Machine Learning Engineer Nanodegree

Capstone Project

Sasa Pavkovic
March 11th, 2019

I. Definition

Having an ability to know what to expect from the future sales is power full for any company. Usual approach is to take a look at historical data and use those in order to have an opinion about the sales in the future.

The need in business for solving this type of problem is very high as it can be generalized relatively easily. Some similar, but different questions can then be answered:

- How much revenue can we expect from diffferent segments?,
- In which product segments to invest most of marketing budget? or how to balance the marketing budget to maxmize CLV. Trying out a one of the approaches with neural networks might be a challenge but also very interesting.

In this project we are going to take a look at the sales made by by one of the largest Russian software firms - 1C Company in the period from Jan 2013 to October 2015. The data is provided through the Kaggle competition 'Predict Future Sales'.

Link: https://www.kaggle.com/c/competitive-data-science-predict-future-sales

Problem Statement

Using the historical data, provided by the 1C Company, we will make predict the total items sold for every product in every store for the month of November 2015.

As we are making predictions about a continuous variable we are going to use a couple of approaches that are based on regression. We will start with a Linear Regression model as a baseline model and then continue on with some more complex models and compare the results from each of them. This will enable us to find the best model for the data that we have.

The data that we are having is going to have an important temporal component, but here we will not attempt a time series analysis and predictions, but focus on how this types of problems can be solved by feature engineering that enables us to use the standard machine learning approaches.

More info on 1C company can be found here https://en.wikipedia.org/wiki/1C_Company

Metrics

In this project we will be using the RMSE (Root Mean Squared Error) as a metric to compare the different approaches. MSE (Mean Squared Error) is the average of the squared differences between predictions and the observed data. The closer it is to 0 the better, but generally the score itself depends on a problem and the data that is available, hence is not easily comparable across different problems. As we will use it for comparison of different problems on the same data set, then this is appropriate metric. It is also the metric used in the Kaggle competition, and we can get the final results from Kaggle during the submissions of the predictions.

More about MSE and RMSE can be found in the links below:

- https://en.wikipedia.org/wiki/Mean_squared_error
- https://en.wikipedia.org/wiki/Root-mean-square_deviation

II. Analysis

Data Exploration

Data for the project has been provided in 5 separate files and here are the data descriptions that are available.

File descriptions

- sales_train.csv - the training set. Daily historical data from January 2013 to October 2015.
- test.csv - the test set. The test that we will use for predictions
- items.csv - supplemental information about the items/products.
- item_categories.csv - supplemental information about the items categories.
- shops.csv- supplemental information about the shops.

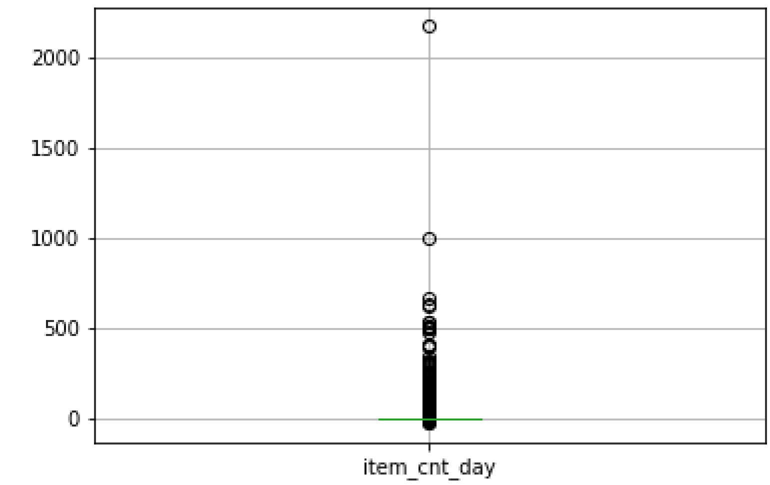
Data fields

- 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 item
- shop_name - name of shop
- item_category_name - name of item category

