Machine Learning Engineer Nanodegree

Capstone Project Report

Topic: Sales Prediction

audi-employee195
April 2019

1. Definition

This section give a first overview over the project itself, the problem statement, as shortly discussed in the proposal and the metrics we use in order to measure the performance of the prediction model. If initial explanations of certain concepts or terms are needed, the reader will find them here.

1.1 Project Overview

The project is about applying machine learning algorithms to time series data. The overall goal is to predict the sales numbers of products in a shop for the next month. We are given the dataset of a Russian software firm called 1C Company. This dataset is provided for a Kaggle competition as well (see: https://www.kaggle.com/c/competitive-data-science-predict-future-sales/overview). Thus, the domain background is set and the information as well as the data are provided there. The dataset consists of 99.4MB of csv-files and contains the following relevant files:

- item_categories.csv
- items.csv
- shops.csv
- sales_train_v2.csv
- test.csv

In addition, the following table shows the columns and describes each column briefly:

File	**Columns**	**Description**
item_categories.csv	item_category_name, item_category_id	name of item category, unique identifier of item category
Items.csv	item_name, item_id, item_category_id	name of item, unique identifier of a product, unique identifier of item category
Shops.csv	shop_name, shop_id	name of shop, unique identifier of a shop,
sales_train_v2.csv	Date, date_block_num,	date in format dd/mm/yyyy, a consecutive month number, used for convenience. January 2013 is 0, February 2013 is 1,..., October 2015 is 33,

	shop_id, item_id, item_price, item_cnt_day	unique identifier of a shop, unique identifier of a product, current price of an item, number of products sold.
Test.csv	ID, shop_id, item_id	an Id that represents a (Shop, Item) tuple within the test set, unique identifier of a shop, unique identifier of a product

The file „sales_train_v2.csv“contains the most relevant information for the predictions. It shows the historical sales data from January 2013 until October 2015 per day per item per shop and its price. That means, we have a sales history of 2 year and 10 months to learn from, in order to predict the amount of sales per item per shop for the next month (November 2015).
The list of shops and products slightly changes every month.

1.2 Problem Statement

In order to know which products C1 company needs to have in stock in which store, it is highly relevant to forecast the sales numbers for every product of the company's portfolio. The better the prediction the lower costs like running the storage, transportation costs for potentially unnecessary larger batch sizes of certain products, etc. Here an appropriate and well-adjusted prediction algorithm would help the company. One can apply many time series regressions to predict the sales forecasts for the next month. The competition requires a prediction for the following month after the data set ends.

Throughout the project we explored the data, needed to understand the specifics in them and invest huge effort in the preprocessing of the data for the actual prediction task (see details in chapter 2). We applied four different regression algorithms to the dataset in order to get a good comparison of which algorithm fits best to the given task.

1.3 Metrics

We will use the RMSE for the evaluation of the different predictions with the benchmark model but also with the submissions on Kaggle.

This is a quantifiable measure that can be used very well to compare predictions. The RMSE is generally used to compare differences between values (see: https://en.wikipedia.org/wiki/Root-mean-square_deviation) and thus a valid metrics for comparing the performance of the prediction models. In order to not deal with positive and negative values for the deviation we square the error measure. In addition, we want to be able to measure the deviation from the prediction in the same unit and hence the square root of the error to obtain a number that is directly interpretable. Since the values for our predictions are quite small, the absolute error will not be enough as error measure (source: https://www.quora.com/What-is-the-difference-between-squared-error-and-absolute-error).

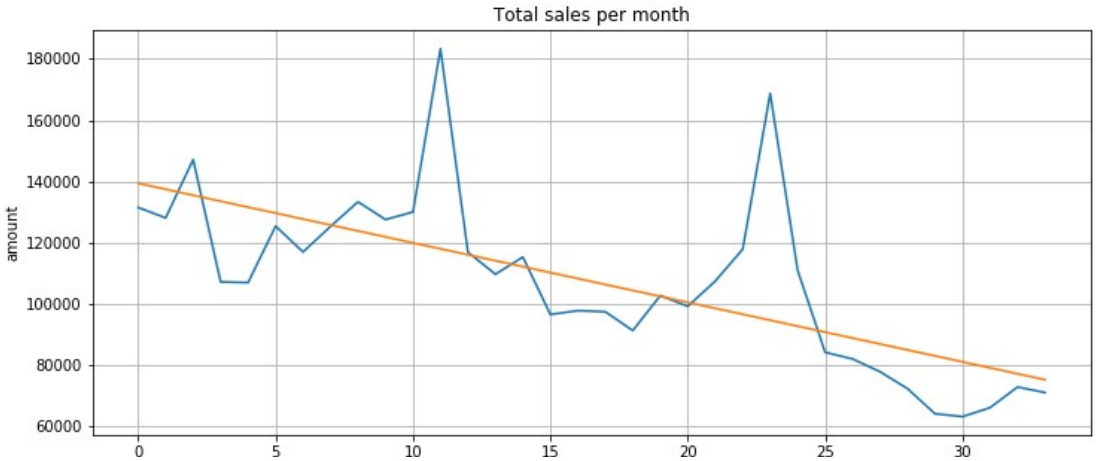
We use it as the only method since it is also used in the Kaggle competition and a widely spread measure for deviations between predicted and actual value.
A twostep evaluation will be conducted in our case in order to assess the performance of the benchmark model first and the various other prediction models second:
The benchmark model (linear regression) predictions will be compared with the Kaggle competition results. As a next step the RMSE will be used to evaluate the other prediction algorithms against this benchmark RMSE. The best prediction model will also be uploaded to the Kaggle platform to get an independent RMSE for the best model.

