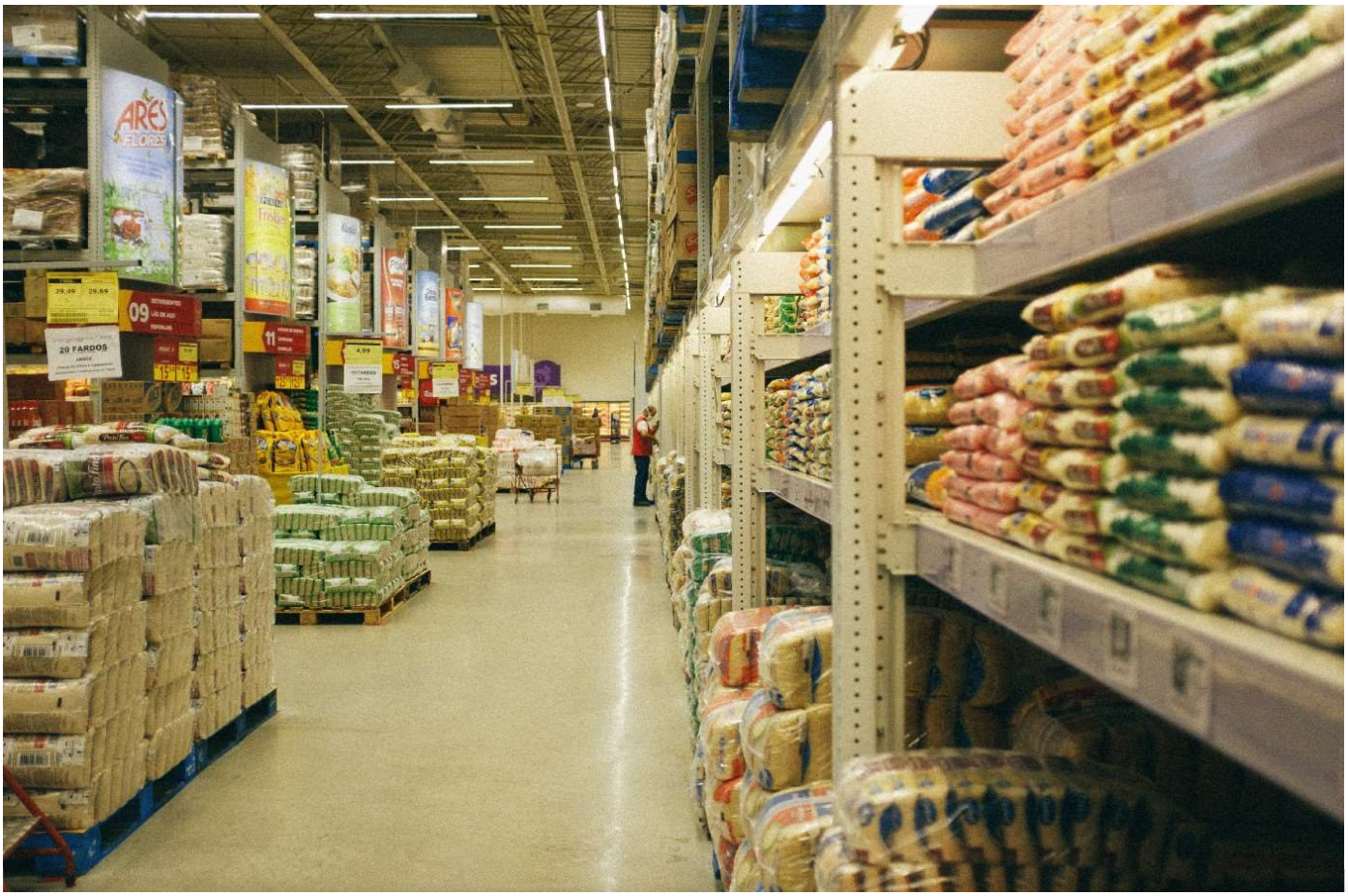
Published in Analytics Vidhya

Graham Fernando X Follow

Feb 2, 2021 · 9 min read · Listen

Case Study on M5 Forecasting - Accuracy | Kaggle Competition



2. *Business Problem*
3. *Why ML is required for solving this problem*
4. *Source of my data*
5. *Dataset*
6. *Hypothesis*
7. *Observations from EDA*
8. *Model Approach*
9. *Conclusion*
10. *Future Work*

I - Introduction:

This blog is about the M5 Forecasting a Time Series Problem approached using Python on Google Colab.

II - Business Problem:

Sales prediction is an important part of modern business intelligence. It can be an complex problem, especially in the case of lack of data missing and the presence of outliers. Sales can be considered as a Time Series. At present there are multiple Time Series models like ARIMA, SARIMA,SARIMAX, etc. Different ensemble based methods are considered for time series on sales/demand prediction and combining all the forecast produced by different algorithms , it's possible to improve the forecasting accuracy.

The main objective of this case study is to estimate precisely the number of product unit sales forecast, in the USA-based Walmart stores in various locations for two 28- day time periods.