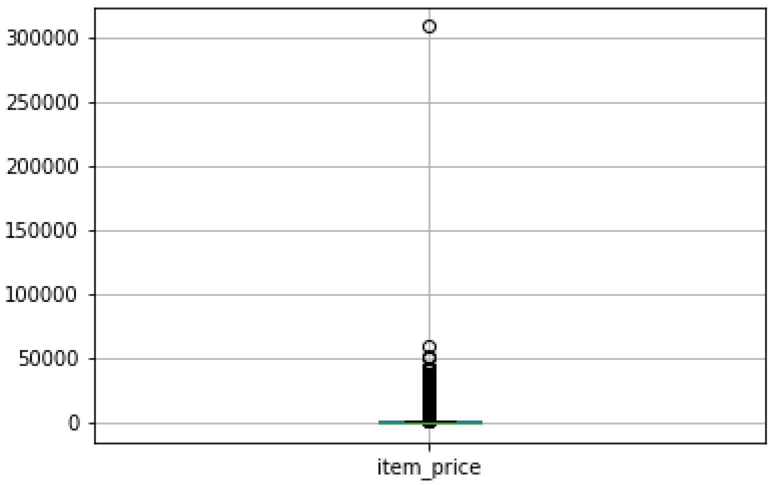
The data prepared for the training is representing daily observations of shop and item sales with their respective price and total number of items sold. Initial inspection of data has shown that the data is in the.csv file format and separated through several .csv files. One of the initial tasks is going to be merging the data from the different files into a format where 1 observation holds all the needed features. The descriptions of the columns are in Russian cyrillic.

Outliers

When looking at the dependent variable for the whole period there seem to be some days when many items were sold. Still, more then 1000 items sold in one day and shop has been reported only once and that is in October, which is a high seasonality period as we have seen, hence these outliers are left in the dataset.



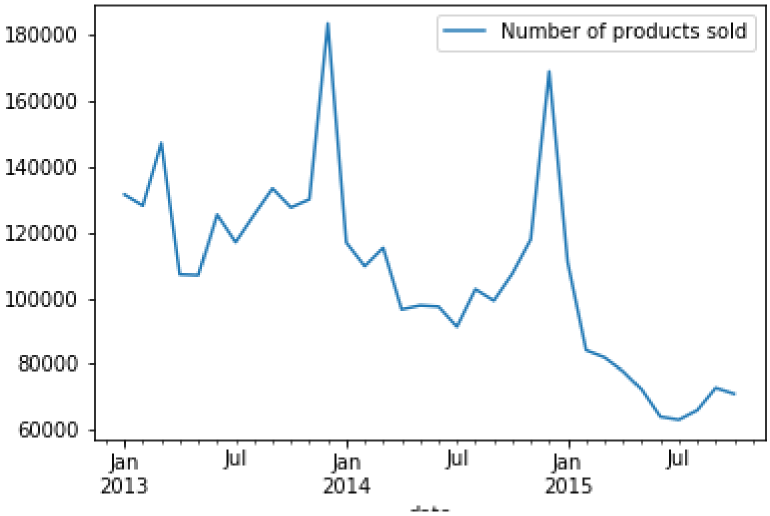
Here the item_price of an item was a more extreme outlier hence this data point was removed from farhter analysis.



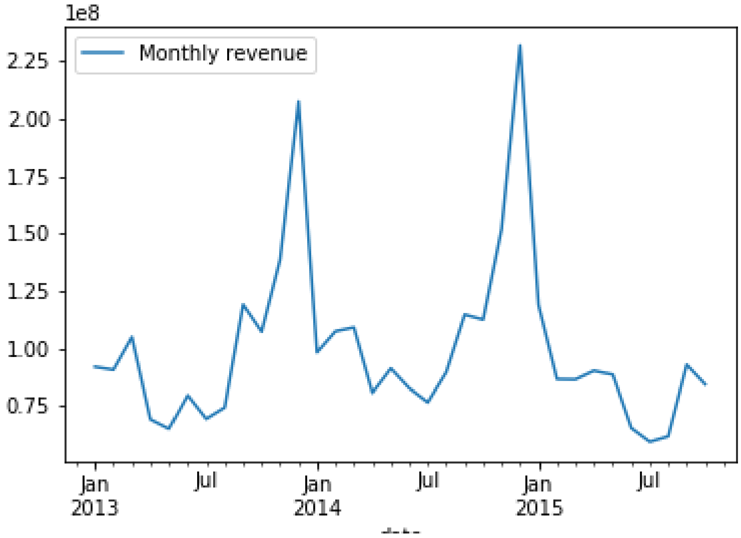
Data inconsistencies

There was just one observation where item_price was below 0, hence this observation was removed from analysis. There are many observations where the total number of items sold was negative, but this is due to more returns then sales happening in the same day for a certain item and shop, so they were not removed from the dataset.