2. Analysis

This section is about analyzing the data sets provided by C1 Company. It is crucial to understand the data in detail and analyze what is given to us before any preprocessing for a prediction algorithm can be started.

2.1 Data Exploration

First, figure 1 shows the total sales per month between January 2013 and October 2015. The months are taken from the data_block_num variable of the sales file and 0 stands for january 2013, 1 for February 2013 etc. The y-axis represents the sum of items sold per month (indepent from the shop). We clearly see a falling trend in the sales numbers (see the orange trend line) and two peaks in month 12 and 24 (most likely christmas sales). The sales trend goes down from around 140.000 units to around 75.000 units in October 2015. Maybe this overall trend is also the reason for the company to ask for a good prediction algorithm. These are the facts



After the analysis of the other data sets: items.csv, item_category.csv, shops.csv and test.csv, one can say, that the all data sets except the test.csv file provide additional information on the data of the file sales_train_v2. The names of the shops, categories and items are written in Cyrillic letters due to the data being provided by a Russian company, but do not contribute to the prediction itself. All

the IDs are connected relate to names show that the data set consists of 22.170 distinct items, 84 item categories and 60 shops. We found that 13 shops are not open anymore, assuming no sales within the last 6 months. None of these shops are found in the test.csv file. This information is of relevance. There is one shop with id 36, which just opened one month before the prediction month, thus sales for this shop will be difficult due to the short history. Most likely the numbers are higher due to opening campaigns.

The data set test.csv consists of 214.200 rows and will be used for the testing of the trained algorithms and shows the combination of shop id and item id. That means that there are all items per shop listed which will be predicted for this project. Since it is best practice to not touch the test set in Machine Learning use cases, we will examine how many products – shop combinations are still valid, meaning the product will still be sold in the month of prediction. The data show, that 12.894 out of 22.170 products have not been sold in the last 6 months. These are around 58% of all items. Checking for these item numbers in the test set, we find 7.812 entries there, which will be predicted with 0 due to the assumption that these products expired. These are only 3.7% of the row but they exist. So for all shop – item combination we can assume 0 for the prediction.

	date	date_block_num	shop_id	item_id	item_price	item_cnt_day
0	02.01.2013	0	59	22154	999.00	1.0
1	03.01.2013	0	25	2552	899.00	1.0
2	05.01.2013	0	25	2552	899.00	-1.0
3	06.01.2013	0	25	2554	1709.05	1.0
4	15.01.2013	0	25	2555	1099.00	1.0

Table 1: file sales_train_v2 first rows

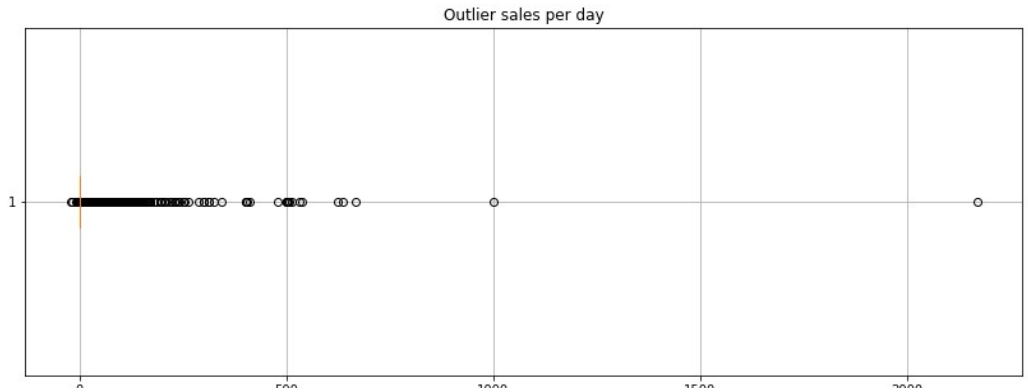
The most relevant data set is the sales_train_v2.csv file. Here the quantity of items sold per shop is listed on daily bases. It starts on 02.01.2013 and ends on the 31.10.2015. We have 2.935.849 entries. The column data_block_num is an id per month, which can be used to group items sold or categories per month easily. Thus, it is not necessary to aggregate the data on a monthly bases beforehand. The column item_count_day also contains negative values, which most likely stand for returns of and item for a certain price in a shop. The analysis revealed that 7.356 rows contain negative numbers. That means 7.356 times someone returned items in the shops. The item prices are also listed in the table but not of high relevance for the prediction of amounts. There can be variations in the price of the same product, which can be explained by temporary discounts or marketing campaigns.
Since this is the most relevant data set, we focus our analysis on these data.

