3.5806	3.5804
1.4	Decision Tree	3.2821	3.2864
1.5	Light GBM	2.8360	2.8153

Without Hyperparameter tuning and any feature engineering the RMSE score is very high. Also, it was observed that there was no major improvement on RMSE score after standardizing the data so we can ignore in our current regression problem.

Out of all the models LightGBM performed well comparatively but the RMSE is quite high which will lead to wrong predictions .

So now our task would be to come up with some meaningful features using feature engineering techniques and optimize our models which can help us to improve our RMSE score.

7. Feature Engineering

7.1 Handling Missing values

Missing data is something that needs to be manipulated so that there remains no discrepancy in the data to be fed into the model. Here there were some missing values in calendar data where in event type and event name columns have more than 80% missing values, we have replaced the missing values with “no_event”.

Then we need Handle the missing values as it comes from sell price data, through mean imputation techniques. Instead of taking simple mean ,we going to take mean price of item with respect to its store because price may vary based on store.

7.2 Encoding Categorical data

Converting all the categorical features by replacing them with their cat codes. LabelEncoder is a class imported from the sklearn library to do this task.

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated.

We have encoded - 'id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'event_name_1', 'event_type_1', 'event_name_2', 'event_type_2'.

```
label_encode = LabelEncoder()

# List with all the categorical features
category = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id', 'event_name_1', 'event_type_1', 'event_name_2', 'event_type_2']
for cat in tqdm(category):
    final_df[cat+'_label'] = label_encode.fit_transform(final_df[cat])

100%|██████████| 10/10 [00:29<00:00, 2.95s/it]
```

7.3 Adding Time-Based features

While doing EDA we found that there is a trend that sales are high on weekend i.e., Saturday and Sunday, sales are high on Feb, March, April and May and on Christmas sales goes to Zero.

So, we have added three features if_weekend, if_month_seasonality and if_christmas which returns 1 if its True.

7.4 Basic Feature Engineering for Time Series Problem

- Lag Features: A critical step for the time-series forecasting is the right determination of the number of past observations (lags). In EDA I found that we can't find periodicity in months or in years, but we can find the periodicity in weeks. Here we have applied Lags on 'quantity sold' column. The maximum Lags taken is 56 days. For example,

Lag 1: This means how many products have been sold one day before

Lag 7: This means how many products have been sold seven days before.

Features Name: ['lag_1', 'lag_7', 'lag_14', 'lag_21', 'lag_28', 'lag_35', 'lag_42', 'lag_49', 'lag_56']

```
lags = [1,7,14,21,28,35,42,49,56]
for i in tqdm(lags):
    final_df_['lag_'+str(i)] = final_df.groupby(['id'])['quantity sold'].shift(i)
```

- **Rolling Window Statistics:** Rolling is a very useful operation for time series data. Rolling means creating a rolling window with a specified size and perform calculations on data in this window which of course rolls through data. Here we have computing Rolling-Mean on 'quantity sold' column. The maximum Window size taken is 42.

Features Name: ['rolling mean 7', 'rolling mean 14', 'rolling mean 28', 'rolling mean 35', 'rolling mean 42']

```
] : # https://pandas.pydata.org/docs/reference/api/pandas.Series.rolling.html
# https://stackoverflow.com/questions/13996302/python-rolling-functions-for-groupby-object
##https://www.geeksforgeeks.org/python-pandas-dataframe-transform/
|
window = [7,14,28,35,42]

for i in tqdm(window):
    func = lambda x: x.rolling(i).median()
    final_df_['rolling median_'+str(i)] = final_df.groupby(['id'])['quantity sold'].transform(func)
```

- **Expanding Window Statistics:** This is an advance version of rolling window. It takes all the past values into consideration.

```
func = lambda x: x.expanding().median()
final_df_['expanding median'] = final_df.groupby(['id'])['quantity sold'].transform(func)
```

7.5 Splitting the dataset into training and test dataset

As the rows of resultant data are quite large so only took rows which have “d_”(day number) is greater than 1400. It reduced the computation time and gave better results.