The data, covers stores in three US States (California, Texas, and Wisconsin) and includes

Misleading business forecasts on product sales could potentially cause opportunity and revenue loss, for instance, if the analyst team at Walmart predicts that it would be sunny in most of the areas for the next few weeks and has stocked only less umbrellas on stores, but the prediction turned out to be wrong and there was a heavy downpour in many areas so there was a high demand of umbrellas more than what Walmart predicted and stocked on the stores. This would lead to revenue loss. Therefore, it is important to have an accurate machine learning models to perform prediction on producing business insights.

III - Why ML is required for solving this problem?

The problem is a time series model and it can be approached as a supervised machine learning model(regression) which can give us a good result, as the machine learning algorithm can find the pattern in the time series data/regression. The reason a time series analysis if treated as a regression problem gives good result is because the main assumptions of regression methods is that the patterns in the past data will be repeated in future. Using this logic in combination with the comprehension of the materials, we can deliver a complete business proposal to the company to construct refinement and thus reduce the loss.

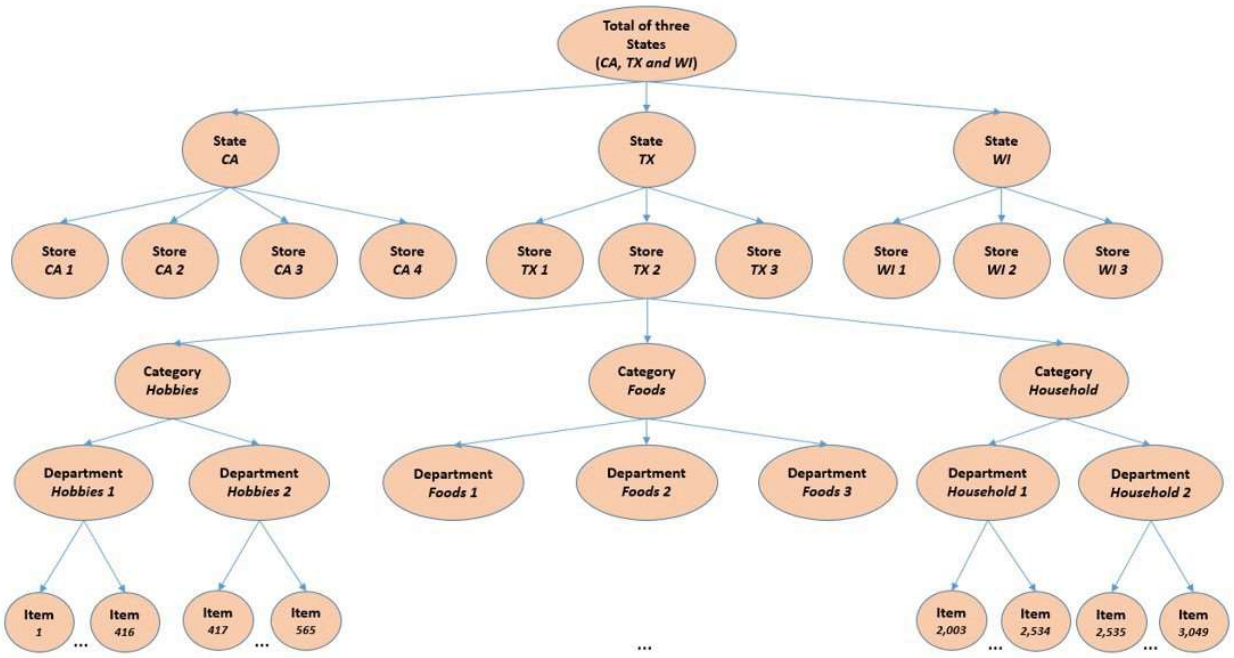
IV - Source of My data:

This is a Kaggle competition, you check the competition details by clicking here. The competition was held by University of Nicosia.

V - Dataset:

There are 3 csv files available:

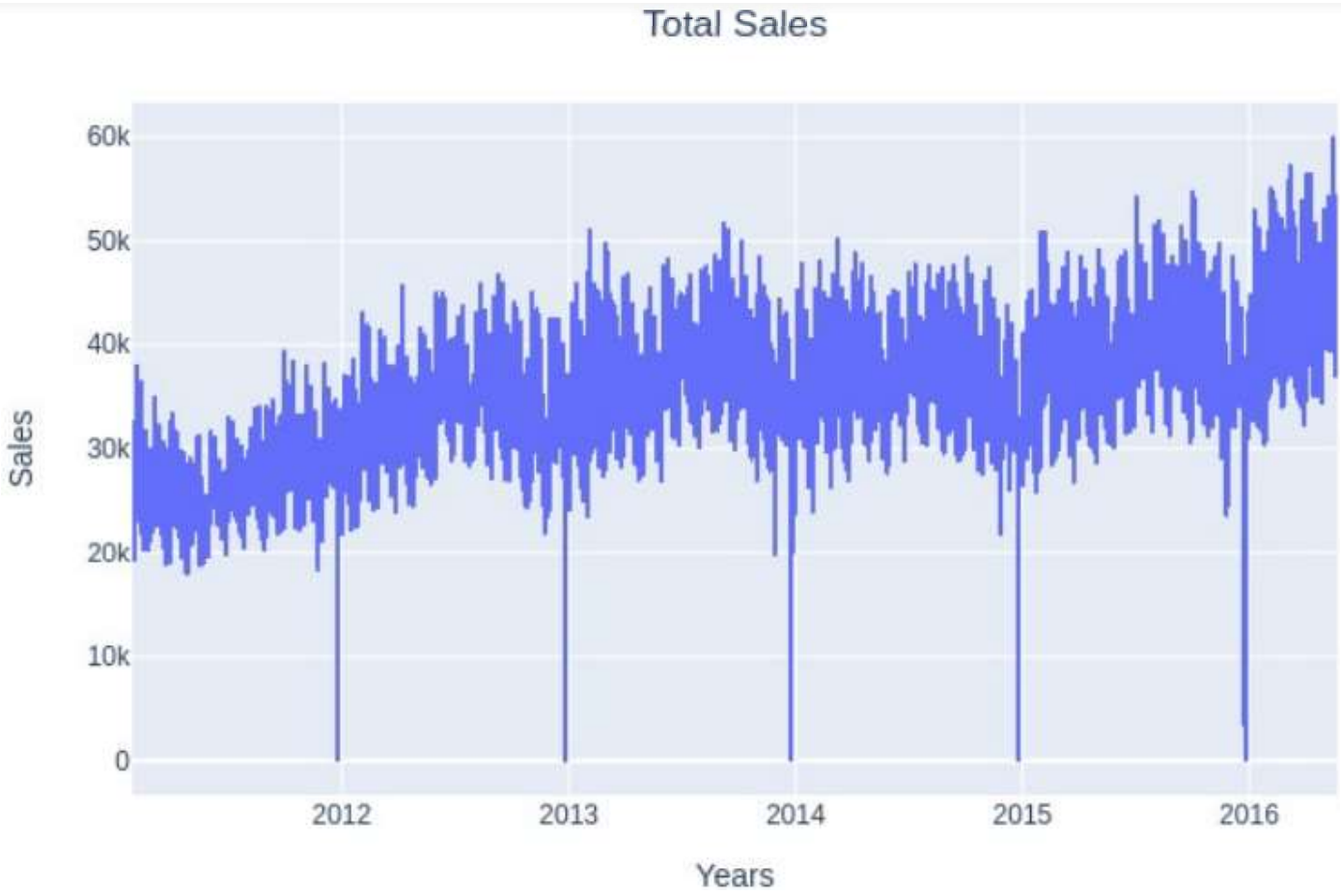
1. **Calendar.csv:** It contains the information about the dates on which the products are sold and the events and programs that held on that day.
2. **Sales_train_evaluation.csv:** It contains the historical daily unit sales of each product on each store from day 1 to day 1941.
3. **Sell_prices.csv:** It contains information about the price of the products on each week

An overview on how the data is provided

We have data from 3 states with 10 stores on total, with 7 departments under 3 types of categories.

VII - Observations from EDA:

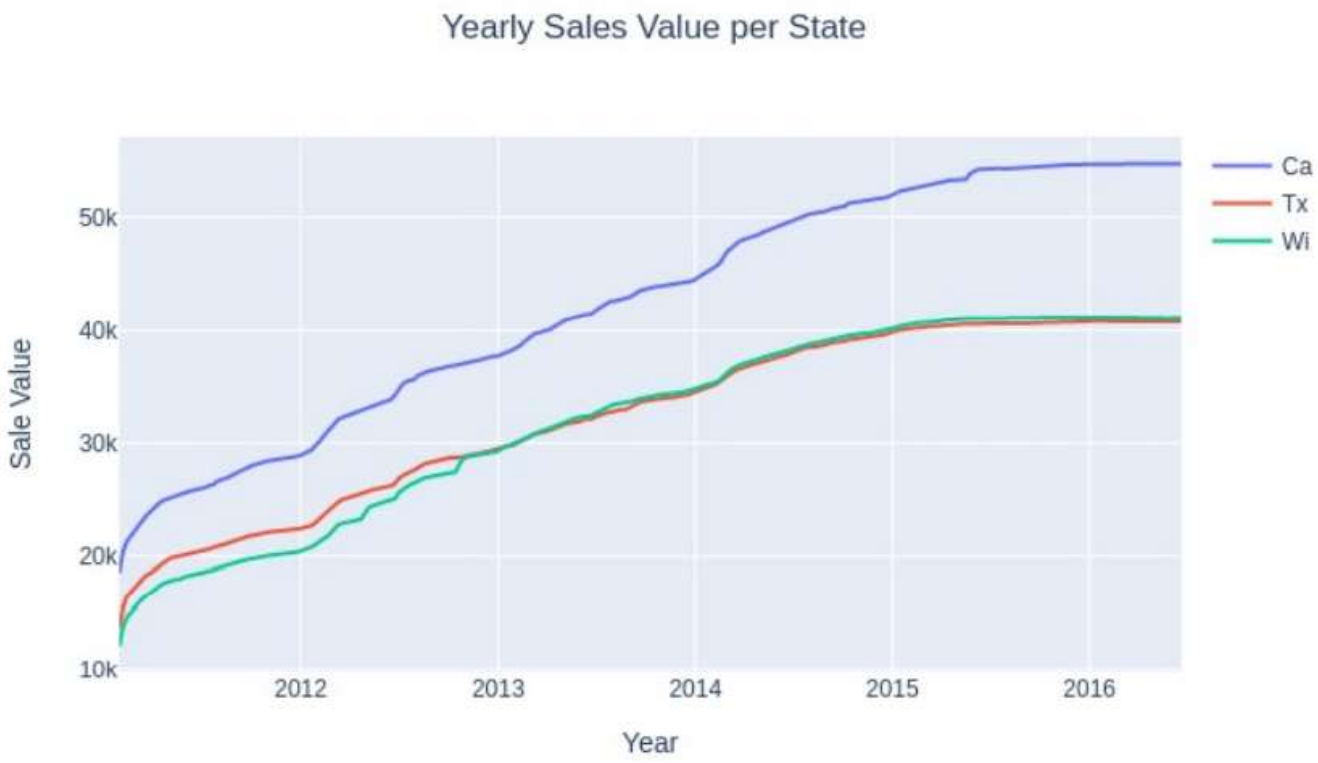


We can see that the data is following a seasonal trend. Every Year on Christmas the sales greatly fall less than 50. The demand is growing every year, not exponentially but smoothly.

The sales on Wisconsin and Texas are almost overlapping each other from 2013. Sales on California tops the chart, this could be because of the increased population on comparison to the others. Although the population of Wisconsin is lot less than Texas and Texas has slightly more GDP than Wisconsin, the difference between these two states is marginal.

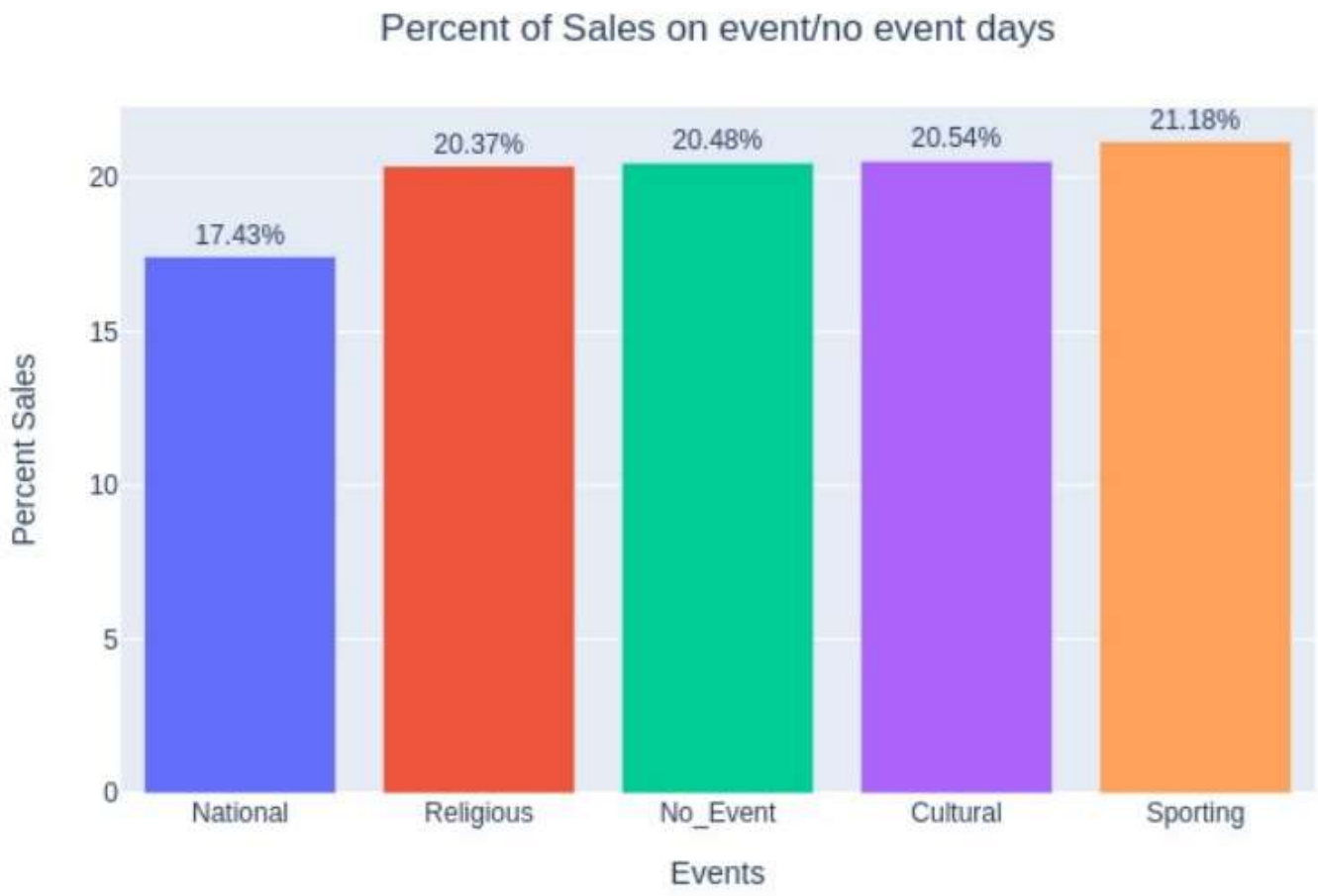
We previously observed that California had the most sales than Texas and Wisconsin, but it seems Store 3 in California ranks first for most sales also Store in California ranks last for the least number of sales. The sales on Store 1 in California differs from sales on other stores in all States by a great margin.