The analysis of duplicate rows of all files revealed that there were only 6 duplicates in the sales_train_v2.csv file which can be ignored for our prediction. Probably the export of the data led to some minor issues. The duplicate rows just deal with an amount of 1 item sold per day and thus are of no relevance. Furthermore, we figured out that there are no invalid values in the data which would have needed to be cleaned.

For better analysis of the test data we set up a pivot table with rows being all items, columns being the months and the values the aggregated sum per month per item in all shops. This helps to identify “old products”, meaning products which were not sold anymore within the last six months and be

seen as deprecated. Thus if they still appear, in the test file, the can be handled with 0 and don‘t need to be fed into a model.

Figute 2 shows a boxplot of all sales amounts per day. On the x-axis we see the amount of the item sold in a day. Most of the data points are between 0 and around 400. We can see that there are just a few outlier greater or equal than 500 items sold in one day. These can be explained by special sales offers to specific customers, or a discount campaign around christmas. There are also negative sales amounts present which stand for returns of products.



The analysis of the shop names revealed interesting insights. We found out that the following shop names are very similar to each other:

- 00 and 57
- 01 and 58
- 10 and 11
- 23 and 24
- 39 and 40

Based on the Jaro Distance between 0.9 and 1.0 (where 1.0 stands for a complete match) and the information that among those the shops 00, 01, 11, 23, and 40 haven’t sold anything in the last six months, we assume that the shops just moved and thus will be condensed to one shop for the time series prediction. The pair 23 and 24 were open for four months in parallel and then shop 23 closed (or had to close) and thus will be treated as one shop.

The analysis of the test data set shows there are 111.404 pairs of shop_id-item_id which are in the training data set and also in the test data set. That means, for these combinations we can train a time series analysis on the training data. But these are just 52% of all shop-item pairs in the test set. Thus, we examined this data set further. We found that 15.246 items exist in the test set but not in the training set. This could be new items or even wrong information. The problem is, that for these items we do not have a training history thus a time series prediction will not work. Maybe a classification is possible- if or if not, this product will be sold. The last share of 87.550 shop-item pairs contain items where we only have item id information. In the prediction phase these three different categories need to be handled differently.

2.2 Algorithms and Techniques

The algorithms we used in the project are:

- Linear regression
- XGBoostRegressor
- AdaboostRegressor
- LSTM (Long short-term Memory)

We need to specify now that these time series predictions will only be applicable to the previously mentioned shop – item pairs with a history in the training data. Otherwise these approaches will not lead to meaningful results. Thus, there will be assumptions made for the other values, which will then be the same for all the algorithms. This might lead to no good results in the Kaggle competition ranking but provide a benchmark we can use to compare all models.

The most important preprocessing step before applying any time series algorithm is the merge of the tables test.csv and sales_train_v2.csv. This step ensures that all relevant shop – item pairs will be considered by the prediction algorithms.

Here we would like to give a short introduction on the algorithms we use in this project.
For good reproduction purposes and the ability to slightly fine tune later we start the standard parameter setup for all algorithms. The focus of this work is not tuning the algorithms as best as possible. Here we want to focus on the basic performance of the algorithms and compare those.

Linear Regression

Linear regression is a time series prediction method which is well research and widely used for presenting linear dependencies in data. In our case the independent variable is the month and the dependent variable is the number of items sold per shop. Combining these for each item sold in each shop we can predict the sales numbers per item per shop for the upcoming month. This is exactly what we want and will be presented in further detail in the implementation section. Due to linearity not always being the best approximation for time series (excluding trends, seasonality, etc.) we assume that this model does not perform well on the data set.