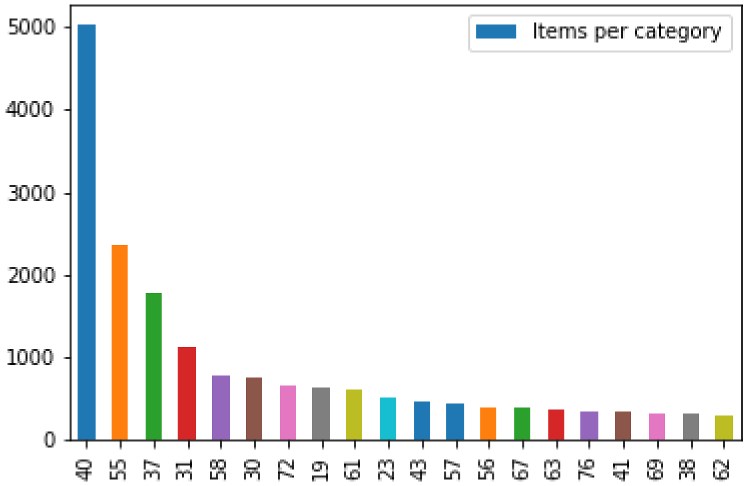
Exploratory Visualization



The above plot shows the development of daily sales over the period od training data. The number of items sold seems to be going down with a negative trend. The seasonality is also present in the data with the last quarter being better then the rest of the year.



The above plot show the revenue for the period of the training data. Here the overall trend is more constant which means that in the recent years less items were sold but for a higher price keeping the revenue constant. Seasonal component of the trend is also present with the same seasonality as for items sold.



Here we can have an overview of the size of the first 20 biggest item categories. There seems to be one that is quite bigger than others which would make that one important.

Algorithms and Techniques

In the project we will use Linear Regression algorithm as a baseline to compare against other algorithms. Other algorithms that we will compare against are:

1. Ridge Regression (Least Squares implementation with l2 regularization), parameters

- Alpha = 0.1 (regularization strength)

2. XGBoost Regressor(Gradient Boosting, parallel tree boosting)
    - link: https://xgboost.readthedocs.io/en/latest/), parameters
    - max_depth=8, (Maximum depth of a tree)
    - n_estimators=1000, ()
    - min_child_weight=300, (Minimum sum of instance weight needed in a child)
    - colsample_bytree=0.8, (This is a family of parameters for subsampling of columns)
    - subsample=0.8, (Subsample ratio of the training instances)
    - eta=0.3, (Step size shrinkage used in update to prevents overfitting)
3. DNN (simple Keras implementation)
    - Input layer
        - 48 nodes
    - input_dim = 32
    - kernel_initializer='normal'
    - activation='relu'
    - Hidden layer
        - 96 nodes
    - kernel_initializer='normal'
    - activation='relu'
    - Output layer
    - 1 node

    - kernel_initializer='normal'
    - activation='linear'

On all ocasions regression will be used to predict the number of items sold for each item and shop. The same dataset format will be used for all the techniques. Firstly the data will be processed in a way that is suitable for all algorithms to consume and then will be used for predictions and the algorithm that achieves the best (lowest) RMSE will be chosen as a solution.

Data will be split into 3 smaller sets, training data (till Oct 2015), validation data (Oct 2015) and the test data. Test data will be checked through the Kaggles submission platform as the actuals for Nov. 2015 are not known.

Benchmark

As the data for this project is coming from the Kaggle competition "Predict Future Sales" many other Kagglers have been making their attempts at predicting the Nov 2015 sales data hence their performances could be used as benchmark. Currently the best score achieved is RMSE of 0.79215 with an unknown algorithm.

III. Methodology

Data Preprocessing

The idea is to combine all the available data so that we have one observation be unique for month, shop and item and a dataframe would be created. Then additional summaries would be added to the dataframe. These would include the engineered features from the temporal component of the data set. Test data set contains items that have not been previously sold hence test and train data have been combined. The data has been split such that the train data is months from 12-32 (first 12 months have been removed due to lagging used), validation set is 33 and the test set is 34.

Here is the list of features that were created by:

1. aggregation

- item_cnt_month (monthly items sold)

2. lagging and aggregations:
    - item_cnt_month_lag_1 (previous period lags of monthly items sold)
    - item_cnt_month_lag_2
    - item_cnt_month_lag_3
    - item_cnt_month_lag_6
    - item_cnt_month_lag_9
    - item_cnt_month_lag_12
    - block_avg_item_cnt_lag_1 (previous periods lags of avg. items sold by month)
    - block_item_avg_item_cnt_lag_1 (previous periods lags of avg. items sold by month and item)
    - block_item_avg_item_cnt_lag_2
    - block_item_avg_item_cnt_lag_3
    - block_item_avg_item_cnt_lag_6
    - block_item_avg_item_cnt_lag_9
    - block_item_avg_item_cnt_lag_12