Time based splitting is done and took generated features as input variables and items sold on that date as output variable which belongs to the real number.

```
# Splitting data for independent variable (Predictors) based on days

X_train = final.loc[final['day'] <=1885]
X_CV = final.loc[(final['day']>1885) & (final['day']<1914)]
X_test = final.loc[final['day'] >=1914]

# Splitting data for dependent variable (Target) based on days

Y_train = X_train['quantity sold']
Y_CV = X_CV['quantity sold']
Y_test = X_test['quantity sold']
```

8. ALGORITHMS USED

8.1 Linear Regression

Linear Regression is the most commonly and widely used algorithm Machine Learning algorithm. It is used for establishing a linear relation between the target or dependent variable and the response or independent variables. The linear regression model is based upon the following equation:

$$w^*, w_0^* = \arg - \min_{w, w_0} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

where, y_i is actual and \hat{y}_i is predicted value. And what we are doing is minimizing loss.

Also, $\hat{y}_i = wx_i + b$

So, the equation becomes

$$L = \sum_{i=1}^n (y_i - (wx_i + b))^2$$

L = is the loss function also called ordinal least square

b = is the constant term for intercept

x_i = are the independent variables

and w = are their respective coefficient

The main aim of this algorithm is to find the best fit line to the target variable and the independent variables of the data. It is achieved by finding the most optimal values for all w. With best fit it is meant that the predicted value should be very close to the actual values and have minimum error.

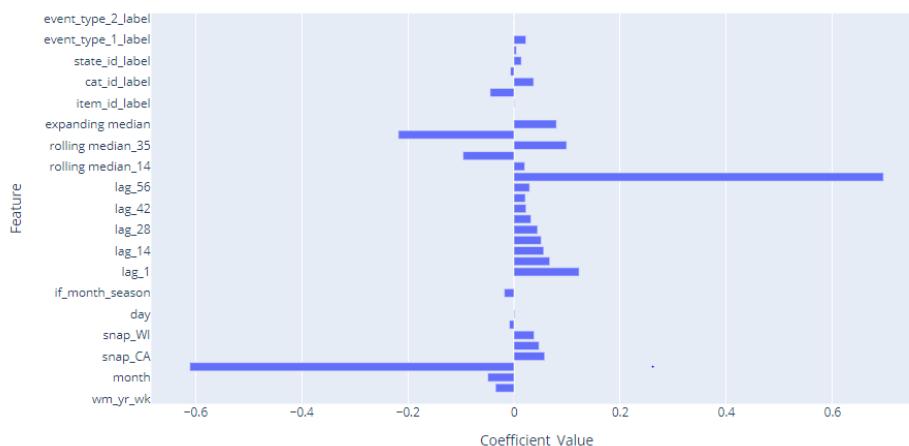
Error is the distance between the data points to the fitted regression line and generally can be calculated by using the following equation, Error = $y - \hat{y}_i$.

Code Snippet:

```
%%time  
  
model1 = LinearRegression(n_jobs = -1)  
model1.fit(X_train, Y_train)
```

Wall time: 29.8 s

RMSE = 1.8989



Linear Regression with L1 and L2 regularization

Regularisation is a process of introducing additional information in order to prevent overfitting. A linear regression model that implements L1 norm for regularisation is called lasso regression

and one that implements (squared) L2 norm for regularisation is called ridge regression. To implement these two, note that the linear regression model stays the same.

- Linear Regression with L1 regularization

L1 or lasso regularizer is a regularization technique that requires us to minimize the sum of absolute values between features and target variable. The word "absolute" here means in this example we only care about the Largest value between feature and target, and other smaller values within the same feature will be ignored(or removed) when training models using this model.