Linear Regression in its simplest form is represented in this way: $y = \beta_0 + \beta_1 + E$
Mathematical convention says that the variables involved in linear regression are x and y. Thus a linear model describes how y is calculated based on x. The model also contains an error, in the general form E, and it stands for the variability on y that cannot be explained by a linear relationship between x and y. Thus the simplest linear estimation regression function is $E(y) = \beta_0 + \beta_1 * x$. E(y) stands for the mean or expected value of y for a given value x. In our case we try to estimate the value of a shop – item combination for the predicted point in time by assuming a linear relationship between time and specific shop – item combination. The linear regression is used to show how variables are related to each other or to what extend there are associated. (source: https://www.thebalancesmb.com/what-is-simple-linear-regression-2296697).

Training the linear model works in this way, that based on historical months values per shop – item pair we train linear relationsship in the time series date. When this is done and we get a certain trend represented by the formula shown above, the we start the prediction phase. There the formula gives the best estimation based on previously seen data.

XGBoostRegressor

“XGBoost has become a widely used and really popular tool among Kaggle competitors and Data Scientists in industry, as it has been battle tested for production on large-scale problems. It is a highly flexible and versatile tool that can work through most regression […].” (source: https://www.kdnuggets.com/2017/10/xgboost-top-machine-learning-method-kaggle-explained.html)
Due to its popularity and ability to deal with time series regression problems we use this algorithm as a first „competitor“against the linear regression predictions.

How does the training work for a XGBoostRegressor. We take the following objective function:

$$\sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_k \Omega(f_k), f_k \in \mathcal{F}$$

Figure 1: objective function for XGBoost

were l is the sum over all training losses and the sum over all omega of f_k shows the complexity of the tree. The goal in the training phase is to optimize this function. The learning is done in an additive training way, called boosting. We start from a constant prediction and then add a new function for every time step. See the following figure from https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf):

$$\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}$$

Figure 2: training steps for XGBoost

here $\hat{y}$ is the formula for the model at training round t, and we always add a new function f_t to the function from the previous round $\hat{y}_i^{(t-1)}$.

We add a new decision tree in each iteration. Then in the beginning of each iteration we calculate the formula mentioned above and use statistics to greedily grow a tree f_t. (source: https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf)
The final structure of the XGBoost algorithm is a decision tree with an optimized objective function that we then use to predict the values for the month we need to predict.

See the information below to show the trained XGBRegressor:

```
Out[45]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bytree=1, gamma=0, importance_type='gain',
                      learning_rate=0.1, max_delta_step=0, max_depth=3,
                      min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
                      nthread=None, objective='reg:linear', random_state=0, reg_alpha=0,
                      reg_lambda=1, scale_pos_weight=1, seed=None, silent=True,
                      subsample=1)
```

Figure 3: XGB Regressor

The Analysis of the feature importance showed a higher F score for the item_id than for the shop_id, which means that the item_id has a bigger influence on the prediction results (see the following figure).

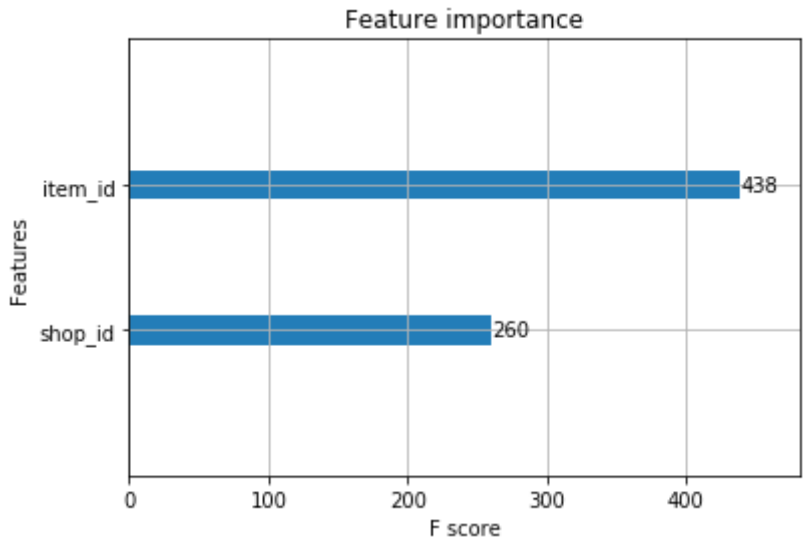


Figure 4: Feature Importance XGBoost

AdaboostRegressor

The AdaBoostRegressor is on algorithm from the family of ensemble methods. Ensemble methods use multiple learning algorithms and try to obtain better predictive performance (source: https://en.wikipedia.org/wiki/Ensemble_learning). “An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases” (source: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html) Due to our case being an easier case it would still be interesting to know, how this regressor performs on the prediction task. We assume no very good results, but in order to show a huge variety of algorithms trying to solve this prediction problem, we decided to apply this algorithm as well.