    - block_shop_avg_item_cnt_lag_1 (previous periods lags of avg. items sold by month and shop)
    - block_shop_avg_item_cnt_lag_2
    - block_shop_avg_item_cnt_lag_3
    - block_shop_avg_item_cnt_lag_6
    - block_shop_avg_item_cnt_lag_9
    - block_shop_avg_item_cnt_lag_12
    - block_cat_avg_item_cnt_lag_1 (previous periods lags of avg. items sold by month and item category)
    - block_cat_avg_item_cnt_lag_12
    - block_shop_cat_avg_item_cnt_lag_1 (previous periods lags of avg. items sold by month, shop and item category)
    - block_shop_cat_avg_item_cnt_lag_11
3. other
    - month (1-12)
    - shop_item_last_purchased
    - item_last_purchased
    - shop_item_first_purchased
    - item_first_purchased

Implementation

The process is that the training data is fed to the model and then the model is predicting the values on the validation data in order to get the idea which one of the competing algorithms is best.

The first model that was attempted was the Linear Regression from the python sklearn library. This was quite straight forward. Then the Ridge model was attempted next, then XGBoost and DNN as defined in Algorithms and Techniques.

Refinement

I have tried adjusting the the Ridge Regression alpha level for 0.5, 10 and 100 and all models produce no improvement when compared to the Linear Regression, hence it seems that a simpler model then the regular Linear Regression does not exist.

The XGBoost model has been attempted with max_depth 6, 8 and 10 and the best was the model with the max_depth=8.

Three different versions of DNN with 3 different structures and the one mentioned in the Algorithms and Techniques is the one that had the best performance. The network that was wider and deeper seemed to perform the best. Still, more time could be spent in creating a more elaborate neural network. This was also at the limit of my hardware capacities.

Scaling of all relevant variables was attempted to check the impact on the models, but no obvious effect on the performance of the models was noticed. Still, the scaling of the variables was kept and used hence forth.

IV. Results

Model Evaluation and Validation

On the data prepared the Linear / Ridge models achieved the RMSE of 1.02256, which is somewhat worse then the benchmark of 0.79215. This seems in the range of what i would expected, and also shows that a very simple model on this dataset already achieves a good result.

XGBoost with selected parameters (max_depth=8, n_estimators=1000, min_child_weight=300, colsample_bytree=0.8, subsample=0.8, eta=0.3, seed=42) achieved an RMSE of 0.99509 which is an improvement from LG, but still some ways away from the benchmark. It is possible that more parameter tuning needs to be done in order to achieve better results.

DNN with the structure as explained in Algorithms and Techniques achieved an RMSE of 1.03713 which is slightly worse then LG. There might be several reasons for such a performance, one of the major ones are the type of the problem to be solved (data is well structured and regression is needed) size of the dataset (DNN 'prefer' abundance of data), and lack of computational power (for DNN).

All the models performed well, maybe only surprising for me was the difference in their performance which was relatively small. This might suggest that the biggest changes might yet be achieved through additional changes to the input data, rather then selecting an additional model.

