**Adrian Maciej**

**Algorithm:** Decisions Trees / LSTM (unfinished)

**Git repository:** <https://github.com/AdrianMaciej/PredictFutureSales>

**Files description:**

* 1.PrepareData.ipynb – notebook to clean the data, create features and save the result in pickle file
* 2.CatXgb\_PredictOnFutureItemCnt\_v2.ipynb – notebook to create the Decision Trees model
* 3.LSTM – an initial try on doing another model with LSTM (lack of time to finish that properly)

**Idea**:

Looking at the data first assumption was to create a model that’s going to predict that month items count based on the price, previous items count values and so on. But when I looked at the test set I noticed that there is no ‘next month price’ or anything so I would need to assume that the price is not going to change which may or may not be true. So instead I went with the design that I use current month items sales and other data and predict the NEXT items sales. That way I only need to use last seen month in the data (33) as my test input.

**Cleaning phase:**

* item\_categories.csv, items.csv and shops.csv
  + Looking at names I thought about using some tokenizers to gather the most occurring words and using them as features but it didn’t improve the algorithm so I discarded that approach
  + I chose to use only the first part of shops (city) and first part of item\_categories (category\_type)
  + Both names have been cleaned from any unusual characters or signs to avoid duplicates and categorized into categorical variables
* sales.csv
  + Looking at outliers I discarded rows with item\_price > 100000 and item\_cnt\_day < 1500
  + I dropped row with the price that was below 0 and removed the ‘date’ column completely because I noticed that for some dates the day and month part were switched so it would not create correct month feature
  + Data was truncated to only those shops and items that appear in the test set to focus on predicting given the items count for given list of shops and items

**Feature engineering:**

* First thing after grouping the daily data into monthly data was to clip the item count to 0, 20 – according to competition information those are the limit that the prediction is going to be clipped into – best to train model on that limit to focus on solving the task more accurately
* Features created:
  + Month and Year – based on the date\_block\_num
  + Shifted data from NEXT month – that’s going to be the label Y for the model
  + Shifted data from previous [1, 2, 3, 4, 6, 12] months for items count, average price and revenue – to catch the last outcomes and last year values
  + Price features
    - min and max price of an item overall and in that shop
    - how that price looks compared to the min and max item price overall and in that shop
  + Rolling values for items count from last [2, 3, 4, 6, 12] months – to get mean and standard deviation on those windows
  + Finally how current price looks compared to last seen price (not necessarily last month price)

**Model creation:**

* For training data I used all the data after first 12 months without the last 2 months. First 12 months are dropped because some feature are generated based on previous 12 months so the first wouldn’t have proper once and it doesn’t seem right that data from past 2 years would make some changes now
* Validation data is the 2nd to the last month
* Test data is the last month
* At first I added mean features based on combination of item\_id, shop\_it and month but at the end I noticed that they are overfitting the model so I discarded those
* First model
  + I used CatBoost as first model to get an idea what features are important because it calculated the result faster than XGBoost
  + I played around with the depth of the tress – 4, 6, 8 – and even thou the differences were not that crucial – depth 8 seemed to perform slightly better
* Second model
  + Using top features shown by CatBoost with score > 1
  + Model parametrs:
    - Basically I increased the max\_depth, and n\_estimaters – I wanted to perform a grid search on those but because of lack of time I went with some higher values just in case
    - I lowered the learning rate to 0.05 and increased min\_child\_weight– again to improve accuracy of the model – 0.1 was just fine but 0.05 gave me some small boost that resulted in the rmse on Kaggle to drop below 1. Min\_child\_weight increased because with higher depth low values could result with overfitting.
    - I lowered the colsample\_bytree and subsample slightly to prevent overfitting

**Final results:**

The train and validation rmse dropped down to values around 0.77-0.81 and 0.82-087 respectively while the rmse on Kaggle scoring was around 0.99 – 1.02 which would suggest that the model is still overfitting rather than generalizing the problem.

One solution I found that would boost the accuracy would be to create several models and ensemble them by creating a linear regression model of all the results.

I was thinking about creating an LSTM model but it’s tricky because of many shop items combinations creating separate model for each one of them would be bad idea. Other option wast to discard the shop and item and just get the trend of items count as an input but that seemed like big generalization. I was trying to prepare data for multivariate LSTM using city and category\_type as features to generalize more the training set but I’m not sure if that would work correctly. Looking over the internet I couldn’t find a good solution for using LSTM for predicting results of different inputs.

At the end – comparing the result with other Kaggle kernels result I’m pretty satisfied with the result – it’s not the best for sure but given that it’s only one model and top of the leaderboard consists probably of the boosted ensemble model it doesn’t look so bad.

I’m pretty happy with the rolling function I wrote – it’s clever and short – adding new rolling value, changing the windows or functions applied is really easy.

I’m not happy with lack of grid search of the parameters – that could improve the model a lot but lack of time and cpu/gpu resources I was not able to do it.