To

LSTM

LSTM stands for Long-short term memory and is a special group of Recurrent Neural Networks. One of their main features is, that they have a memory, comparable with humans who don't start thinking from scratch every second (source: https://colah.github.io/posts/2015-08-Understanding-LSTMs/). These models have the ability due to the built in loops (see figure below) the "remember" previous input. They are also often used for text classification or auto completion tasks but can as well be used for time series analysis. Trends and seasonality can also be "stored" in such a network. We do not want to dive to deep in the physics and implementation details of this model, but give a rough understanding of it.

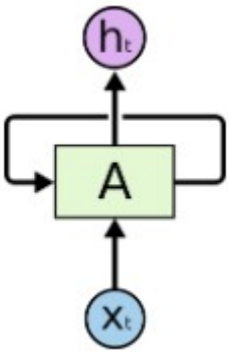


Figure 5: LSTM architecture

For a slightly better understand of the LSTM architecture see the figure below to look into detail of the A-section from figure 5.

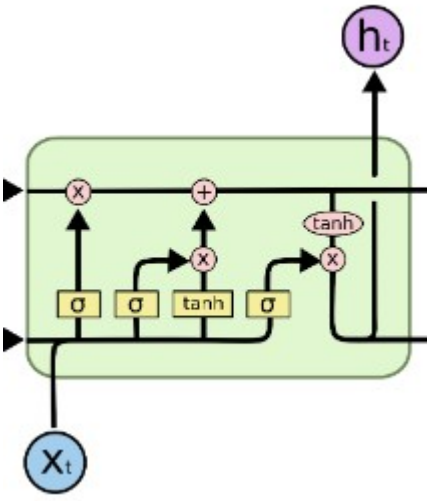


Figure 3: Details LSTM architecture

All the the yellow fields are neural network layers. The pink dots represent pointwise operations and the black arrows are vector transfers. X is the input to the network at a time step and h is the out at this timestep which is then fed in the next network layer. So what happens step by step from the left to the right of the layer. The first step is to decide which information will be thrown away. The first left layer, the sigmoid layer decides what to throw away, called the forget gate layer. It looks at

the previous time step and the new input x and and outputs a number between 0 and 1 for each number in cell state (see the black line in the top, which passes information though the network, if nothing will be changed). 1 is means “keep this” and 0 means “get rid of this information”. The next step is to decide which information is going to be stored in the cell state, consisting of 2 parts. First a sigmoid gate layer called the “input gate layer” (the second sigma box from the left) decides which values to update. Next the tanh function (3rd yellow box from the left) creates a vector of new candidate values that could be added to the state. Then the combination of both will create the update. Now the old cell state still needs to be updated. So the left x-pointwise operand connects the output of the forget gate layer with the cell state and the + pointwise operand adds the combination of input gate layer and tanh-function layer action to the cell state, to say which values of the cell state to update. In the end (most right sigmoid layer, x-pointwise operand and most right tanh-operand) we need to decide what to output. First we run the sigmoid layer on the previous input – current input which decides which part of the cell state to output, then the tanh-operation brings the values of the cell state between -1 and 1 and then multiply it with the output of the sigmoid gate to output what we want to. (source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/).

This explains the learing steps of the LSTM. An LSTM looks at all the 214.200 input variables of one month slice in our case inputs them into the layer and the the above mentioned process takes place. The prediction step then happens in the way that the trained model with all the cell state and layer settings gets a new input (the wanted month to predict) and then outputs numbers as best as it can based in the „inner settings“.

2.3 Benchmark

The linear regression, as the benchmark model in our case, returned a value of 1.24 after the upload to the kaggle competition. This RMSE will serve as a benchmark model for the other algorithms. Unfortunately, we don’t have access to the validation data set in the kaggle competition and thus will use the predicted values from the linear regression as benchmark for the other prediction algorithms. Knowing that the score of 1.24 is very far away from the true value, we will conduct the RMSE evaluation in two steps. First again the predictions of the linear regression and second by uploading the predictions to kaggle platforms. Both values be presented in section four of this report.
The initial assumption was to be better than the current mean of all RMSE values in the competition, which is at 1.09. We see that the benchmark model is (not surprisingly) worse than this value. The goal is to become better than this with at least one of the remaining three algorihms: XGBoost, Adaboost or LSTMs.