Justification

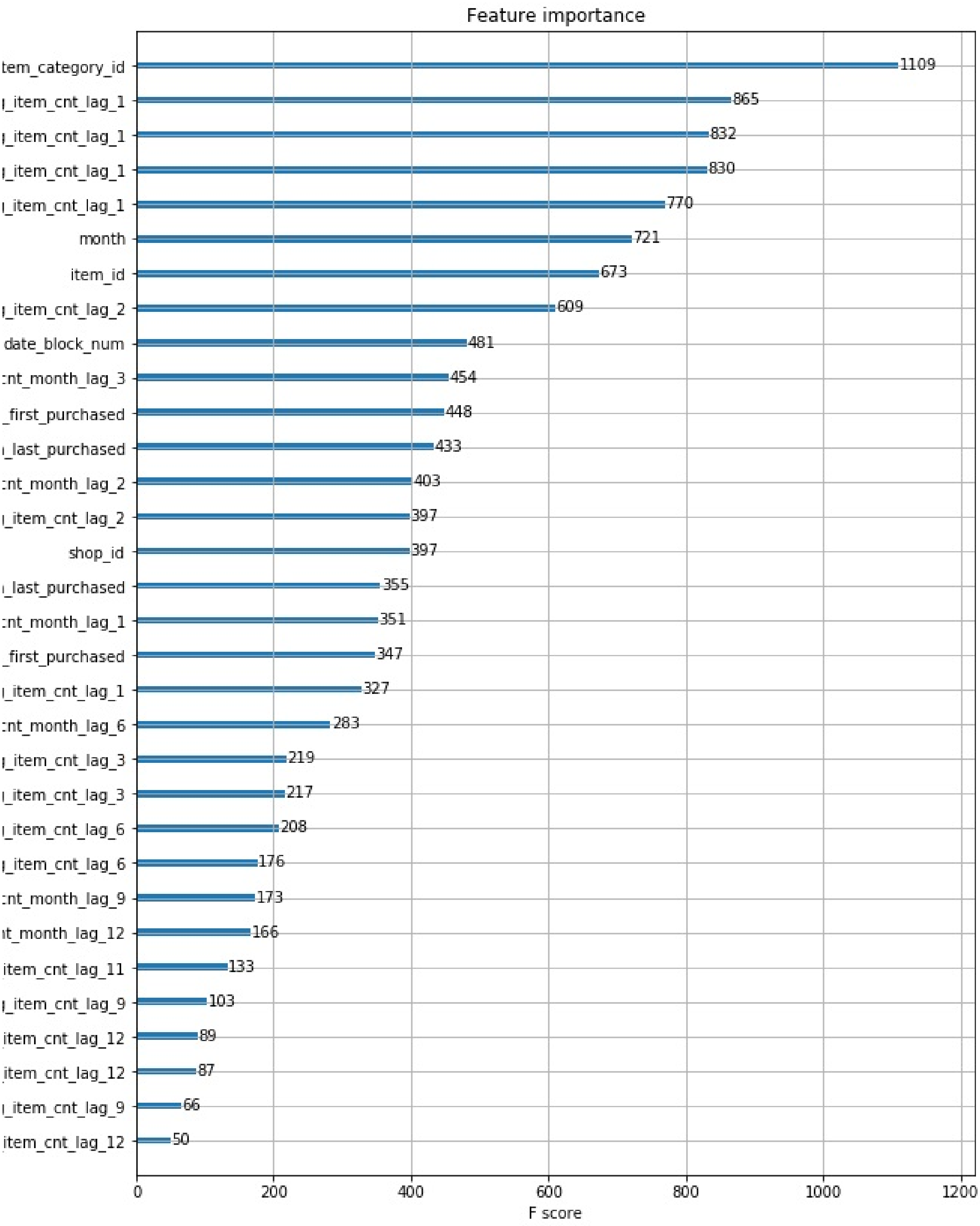
Although the performance of the best algorithm used was weaker then the benchmark, still a good result has been achieved. The model that would be selected for the production would be XGB or also possibly Random Forrest algorithms which should achieve a bit worse but still respectable results.

If the sales behave similar to previous years, then an improvement in ability to predict future sales have been achieved. Understanding the market is difficult, and there might be some features about it that have not been included in the data, hence sudden changes would affect any model seriously.

The performance of the algorithms is somewhat consistent with the research done by: Caruana, Rich, and Alexandru Niculescu-Mizil, "An empirical comparison of supervised learning algorithms." .

V. Conclusion

Free-Form Visualization



Above is the chart of feature importance scored by the F score from the XGBoost model (best performing). It is interesting that the most important feature is the item_category_id. In my mind this suggest that more could be done with the feature engineering.

Also the more important sales features are the ones that have happened in the recent times. That makes sense as the products that were 'hot' last month are also likely to be important in the next month, but not so much in the next year (the types of products are computer games and all related to it). Also month is a fairly important feature, which also makes sense due to the seasonality that we observed previously.

Reflection

The initial problem of predicting the sales in the next month was set up as a problem where we want the predict a continuous variable. The most consuming task was preparing the data for the model. As we have also a temporal component we had to find ways to transform that information into some quality features. We have applied a couple of algorithms and checked their performance on the same metric and also compared to a benchmark from Kaggle.

It was interesting to find that the linear regression was not that far off from the other algorithms that we tried. I would have expected a huge improvement from the more advanced methods. Perhaps more tuning or changing the structure of the DNN might change the difference to be more drastic.

The most difficult aspect of the project was data wrangling and thinking how to create features that bring most out of the dataset. This is usually the most time consuming aspect of many projects not just this one.

The best model, XGBoost, has been used in many competitive challenges and also in the production environment of many famous companies and it is very robust to the type of problem that we have here. In a way it is not a surprise that it performed best.

Improvement

There is potential for improvement in this project! For instance i believe that still more can be done with feature engineering, especially with combining the item_category_id with some other temporal aspects as maybe the recent item categories are more relevant then the distant ones. Also, we have some item_ids in the test set that we do not have in the training set, hence maybe using the averages of the item categories can help the model in finding the future sales of those items.

More complex architectures of DNNs could be employed with additional layers as well as some already prepared NN could be used that are already prepared for these types of problems. Then their parameters could be fine tuned.

Finally, XGBoost parameters could be expanded and more testing done in order to find a better model.

We know that better results can be achieved as the benchmark is still better then the solution we tried here.