L1 norm: $||w||_1 = |w|_1 + |w|_2 + |w|_3 + \dots + |w|_n$

Main Mathematical equation:

$$L = \sum_{i=1}^n (y_i - (wx_i + b))^2 + \lambda |w_i|$$

Where,

λ is called the regularization parameter and $\lambda > 0$ is manually tuned

w_i = weights of each feature-target pair

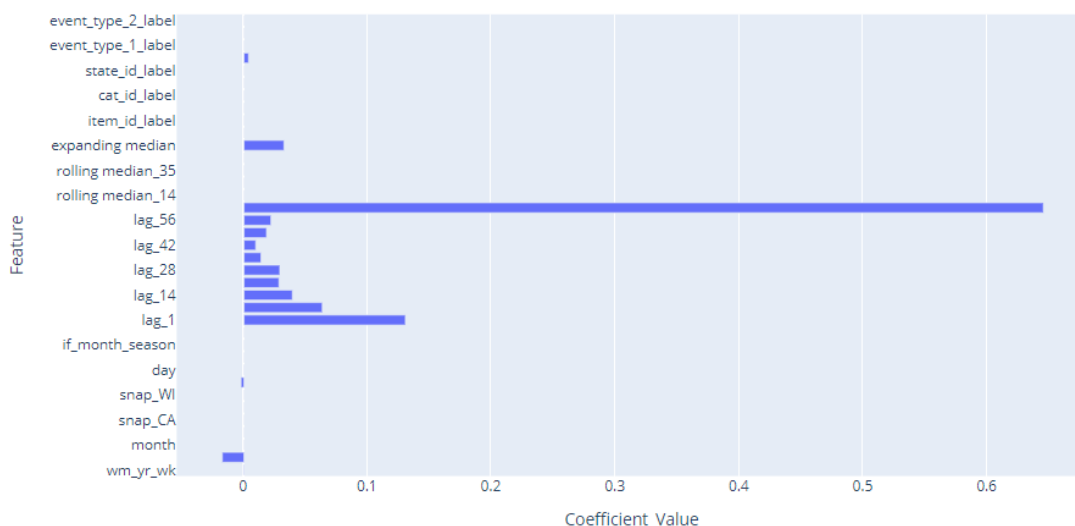
L1 Penalization works well when dealing with high dimensional data because its penalty parameter λ allows us to ignore or remove irrelevant features.

Code Snippet:

```
%%time  
  
model2 = linear_model.Lasso(alpha = 0.1)  
model2.fit(X_train,Y_train)
```

Wall time: 52.3 s

RMSE = 1.9088



- Linear Regression with L2 regularization

Using L2 or Ridge Regression for linear regression and Logistic Regression , Ridge Regressor adds a term called regularizer weight between input variables X_i and output y_i which represents the sum of squares of weights in order to prevent coefficients from being zero.

$$\text{L2 norm: } ||w||_2 = \sqrt{(w_1^2 + w_2^2 + w_3^2 + \dots + w_n^2)}$$

Main Mathematical equation:

$$L = \sum_{i=1}^n (y_i - (wx_i + b))^2 + \lambda |w_i|^2$$

Ridge Regression allows us avoid overfitting by shrinking large coefficients towards zero and leaving small ones unchanged

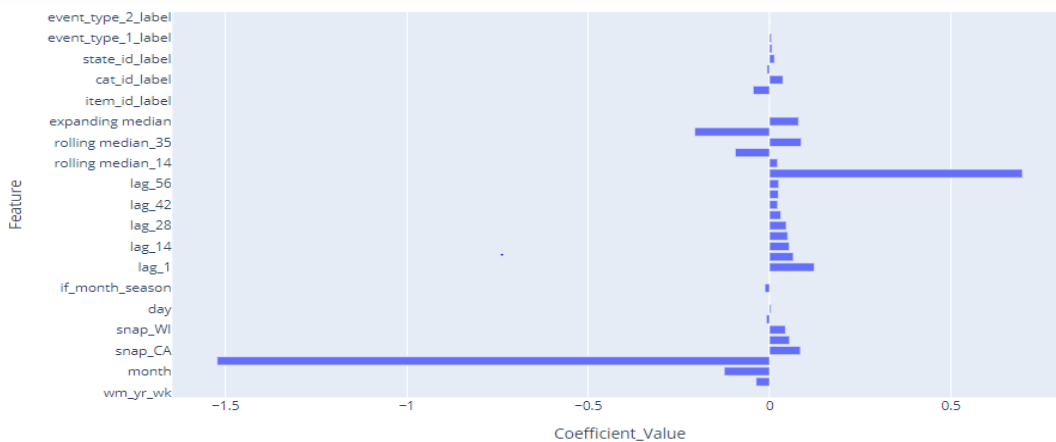
Code Snippet:

```
%%time

model3 = linear_model.Ridge(alpha=0.1)
model3.fit(X_train,Y_train)
```