3. Methodology (3-5 pages)

This section consists of three parts: data preprocessing explained in more detail, the implementation and further explanations on it and last but not least a refinement section discussing the improvement steps throughout the project from the benchmark model to the best prediction model.

First, we needed load the data into data frames and a first look at the first five lines and the length of the files gave a good first overview over the content of the files. We realized that the file sales_train_v2.csv contains most of the information with a length of 2.935.849 entries representing 33 months of sales data on daily bases.

We realized that there were almost no duplicates in the data, except 6 rows in the sales_train_v2.csv, which can be ignored due to the huge amount of data there. It was very obvious that the data sets were very clean in terms of this aspect. In addition, there were not unknown values in the data sets. An analysis of all data sets revealed zero “NaN” values, which thus also did not need special attention.

Some first analysis on the overall sales numbers have been conducted to understand the trend of all products without considering the shops or product categories. Afterwards a different representations of the sales per item per month have been conducted to understand interruptions in sales or maybe the opening of a new shop. On top of that we realized that there are 7.356 entries with a negative number in the column “item_count_day”. Most likely these represent returns of products. They have been sold ones, but unfortunately for the firm were returned. The information in the data do not reveal for which reasons that happened.

To understand if there are outdated products, we assumed that if a product has not been sold in last six months of the training data, this product is old and any appearances of them in the test data will be handle with zero amount in the prediction. These were already 58% of the products. A quite high amount according to our estimation. 7.812 of them could be found in the test data file. This is not much when you see the number of rows of 214.200 in this file, but they exist.

An analysis of outliers in the sales per day revealed only a few very high numbers greater than 500 which will not be consider by the algorithms. Luckily there was no need to smoothen these values. We also looked at the amount of sales per shop per month to better understand. The implemented pivot table helped for a quick overview. We also found out, that there are 13 shops which have not sold any items in the last six months. This led to the assumption that they were closed at some point in time. The test file though showed that none of these shops was listed there anymore, so we can forget this information. Only one shop was new, which opened the month before the prediction. Here it is hard to really predict based on historical values. The predictions for that shop will be vague.

Analyzing the shop names revealed that there are very similar shops.

- Shops 0 and 57
- Shops 1 and 58
- Shops 10 and 11
- Shops 23 and 24
- Shops 39 and 40

In combination with the information that Shops 0, 1, 11, 23 and 40 did not sell anything, the assumptions are there that the old shops moved to the newer ones. This information are also considered in the prediction, due to old shops not selling anymore.

Analyzing the test set revealed that 214.200 predictions need to be made. There you find the shop and the item which is sold in that shop. The goal is to predict the quantity for each shop-item combination. We found out that there are 111.404 valid shop – item combinations with items that

have a certain “history” in the training data set. 15.246 item are in the test data set but not in the training data set. That means that we do not have any history for these items.

3.2 Implementation

As basic implementation and preprocessing for the algorithms we first created a pivot table from the sales_train_v2.csv file with item_id and shop_id as index, the months as columns and the values are the aggregated numbers of the item_cnt_day values per item per shop per month. This representation helps us to conduct certain analysis as described in the preprocessing step very fast and efficient.

Another important preprocessing aspect was then the merge step of the test.csv file and the sales_train_v2.csv file in order to create the right value pairs for the prediction later on. In this way we narrowed down the input from the training data file and focus only on the relevant shop – item pairs. The length of this file then was always the same for all four algorithms.

For all algorithms we used the default values for the parameters in order to find out how good they perform in their original form. The coding attached to this document will proof it. All prediction results were exported into a csv file which was then used to upload it to the Kaggle competition in order to get an independent RMSE score on the validation data set which was not available for us. Thus, the internal benchmark comparing RMSE of the predictions with the results of the linear regression are very inaccurate.

One specificity of the implementation of the LSTM algorithm needs to be mentioned and explained. Since this algorithm requires slices of the data set which are then fed into the network recurrently, we needed to add a third dimension to the train and test data. It was trained on 214.200 entries on a monthly basis and predicted all 214.200 values at once afte the training. It structure looks like this (see figure):

