CS725 FML-PROJECT

Predict Future Sales

Deepak Singh Baghel (203050005)
Ankush Agrawal (203050007)
Abhinandan Singh (203050031)
Ajay Sarup Jain (203050036)
Shubhranshu Maurya (203050096)

8 December, 2020

Contents

1 Introduction

Predict future sales is the final project for the "How to win a data science competition" Coursera course. In this competition we were required to work with a challenging time-series dataset consisting of daily sales data, kindly provided by one of the largest Russian software firms - 1C Company.
It was asked to predict total sales for every product and store in the next month.

2 Abstract

We are provided with daily historical sales data. The task is to forecast the total amount of products sold in every shop for the test set. It is worth noting that the list of shops and products slightly changes every month. Creating a robust model that can handle such situations is part of the challenge. Time series problems have series of data and forecasting is done for next values in series but in this future sales prediction problem forecasting is desired for specific shop and item combination, building model which can forecast around 0.22 Million of series in test data is big challenge. Since we are provided with daily historical sales data and we are asked to predict the sales of each product for each shop over the period entire month, data pre-processing was a very crucial part of this project which was performed to reshape our dataset according to the problem statement as well as controlling the outliers was also required to be done as part of pre-processing. This certainly helped to produce better scores for the prediction models. Analysis of various prediction models is performed to come up with the model which can produce better result for the purpose of future sales.

3 Detailing the dataset

3.1 Link for dataset

Dataset

3.2 File descriptions

- **sales_train.csv** - the training set. Daily historical data from January 2013 to October 2015.
- **test.csv** - the test set. You need to forecast the sales for these shops and products for November 2015.
- **sample_submission.csv** - a sample submission file in the correct format.
- **items.csv** - supplemental information about the items/products.
- **item_categories.csv** - supplemental information about the items categories.
- **shops.csv** - supplemental information about the shops.

3.3 Data Fields

- **ID** - an id that represents a (Shop, Item) tuple within the test set
- **shop_id** - unique identifier of a shop
- **item_id** - unique identifier of a product
- **item_category_id** - unique identifier of item category
- **item_cnt_day** - number of products sold. You are predicting a monthly amount of this measure

- **item_price** - current price of an item
- **date** - date in format dd/mm/yyyy
- **date_block_num** - a consecutive month number, used for convenience. January 2013 is 0, February 2013 is 1,..., October 2015 is 33
- **item_name** - name of
- **shop_name** - name of shop
- **item_category_name** - name of item category

4 Graphs

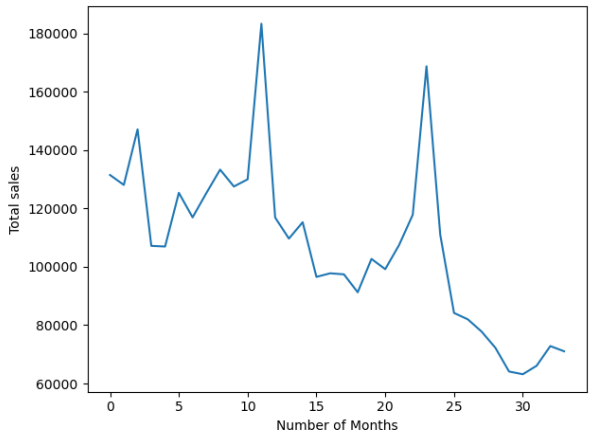

Figure 1: Sales vs Month

As we can see in the figure 1 total sales has decreasing trend over the months. Total sales includes sales of all the shops in data-set for each month from January 2013 to October 2015.

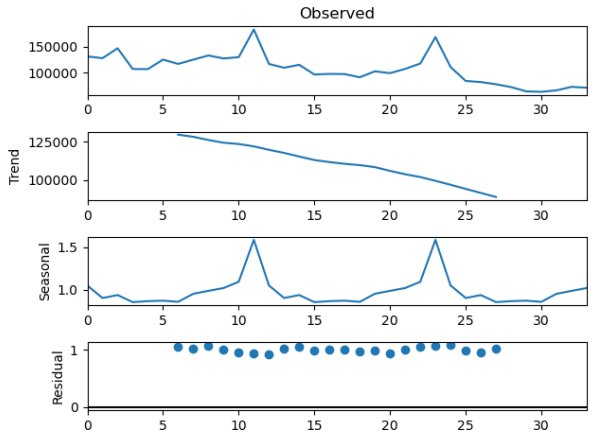

Figure 2: Seasonal Decomposition

As we can see in the figure 2 overall tread is decreasing over time and no seasonality is observed since only two times sales were high in span of 34 months which is not indication of seasonal behavior of data,

residual is also stationary. We can come to conclusion that the null hypothesis is rejected and alternate hypothesis is accepted because data is stationary.