The total sales cost has increased every year till 2015. After 2015, the sales has almost been constant with no ups or downs.

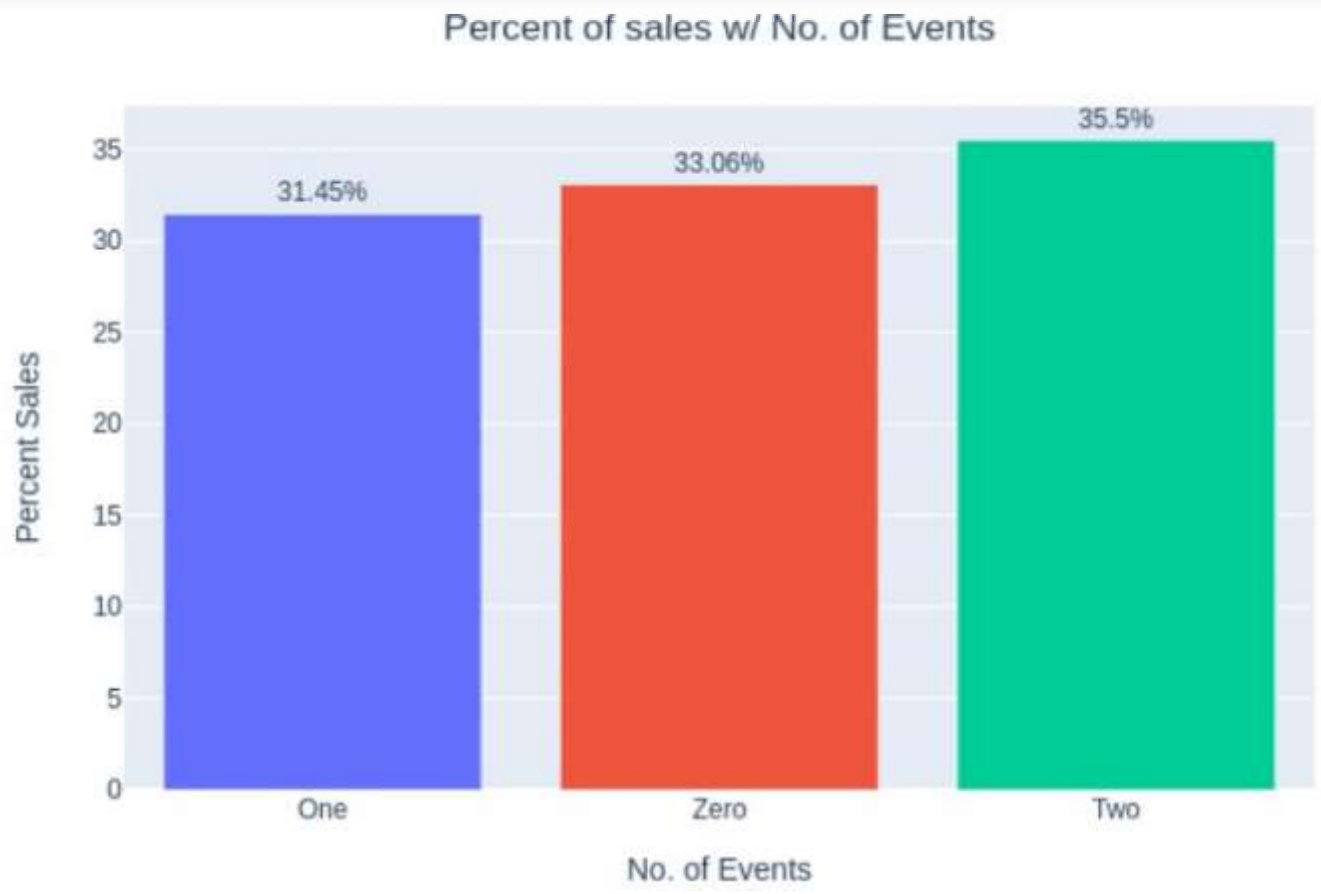


California tops the sales every year, whereas sales in Texas and Wisconsin are almost the same. We can understand that same type of prodcts are being sold almost equally across all the states as the total sale price graph is sames as the number products sold on each state.