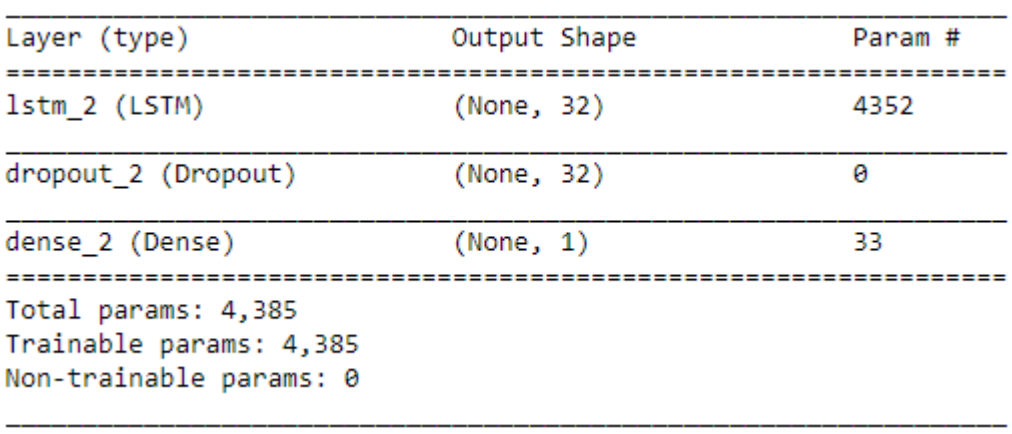


Figure 6: Structure LSTM

The lstm_2 layer is the “brain” of this network. The next layer dropout exists, because the network should not learn the training data by heart, but it should generalize well. A ratio of 20% in the implementation means that for every iteration 20 of the nodes in the hidden layer will be “turned off”. This leads to better generalization results. The dense layer in the end “is just a regular layer of

neurons in a neural network" (source: https://www.quora.com/In-Keras-what-is-a-dense-and-a-dropout-layer) where each layer is fully connected with the previous layer.

Implementing daily slices and weekly sliced have not been applied due to the complexity and analysis of time series predictions, which was a very new field. The benchmark of different mostly completely new algorithms led to a lot work assessing and understanding the concepts.

3.3 Refinement

The initial solution for this project was the linear regression prediction on the test data set. Attached to this document you find it in the predictions folder with the name "lin_reg_pred.csv". This is a really rough and first benchmark on the prediction and with every algorithm we implemented there was in improvement by applying their basic configuration.
The chain from bad to better is as follows: Linear Regression → XGBoost → AdaBosst → LSTM.
Even though this order has been chosen randomly it was surprising that with every algorithm we climbed up the Kaggle score ladder. Thus no refinements in the algorithms were conducts (was also not intended).
By trying to achieve better results in terms of RMSE towards the linear regression predictions, we refined the unit selection and the percentage of the dropout layer. Initially we started with 128 units as an initial value. The unit parameter stands for the dimension of the inner cells of the LSTM. The result led to a bigger RMSE and thus we tried to change the unit numbers, by applying 64 and 32 and 256. Among these values the RMSE for 32 was the best.
The refinement of the dropout value of 0.2 has not been conducted and based on example implementations from various implementation example we stayed with that.

For the Adaboostregressor we

All predictions can be found in the attached file collection in folder "predictions". The order is

1. Lin_reg_pred
2. Xgboost_pred
3. Adaboost_pred
4. Lstm_pred

Certain slight tunings on the input variable lead to slight improvements and here you only find the final versions of the predictions.

4. Results

This section discusses the results of the project and justifies the decisions made as far as possible. The focus should be on the model evaluation because this is what the company could use.

4.1 Model Evaluation and Validation

Table 1: RMSE results

Model	RMSE against benchmark	RMSE on kaggle
Linear regression	-	1.240
XGBoost	0.035	1.238
AdaBoost	1.114	1.210
LSTM	0.183	1.184

The table above shows the RMSE of different algorithms against the benchmark model and against the Kaggle competitors. The problem is, that the RMSE again the benchmark model cannot be trusted due to the fact, that this a prediction which has only been validated via the RMSE score but again real values that were available to us. Thus, these variation in the RMSE of the other algorithms against the linear regression model should not be considered much.

On the contrary the RMSE on Kaggle help for a better evaluation of the model's predictions. We see that even the best RMSE score is still way above the estimated mean of 1.09 in the proposal. The slight improvements in the RMSE score on Kaggle also show, that all models still do not present a very good prediction for all the 214.200 rows that exist in the test file.

In addition to that, almost all absolute values in the predictions were between 0 and 1 but rarely exactly 0 or 1. Summing up over all rows lead to reasonable sales amounts per month, which are comparable with the trend, but this wide spread amount more than 200.000 rows leads to very low numbers in per category. In addition, It is very rare that huge amounts of products are being sold in one store, thus mean or average predictions tend to be between 0 and 1 only.

4.2 Justification

Overall, we can say that the final results of the algorithms we tested were stronger than the benchmark model, considering the score on Kaggle. Still there is a lot of room for improvement and reflection on these results (see chapter 5 for details). One reason for the poor performance of these models can be the mount of data the models learned from. 33 months are not that much to identify such reliable numbers. The fluctuation in the products and shops also does not help for a good prediction. The low amounts per shop – item combination make it hard for a (complex) machine learning model to learn from it and predict accurate numbers in this granularity.