5 Proposed Models

In this section we will describe the models used for predicting Future Sales and the results obtained on the sales dataset using these models. We have used 2 dataset for this project which are the essential ones: sample_submission.csv and test.csv

5.1 Neural Network with Backpropagation

In the train dataset we have 5 features: date, month(block no.) [0-33] as 34 months data in train set, item_id, shop_id and item price with their item_count which represents the item sold on that day or month. We have to predict item count for each shop_id and item_id present in the test set.
In this Neural Network model, we have dropped the date and item_price feature from the train set.
Reason:

- Monthly prediction is required so we thought either date is required or block_no but not both. So, we dropped the date feature in this model.
- item_price is not of significant importance as it is not present in test set, so during calculation of feed forward neural network it is dropped.

We are left with 3 important features: block_no, item_id and shop_id.
The train data is split into train and dev set into following percentage:

- Train data : 99.9999%
- Dev data : 0.0001%

During calculation of weights for Neural Network all 3 features are groupby and monthly sum of item_count is done. This train model is used on dev data to find the prediction. The test data has 2 features, so we will add next feature block_no, which is 34 because we have to do next month prediction. The model is run and the output file is generated which consist item_count for next month and it is submitted on kaggle. The results obtained are shown in Table 3. All error(values) are rsme(root mean square error).

max_epochs	2
batch_size	25
learning_rate	0.01
num_layers	1
num_units	100
lamda	5

Table 1: Hyperparameters for NN.

num_layers : No. of hidden layers in Neural Network.
num_units : No of units in a hidden layer.
lamda : It is the constant of regularization.
max_epochs : The number of times our model is trained.
batch_size : It is the size of data which is processed in NN.
learning rate : It is the constant used to calculate updated weights and biases.

Our problem statement is time-series model but from the result shown above it can be said that "The Neural Network with backpropagation shows good result in forecasting".

5.2 Linear Regression

We have provided with the data which has sales value for each day. As we need to predict in total sales for month so we need to aggregate data on monthly basis. Secondly , the data is dynamic in nature i.e. new shops and new items are added or removed frequently due to which test data have some item-id and shop-id combination which are not present in test data. We have done data pre-processing to handle these 2 cases.

We have first create a data frame with combination of all shops-ids and item-ids for each particular month. This will incorporate even the combination of item-id,shop-id not present in training data but which we may come across in testing data. In this data frame we have left join with aggregated training data to bring in aggregated sales and fill Na's with zero(Na's are there as some combination of shop and item for particular month is not present in training data). Zero indicates no sales for the particular item on that particular shop. We have also bring in mean price of each item in the above data frame. At last we have used items.csv file to bring in item category id of the item by left joining on item-id.

The above step gives us a data-set that contains aggregated sales monthly for each month(date-block-num). Now,we tried to incorporate data of previous month as feature but we need to figure how many previous month data will give the most accurate result.

We have tried various combination and comes to conclusion that previous 4 month data is most appropriate as increasing the month further will only increase the features(time overhead) but don't show any significant improvement in the result.

Now, we have data set ready which have 9 columns(shop-id,item-id,item-category-id,price,4 columns of previous month sales and sales of that month). we split data set into two sets train and dev with split ratio 0.99. we have applied linear regression with regularization to avoid over-fitting to the training set. After tuning the model we have get the train error as 6.627 but its performs far better on the dev data and gives error of 2.162.

In test data we are only provided with shop-id,item-id so we need to make this test data align with train data by adding required features. Firstly, we directly add a column name date-num-block and fill it with 34. After using train data by left joining on columns we bring previous 4 month sales and mean price in the test data. Next using items.csv, Similarly, as done in train data we have added item-category-id to our test data. Now our data has all the required columns so we have predicted the sales of month 34 using the parameter learned while training the model. we have got the following result as mentioned in the table 3.

The regularization parameters used in model :

$$\lambda = 1e - 08$$

5.3 Random Forests

In Random forest the data pre-processing step is similar to what we have done in Linear Regression. Here, After processing the data and generating the final training set we have used sklearn library for implementing this model.