Wall time: 8.46 s

RMSE = 1.8990



8.2 Decision Tree Regressor

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. It works Decision trees work by basically dividing the features space using axes parallel separation boundaries just like KD trees.

Code Snippet:

```
%%time

for _ in tqdm(range(10)):
    max_depth = np.random.randint(1,200)
    min_samples_split = np.random.randint(10,500)
    max_leaf_nodes = np.random.randint(20,400)
    dtr = DecisionTreeRegressor(max_depth = max_depth, min_samples_split=min_samples_split, max_leaf_nodes=max_leaf_nodes)
    dtr.fit(X_train,Y_train)
    y_pred_cv = dtr.predict(X_cv)
    print(f"max_depth: {max_depth}, min_samples_split:{min_samples_split}, max_leaf_nodes: {max_leaf_nodes},RMSE:{RMSE_calc(y_pre
    print("~"*100)
```

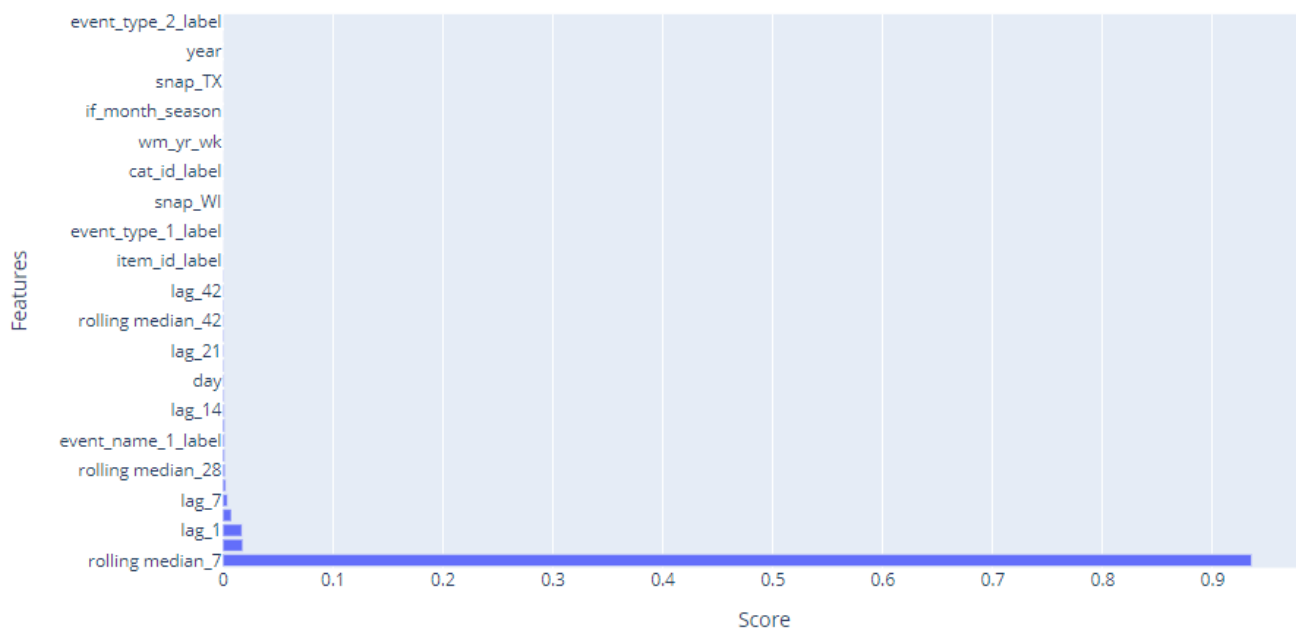

After hyperparameter tuning, best parameter we got are: max_depth =13 ,min_samples_split =277 ,max_leaf_nodes =386.

```
%%time
```

```
model14= DecisionTreeRegressor(max_depth=13 ,min_samples_split=277 ,max_leaf_nodes=386)  
model14.fit(X_train,Y_train)
```

Wall time: 2min 33s

RMSE = 1.8931



8.3 Random Forest Regressor

The word 'Random' in Random Forest means random sampling with replacement. The word 'Forest' means a group of trees. So basically, Random forest is an ensemble technique which uses Decision Tree as base learners, adding bagging on top of the base learners and adding column sampling (Feature Bagging) on top of it.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Code Snippet:

```
%%time
```

```
for _ in tqdm(range(10)):  
    max_depth= np.random.randint(1,10)  
    min_samples_leaf= np.random.randint(2,8)  
    n_estimators= np.random.randint(50,100)  
    rf = RandomForestRegressor(max_depth=max_depth, min_samples_leaf=min_samples_leaf, n_estimators=n_estimators, n_jobs=-1)  
    rf.fit(X_train,Y_train)  
    y_pred_cv = rf.predict(X_cv)  
    print(f"max_depth:{max_depth}, min_samples_leaf:{min_samples_leaf}, n_estimators: {n_estimators},RMSE: {RMSE_calc(y_pred_cv,\n    print("~"*100)
```

After hyperparameter tuning best parameter we got are: max_depth= 9, min_samples_leaf=6, n_estimators=84.

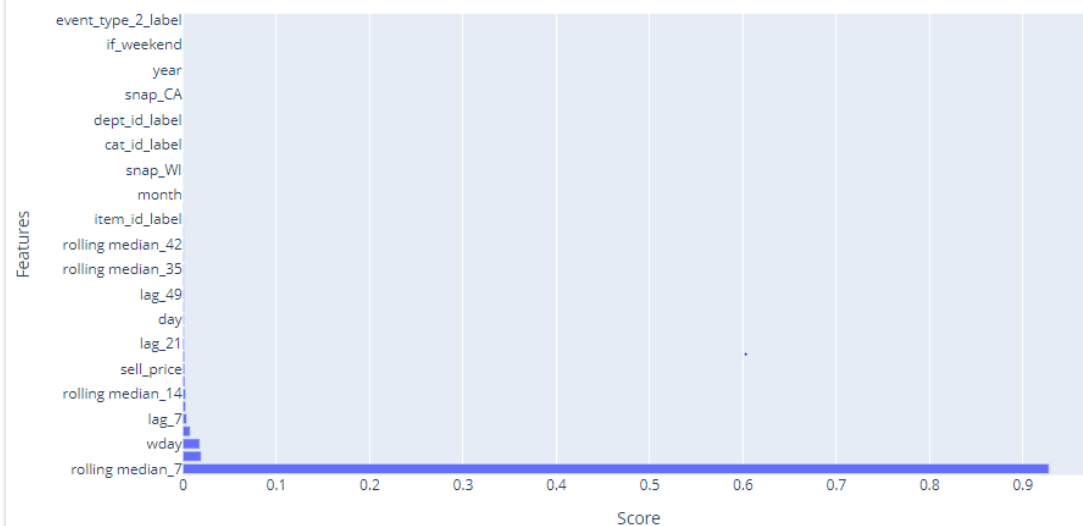
```
%%time
```

```
model15 = RandomForestRegressor(max_depth= 9, min_samples_leaf=6, n_estimators=84, n_jobs=-1)  
model15.fit(X_train,Y_train)
```

Wall time: 1h 7min 55s

Parser : 245 ms

RMSE = 1.8738



8.4 LightGBM Regressor

Gradient Boosting refers to a methodology in machine learning where an ensemble of weak learners is used to improve the model performance in terms of efficiency, accuracy, and interpretability. These learners are defined as having better performance than random chance. Such models are typically decision trees and their outputs are combined for better overall results.