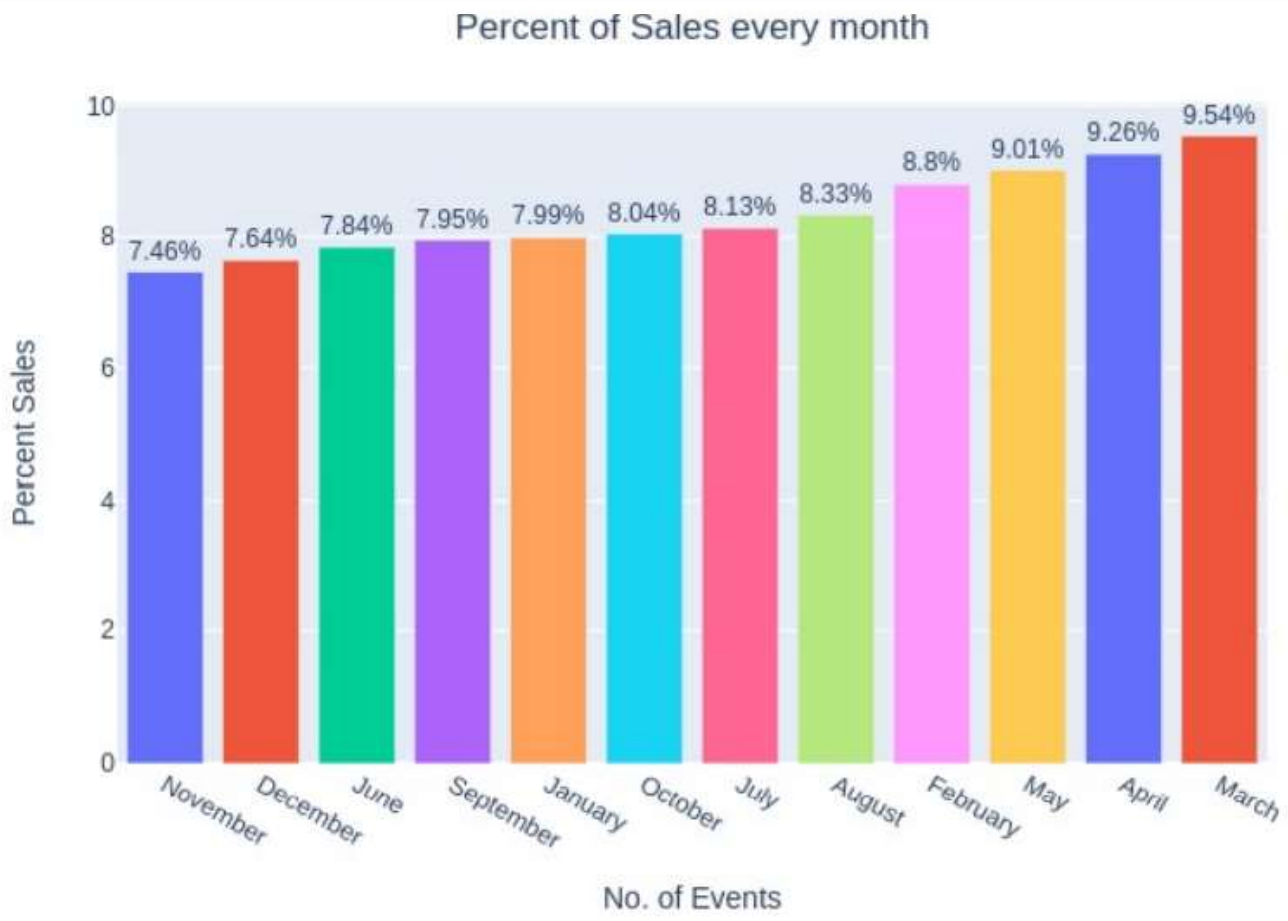
There's a huge number of sales happening during the weekends. The sales rise and fall before and after weekends from Friday to Monday.



The sales happened during sporting event times are slightly more than sales happened around no events and other events. The sales happened on National event days are little lower, but we can conclude it's the cause of low sales.



When there were 2 events on a day the sales happened are little high than days when there were no events. But on the contrary, when there were 1 event, the sales happened were little less than days when there were no events.



Sales are maximum during the month of March and least during the month of November. This could be because people after November tend to go on a trip to different places to enjoy their holidays and this happens the same during Summer.

VIII - Model Approach:

After reading all the files required for the problem, we are creating new features required for the prediction, for days from 1942 till 1969. Then we are creating the time series problem to machine learning problem and merging all the csv files together and filling the nan values on sell_price with the average sell_price of the product. Then we are dropping the features which are not required and converting all the categorical features by replacing them with their cat codes.