First we import the library and have built the basic model with the default values. It will result in over-fitting of the model as we haven't control the depth of the tree so it tries to completely fit the training set which will result in very low training error but when checks on test data the error is quite large.

The error metrics for default random forest models is : Train error = 0.754 and test error = 4.825

As from the above data we can clearly see this is a case of over-fitting so we have tried to control it by restricting the depth of the tree. First, We have tried **max-depth = 15** which gives the following results : Train error = 1.525 and test error = 2.836

From the above data we can observe by restricting the height we have improve the performance on the test data. The test error is almost half-ed for this configuration of the random forest model. We think of restricting it further as still there is gap between test and train error. so we have run one more iteration but this time we restrict **max-depth = 10** and it gives the following result : Train error = 1.789 and test error = 1.725.

Now we can clearly see we are able to get test error less than train error and this configuration seems best do far. Hence, The max depth parameter of the model - **max-depth = 10**.

5.4 Weighted Moving Average

As seen in the figure 2 Trend is decreasing over months that indicates total sales is decreasing. A simple observation from data-set is that the probability of selling of item for a given shop highly depends on the history of sales of same item in the same shop and more interestingly that dependency on history is short term means sales of item depends on sales of same item in the same shop in the last month.

In sales data file(sales_train.csv) only block number(month index from January 2013), shop id, item id, and item count is taken because we have to predict the total sales in next month hence price of item becomes useless also date of sale is neglected since we have to group the entire data-set month wise which can be achieved by block number.

The strategy here is to find out the sales of shop id, item id pair for each month and then use weighted moving average model to predict the sales for next month to do that we need to groupby the train data by block number, shop id, and item id and use the aggregate function to find the sum of sales for corresponding item. The real challenge comes when we need to find out the total sales for each item of each shop because train data has around 3 Million entries and it takes lots of time to train the model for each combination of shop id and item id but we are given with test data to predict hence we should only compute the result for shop id, item id given in test data ultimately we are going to work with test data only so computing result for every combination of shop id and item id is unfruitful.

Weights given to history of sales is as follows.

- Last month = 60 %
- Second last month = 20 %
- Rest of the months = 20 %

Weight given to each month is hyper-parameter here and one need to tune it to get the minimum error. Left outer join is used on test data and if the given shop id and item id is not in the history of sales simply zero is taken in place of NaN(produced by left outer join).We can see in the table 3 this model has performed very well on test data and able to perform better than rest of the models used.

5.5 LSTM : Long short-term memory

The dataset given is a time series dataset so its intuitive that a recurrent neural network(RNN) will work. But RNN is known to suffer "vanishing gradient problem", therefore to overcome this problem LSTM(Long

Short Term Memory) is used. The LSTM model that is used here is taken from the `tensorflow` library. Out of the 5 features namely `date`, `date_block_num`, `shop_id`, `item_id` and `item_price`, we use only 3 features namely `date`, `shop_id` and `item_id` to predict item count and we drop the features `date_block_num` and `item_price` because they `date_block_num` is useless if we use the `date` feature and using `item_price` has no impact on the predictions. Also the `date` feature is converted from object dtype to Datetime dtype. And then this `date` feature is set as the index. A MinMaxScaler has been used to preprocess the data. This scaler function is taken from the `sklearn` library. A 5 layer LSTM is created using the following number of neurons in each layer: 100, 100, 100, 100, 50. A dropout rate of 0.1 has been defined for the first 4 hidden layers. In each layer ReLu is used as an activation function. Also "Adam" optimiser has been used. In this optimiser the learning rate is varied with each gradient.

Scaler Function	Min-Max
Number of hidden layers	5
Neurons per hidden layer	100, 100, 100, 100, 50
Dropout rates per hidden layer	0.1, 0.1, 0.1, 0.1
Activation funtions	ReLu(for each layer)
Optimizer	Adam
Batch Size	30000
Epochs	5

Table 2: Hyperparameters Of LSTM.

While training the model a batch size of 30000 is used. At first it was decided that 50 epochs will be used while training the data but due to the lack of computing power it would have taken several hours to process the huge dataset so later only 5 epochs were used in training the model. And since there were less epochs, the train error was 4.1477.

Lastly the the trained model was used on the test data and the predictions were obtained. These predictions were submitted on kaggle and the score 1.41241 was obtained.