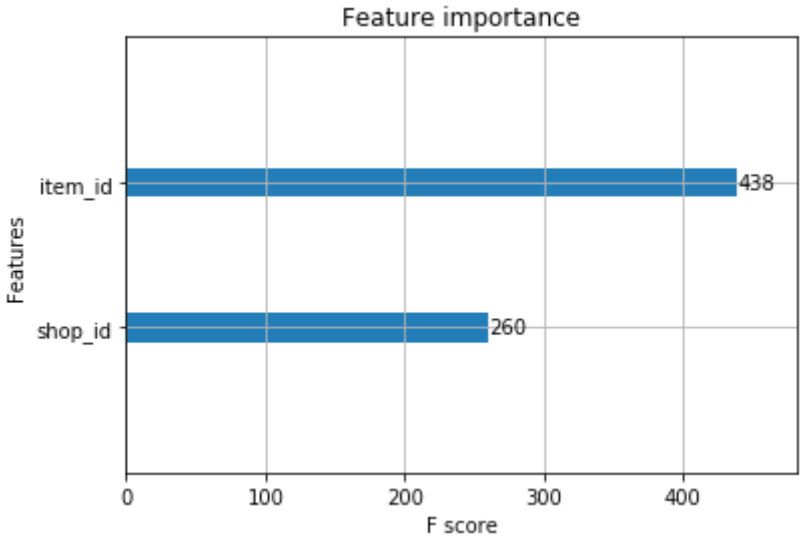
The final solution of the prediction of the LSTM reaches from -0.28 to 7.55 in absolute values. It is interesting to see, that the algorithm even predicts returns but based on the low numbers a significant relying on these numbers cannot be guaranteed.

5. Conclusion

Figure 7: learning curve (source: https://en.wikipedia.org/wiki/Learning_curve)

I can say that as we see in the figure above, that learning curve went steep out throughout this whole project. After all the input and analysis from various project before and especially now this end to end project, there is still a long way to go and many things to learn.

To provide one visualization of a a final model, the figure below shows the feature importance of the XGBoost model. We can see that the feature item_id is more relevant for the predictions than the shop_id, due to the F score being higher for this. The function "plot_importance" already ranks the the features based on the importance top down.



We can say that it matter more which item needs to be predicted than in which shop items are sold.

5.1 Reflection

This prediction project helped to learn and gain experience in another dimension of Machine Learning besides content of previous course projects. It made the executor of the project think again

from many more angles and there was no safety net while executing the task in this project. The Guidelines helped to not lose track completely, which can happen very fast in such a learning stage.

It is important to mention that “playing” around with many more algorithms, amongst other for this prediction use case, should be allow and tolerated. Unfortunately, the factor time plays an important role, especially concerning the budget of the firm in this case or the customer in general. One needs to say that the result of this project requires refinement on many dimensions: benchmarking, choice of algorithms, data set slicing, focus on details in the data by conducting further analysis, etc.

Overall it was difficult to start such an end to end project. It was not completely clear which details to focus on first. The data sets very clean, thus helped to focus on the most complex file sales_train_v2.csv. But within this file the right aggregation methods, slicing approaches, algorithms to apply to this data set etc. was not an easy task. I learned a lot in terms of how to tackle such a project in detail. The guidance helped a lot to not completely lose track but still a lot of practice is still needed.

Now it feels like getting the driver’s license – you learned how to drive but you become a good driver by driving a lot. I feel equipped for driving.

5.2 Improvements

Further improvements can be done by analyzing for example the product categories in more details. There has not been so much analysis yet on the importance of the categories in a certain store and the sales numbers for this category. This might be interesting as well as an even higher aggregation on a product level for example. The shop – item pairs led to mostly very small quantities in sales which, considering the huge diversity of these combinations, created a certain imbalance. The samples were big.

One could also argue that narrowing down the product portfolio can be an option. If certain products are not being sold well, the owner should consider stopping to sell them – especially knowing the overall trend of declined sales. In addition, the fluctuation of 13 shops closing/moving might also be worth considering more. Maybe less stores with more sales volume will lead to an upturn.

One could still fine tune the algorithms or even use classical time series methods like double exponential smoothing, which I did not know how to use. A rough research on well research statistical methods might also help to solve this problem, which even using complex machine learning models. I think that LSTMs or Recurrent Neural Networks in general are a bit too much and Adaboost or ensemble learning methods are to complex for this use case. I will keep on working on this project trying out and experiencing further time series predictions and the Kaggle competition will help measure the performance.