Then we are adding only lag shift features as adding more features causing our model to overfit.

```
lags = [28,30,35,42,49,56,63,70]
for lag in tqdm(lags):
    data["lag_" + str(lag)] = data.groupby("id")["demand"].shift(lag).astype(np.float16)
```

gistfile1.txt hosted with ❤ by GitHub view raw

Then we are picking up only the data after 1000 days, for making the processing easier and to avoid any memory errors. We are then splitting up our data into Train, Test and Val, were Train(X_train) has data from days between 1000 and 1913, validation(X_val) has data of days between 1914 and 1941 and finally test(X_test) is the data we need to predict for which is of days after 1941.

The first model I tried was the LightGBM and it turned out to be the best model.

```
# We are training our model with multiple parameter values to get the
# best parameter to be used for the model.

# We are randomly creating values to chose the best model out of it.
for i in range(15):
    lr = np.round(np.random.rand()/10,3)
    num_leaves = np.random.randint(30,150)
    min_data_in_leaf = np.random.randint(50,150)

    lgb = LGBMRegressor( learning_rate=lr ,
                         num_leaves=num_leaves ,
                         min_data_in_leaf=min_data_in_leaf)

    lgb.fit(X_train, y_train)

    # After training the model, we are trying to predict the model on X_val
    # to check it's accuracy
    y_pred = lgb.predict(X_val)
```

```
print( * 80)
```

gistfile1.txt hosted with ❤ by GitHub view raw

For the above code, our output will be like the below

```
For learning rate 0.002, num_leaves 88 and min_data_in_leaf 101 the RMSE is 10.832683887724896
********************************************************************************
For learning rate 0.098, num_leaves 75 and min_data_in_leaf 137 the RMSE is 4.968174905835645
********************************************************************************
For learning rate 0.002, num_leaves 141 and min_data_in_leaf 112 the RMSE is 10.79968686064402
********************************************************************************
For learning rate 0.046, num_leaves 48 and min_data_in_leaf 59 the RMSE is 5.087707910063227
********************************************************************************
For learning rate 0.003, num_leaves 127 and min_data_in_leaf 121 the RMSE is 9.869213241625621
********************************************************************************
For learning rate 0.048, num_leaves 83 and min_data_in_leaf 60 the RMSE is 5.048882772795383
********************************************************************************
For learning rate 0.018, num_leaves 122 and min_data_in_leaf 99 the RMSE is 5.45109618186815
********************************************************************************
For learning rate 0.092, num_leaves 94 and min_data_in_leaf 102 the RMSE is 4.964911198706563
********************************************************************************
```

We'll be getting 15 outputs since the loop is running 15 times. From the MSE values obtained, I chose the parameter value which has the least MSE value, and retrained the model. This was done the same for the CatBoostRegressor, were on the place of LGBMRegressor, I replaced it with CatBoostRegressor, found the best parameter by the least metric value it gave.

But on the case of XGBoost Regressor, I made small changes like the tree method to 'hist' and grow_policy as 'lossguide'.

```
1  xgb = XGBRegressor(objective='reg:squarederror',tree_method = 'hist',
2                     learning_rate = lr,min_child_weight = min_child_weight,
3                     subsample = 0.7, colsample_bytree = 0.7, n_estimators = 100,
4                     grow_policy = 'lossguide',
5                     max_leaves = max_leaves, n_jobs=-1)
```

gistfile1.txt hosted with ❤ by GitHub view raw

Linear Regression was straight and simple as it doesn't have any parameters to tune. Then


```
alp = [0.01,0.1,1,10]
l1_ratio = [0.01,0.1,1]

for i in alp:
    for j in l1_ratio:
        model = ElasticNet(alpha=i,l1_ratio=j, max_iter=250)
        model.fit(X_train,y_train)
        y_pred = model.predict(X_val)
        rmse = mse(y_val,y_pred)
        print(f"For alpha {i} and l1_ratio {j} the RMSE is {rmse} ")
        print('*'*80)
```


For the above code our output will be like the below

```
For alpha 0.01 and l1_ratio 0.01 the RMSE is 5.155661656035502
********************************************************************************
For alpha 0.01 and l1_ratio 0.1 the RMSE is 5.15576413040694
********************************************************************************
For alpha 0.01 and l1_ratio 1 the RMSE is 5.157393379233565
********************************************************************************
For alpha 0.1 and l1_ratio 0.01 the RMSE is 5.156879754507387
********************************************************************************
For alpha 0.1 and l1_ratio 0.1 the RMSE is 5.158724404273319
********************************************************************************
For alpha 0.1 and l1_ratio 1 the RMSE is 5.1707991599495635
********************************************************************************
```

We'll be getting 12 output as we have 12 combinations.