Some of the most popular boosting algorithms widely used in enterprises and data science competitions are XGBoost (eXtreme Gradient Boosting), LightGBM (Light Gradient Boosting Machine), and CatBoost (Category Boost). But due to lack of computational resources we have not trained on XGboost and CatBoost takes a lot of time and memory to train.

Why LightGBM model?

Since LightGBM is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.

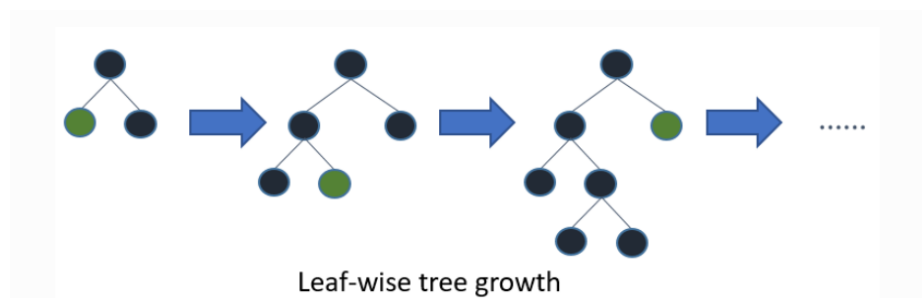
So, when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it can handle large size of data and takes lower memory to run hence the word 'Light' in prefix is used.

Level-wise tree growth in XGBOOST



[Source](#)

Leaf wise tree growth in Light GBM



[Source](#)

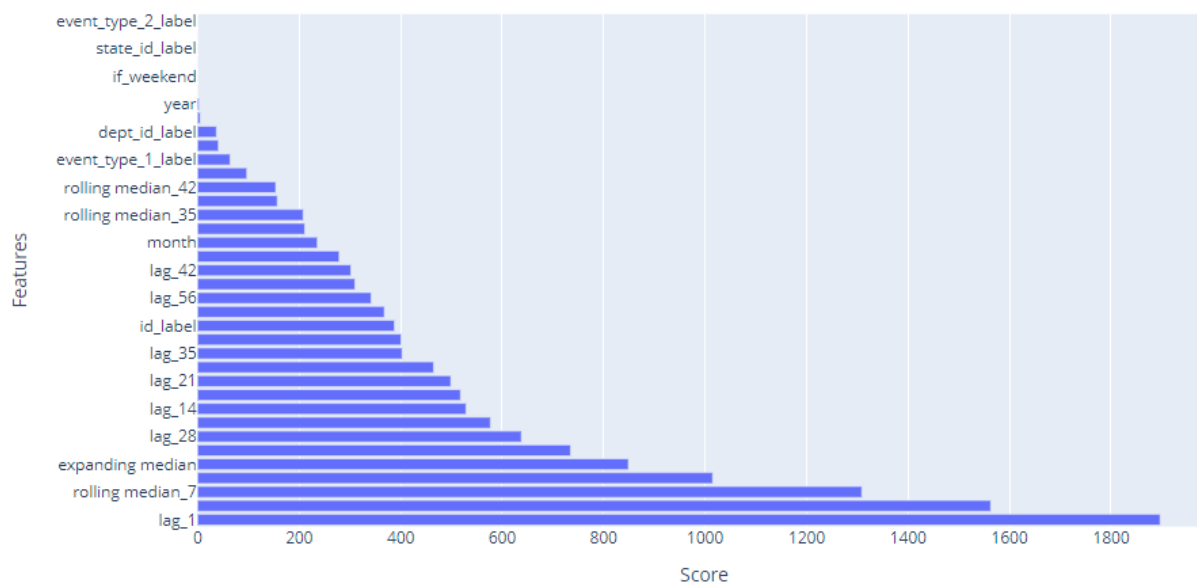
Code Snippet:

```
%time
for _ in tqdm(range(10)):
    learning_rate = np.round(np.random.uniform(0.001,0.05),4)
    max_depth = np.random.randint(5,200)
    num_leaves = np.random.randint(10,200)
    lgbm = LGBMRegressor(learning_rate=learning_rate, max_depth=max_depth, num_leaves = num_leaves,
                        n_jobs=-1,n_estimators=100)
    lgbm.fit(X_train,Y_train)
    y_pred_cv = lgbm.predict(X_CV)
    print(f"learning_rate:{learning_rate}, max_depth:{max_depth}, num_leaves:{num_leaves}, RMSE:{RMSE_calc(y_pred_cv, Y_CV)}")
    print("~"*100)
```