5.6 ARIMA Model

ARIMA model is ideal choice for time series forecasting problems as we have seen the other models described in report tried to produce best result but time series specific models generally perform better. As seen in the section **??** Trend is decreasing over months that indicates total sales is decreasing and there is no seasonality because out of 34 months only two times sales were high hence we can conclude from data that there is not seasonal factor associated, from the Residual figure one can observe that it is almost stationary hence we can finally conclude that data is stationary and null hypothesis is rejected since there is no time depended structure.

In the sales data file(sales_train.csv) only block number(month index from January 2013), shop id, item id, and item count is taken because we have to predict the total sales in next month hence price of item becomes useless also date of sale is neglected since we have to group the entire data-set month wise.

The strategy here is to find out the sales of shop id, item id pair for each month and then use ARIMA model to predict the sales for next month. The real challenge here is we need to predict the sales of each item of shop for the next month and there are around 3 Million combination of shop id and item id, ARIMA model takes entire series as input and output the forecasting for requested number of elements in series, to use ARIMA model in our problem we need to use last 34 months sales of each item of shop as series to input in ARIMA model but it takes lots of time, to achieve this task we need to groupby

the train data by block number, shop id, and item id and use the aggregate function to find the sum of sales for corresponding item, we should only compute the result for shop id, item id given in test data ultimately we are going to work with test data only so applying ARIMA model to each combination of combination of shop id and item id is undesirable.

Given only 34 months of sales data as series it is really hard for ARIMA model to train even though most of the entries are zeros because of either no sales or dynamic shop id, item id in test data-set which was not in the train data, to train the model order = (0, 0, 0) is used where p = 0, d = 0, and q = 0 are the parameters of model.

Performance of model is given below:

- Train error = 1.002
- Test error = 1.17767

As we can see in the above table test error is less than the rest of the models used but it is greater than the weighted moving average model because ARIMA could not be trained well with just 34 entries in time series and most of them were zero too, given this constraint performance of ARIMA model is considerably good.

6 Results

Model	Train error	Dev error	Kaggle Score
Neural Network	0.677	0.677	1.217
Linear Regression	6.627	2.162	1.386
Random Forests	1.789	-	1.725
LSTM	4.147	-	1.412
Weighted Moving Average	0.939	-	1.057
ARIMA	1.002	-	1.177

Table 3: Comparison of Various Models Used.

7 Github link

This is the link for the Github repository of our project.

8 Kaggle link

This is the link for the kaggle competition.

9 Conclusion

- Sales vs month plot shows that sales are decreasing over time
- The second plot also shows a decreasing trend, also we observe that there is nothing like seasonality. In over 34 months span sales was good in only 2 seasons, hence data is stationary and null hypothesis is rejected since since there is no time depended structure.
- As per the kaggle submission scores we observe that for the given underlying dataset weighted moving average performs best since there is no seasonality present in the data

- In the Neural Network model it can be concluded "that backpropagation shows good result in forecasting item_count especially for our dataset".
- Linear regression is not able to fit training data as there are many variations in the data set which are difficult for it to generalize but when apply on the test data it shows good result which may be due to sparse data set.
- In the Random forest it can be concluded that on restricting the depth of the tree, we are able to avoid over-fitting and hence it has shown better performance on test data as compared to default model where depth is not restricted.
- Weighted moving average modle was able to perform better than all the models used because future sales was highly dependent on sales of last month up to 60 % and 20% on second last month.
- For LSTM the number of neurons were high so each epoch took longer and more memory space, hence only 5 epochs were done which resulted in higher train error but running the LSTM for more epochs say 50 epochs would have resulted in much lower train error.
- ARIMA model has shown significant performance with constraints that input series to model had only 34 entries for each month, and model had perform better than most of the models used since it was time series based problem.

References

[1] https://towardsdatascience.com/time-series-forecasting-using-auto-arima-in-python-bb83e49210cd

[2] https://www.kaggle.com/jayantawasthi/rnn-used-to-predict-future-sale

[3] https://www.kaggle.com/szhou42/predict-future-sales-top-11-solution

[4] https://github.com/VipinindKumar/Predict-Future-Sales/blob/master/sales1(oct_as_pred).ipynb

[5] https://github.com/krishnaik06/ARIMA-And-Seasonal-ARIMA/blob/master/Untitled.ipynb