This is the code I used to tune the best parameter for ElasticNet, here for every loop a model is trained with an alpha value chosen from the list 'alp' and the second/inner loop runs with the L1_ratio values. With the model trained we are predicting the values of X_val and calculating it's MSE. The same is done for Ridge and Lasso Regression but with only 1 loop running for choosing the best alpha value.

Now for every model, after choosing the best parameter values, I predicted the value for the X_val and X_test which is required for the submission and reshaped it, because

```
2   pred_test_array  = model.predict(X_test)
3
4   pred_val_array = np.reshape(pred_val_array, (-1, 28),order = 'F')
5   pred_test_array = np.reshape(pred_test_array, (-1, 28),order = 'F')
```

gistfile1.txt hosted with ❤ by GitHub view raw

Finally we are making a form eligible for the submission

```
1   sub = pd.read_csv("sample_submission.csv")
2   sub_1 = sub.iloc[:30490,:]
3   sub_2 = sub.iloc[30490:,:]
4   f_cols = sub.columns[1:]
5
6   for i in range(len(f_cols)):
7       sub_1[f_cols[i]] = pred_val_array[:,i]
8       sub_2[f_cols[i]] = pred_test_array[:,i]
9
10  sub = pd.concat([sub_1,sub_2])
11  sub
```

gistfile1.txt hosted with ❤ by GitHub view raw

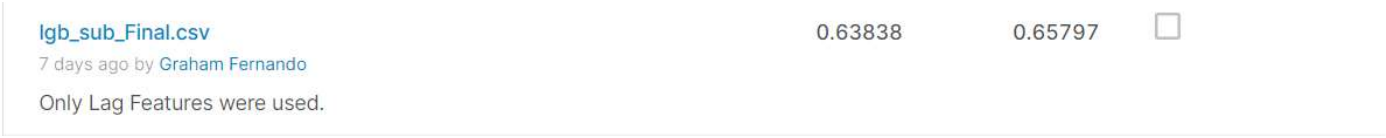
First we are reading the submission.csv file, then taking the first 30490 rows which are the validation rows or the rows used for public score validation and the last 30490 rows are the evaluation rows used for the computation of the private score. Creating two DataFrames and filling it with values we previously predicted and have converted/reshaped, then concatenating the two DataFrames as one whole DataFrame. These methods are the same/common for all the model.

	id	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
0	HOBBIES_1_001_CA_1_validation	0.934385	0.759724	0.740898	0.481377	1.000074	0.873161	0.996481	0.848803	1.553789	0.978018	1.509932	1.2468
1	HOBBIES_1_002_CA_1_validation	0.624992	0.447826	0.332735	0.566931	0.298697	0.490580	0.454386	0.516468	0.344637	0.242859	0.656416	0.3884
2	HOBBIES_1_003_CA_1_validation	0.457367	0.404617	0.330503	0.215410	0.501450	0.924517	0.723333	0.419403	0.653685	0.259109	0.496410	0.4855
3	HOBBIES_1_004_CA_1_validation	2.074015	1.215939	0.714386	1.473875	1.328342	1.892345	3.258144	1.654869	1.398039	0.973421	1.603037	1.7472
4	HOBBIES_1_005_CA_1_validation	1.289919	1.126062	1.553137	1.213806	1.267496	1.300779	1.650105	1.159681	1.622606	1.053520	1.374912	0.9243
...	...	...	...	...	...	...	...	...	...	...	...	...	...

This will how the output of the above code look like, we'll have the product id and it's forecast for the next 28 days, F1 - F28.

The final score for all the models received

	Model	Private_Score	Public_Score
0	LightGBM	0.63838	0.65797
1	CatBoost	0.70170	0.73363
2	XGBoost	0.72175	0.74070
3	Linear_reg_with_rolling_window	3.73342	0.75996
4	Linear_reg_without_rolling_window	0.70399	0.72709
5	Elastic_net	0.70314	0.72870
6	Ridge_reg	0.70462	0.72694
7	Lasso_reg	0.70403	0.72746



IX - Conclusion:

After trying multiple algorithms, LightGBM gave the best score, with **0.63838** on the private which is on the **top 5% on the private leaderboard**. Training the models also doesn't take a lot of time.

X — Future Work:

This model can be further improved by using LSTM a deep-learning model, or with a better cross validation model, like making the whole model into 3 or more cross validations for better training. Few extra features making sure the model doesn't overfit as this problem easily overfits. Some features with demand and prices, although I tried few

Reference:

1) M5-Accuracy/Notebook M5 Accuracy.ipynb at master · aakashveera/M5-Accuracy (github.com)

2) M5_accuracy/EDA_FE.ipynb at main · Deshram/M5_accuracy (github.com)

3) Simple Exponential Smoothing for Time Series Forecasting | by Nicolas Vandeput | Towards Data Science

4) Data | Free Full-Text | Machine-Learning Models for Sales Time Series Forecasting | HTML (mdpi.com)

5) Machine Learning in Retail Demand Forecasting | RELEX Solutions

6) Applied AI Course

Link to the solution can be found on my GitHub page.

You can check my LinkedIn profile Here.