After hyperparameter tuning best parameter we got are: learning_rate= 0.0323, max_depth= 82, num_leaves=147

```
%%time
model6 = LGBMRegressor(learning_rate= 0.0323, max_depth= 82, num_leaves=147, n_jobs=-1, n_estimators=100)
model6.fit(X_train,Y_train)
```

RMSE: 1.8504



9. Conclusion

9.1 Models Comparison

Below is the comparison of all models we have used for this problem:

S No	Model	RMSE: on Validation Data	RMSE: on Test Data
1.	Simple Linear Regression	1.8704888	1.8989383
2.	Linear Regression with L1 Regularization	1.8825738	1.9088581
3.	Linear Regression with L2 Regularization	1.8709787	1.8990191
4.	Decision Tree	1.8357908	1.8931382
5.	Random Forest	1.8357908	1.8738018
6.	LGBM	1.8111131	1.8504773

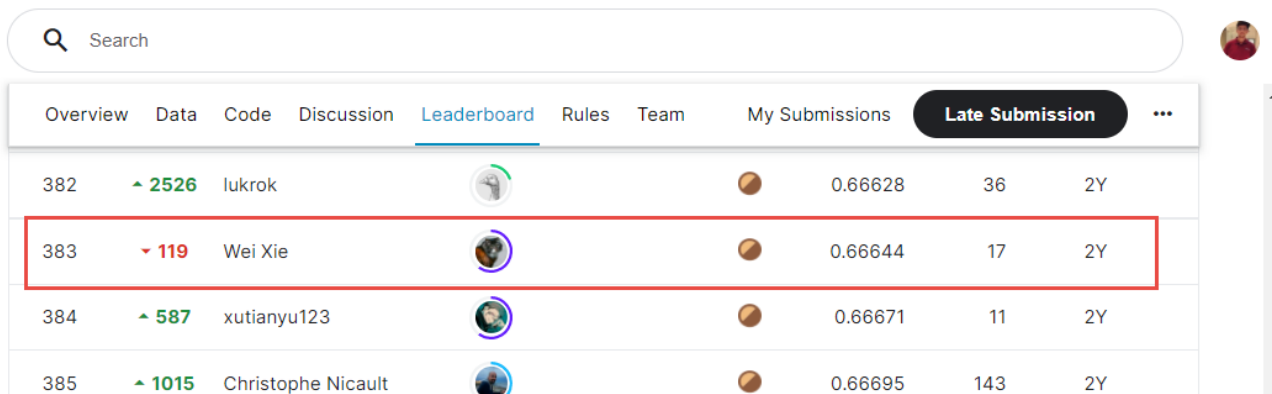
9.2 Kaggle Score









Kaggle public & private scores for each model is as follows:-

Submission and Description	Private Score	Public Score	Use for Final Score
lgbm_submit.csv 13 minutes ago by Akansh Sharma Model6: LGBM Regressor	0.66645	0.78007	<input type="checkbox"/>
rf_submit.csv 21 minutes ago by Akansh Sharma Model5: Random Forest Regressor	0.68845	0.79413	<input type="checkbox"/>
dtr_submit.csv 16 hours ago by Akansh Sharma Model4: Decision Tree Regressor	0.68992	0.79021	<input type="checkbox"/>
lr2_submit.csv 16 hours ago by Akansh Sharma Model3: Linear Regression with L2 Regularization	0.79947	0.86411	<input type="checkbox"/>
lr1_submit.csv 16 hours ago by Akansh Sharma Model2: Linear Regression with L1 Regularization	0.81411	0.91611	<input type="checkbox"/>
lr_submit.csv 17 hours ago by Akansh Sharma Model1 : Simple Linear Regression	0.80660	0.85359	<input type="checkbox"/>

From the above private leader board score we observe that of all the models light GBM model performs well as it predicted sales with lower score(WRMSSE) = **0.66645 (private score)** which holds the **383 rank out of 5,558 participants** and stands in **top 7% on private leader board**.

Kaggle Competition Leaderboard:

A screenshot of the Kaggle Competition Leaderboard interface. At the top, there is a search bar with a magnifying glass icon and the word 'Search'. Below the search bar is a navigation bar with tabs: 'Overview', 'Data', 'Code', 'Discussion', 'Leaderboard' (which is highlighted with a blue underline), 'Rules', 'Team', 'My Submissions', and a 'Late Submission' button. To the right of the navigation bar is a user profile picture. The main content is a table with five rows of data. The second row, for user 'Wei Xie', is highlighted with a red border. The table columns are: Rank, Change, Username, Profile Picture, Medal, Score, Votes, and Time.

Rank	Change	Username	Profile Picture	Medal	Score	Votes	Time
382	▲ 2526	lukrok			0.66628	36	2Y
383	▼ 119	Wei Xie			0.66644	17	2Y
384	▲ 587	xutianyu123			0.66671	11	2Y
385	▲ 1015	Christophe Nicault			0.66695	143	2Y

9.3 Final Conclusion

Sales forecasting plays a vital role in the business sector in every field. With the help of the sales forecasts, sales revenue analysis will help to get the details needed to estimate both the revenue and the income.

Different types of Machine Learning techniques such as Support Vector Regression, Gradient Boosting Regression, Simple Linear Regression, and Random Forest Regression have been evaluated on Walmart Sales Data to find the critical factors that influence sales to provide a solution for forecasting sales.

After measuring the performance metrics RMSE i.e., Root Mean Squared Error, the Light GBM is found to be the appropriate algorithm according to the provided data and thus fulfilling the aim of this thesis.

9.4 Future Work

We can consider following points to get better results:

1. Since due to lack of computational resources, we have considered sales from day 1400 for training. So, by taking whole data there could be chances to get good results.
2. Adding new features to the model can improve our model efficiency.
3. Using deep learning techniques like LSTM (Long Short-Term Memory) can improve the RMSE score and better forecasting result.

9. References

- a. <https://www.appliedroots.com/>
- b. <https://mofo.unic.ac.cy/m5-competition/>
- c. <https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/138881>
- d. <https://machinelearningmastery.com/basic-feature-engineering-time-series-data-python/>
- e. <https://www.kaggle.com/tarunpaparaju/m5-competition-eda-models>
- f. <https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/163216>
- g. <https://www.kaggle.com/c/m5-forecasting-accuracy/discussion/163684>
- h. <https://medium.com/thecyphy/m5-forecasting-accuracy-af6c45fb7d58>
- i. <https://www.analyticsvidhya.com/blog/2019/12/6-powerful-feature-engineering-techniques-time-series/>
- j. <https://medium.com/analytics-vidhya/m5-forecasting-accuracy-3668d6103130>
- k. <https://dipanshurana.medium.com/m5-forecasting-accuracy-1b5a10218fcf>
- l. <https://medium.com/@shantanuekhande19/m5-forecasting-accuracy-73343a873685>
- m. <https://medium.com/thecyphy/m5-forecasting-accuracy-af6c45fb7d58>

