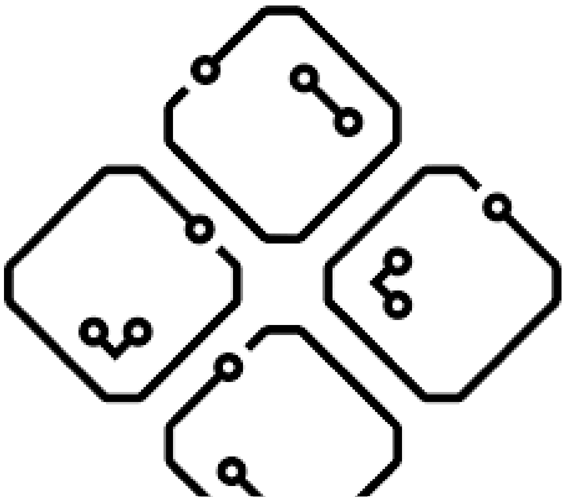
OPTIMUS PRIME

Sales forecasting



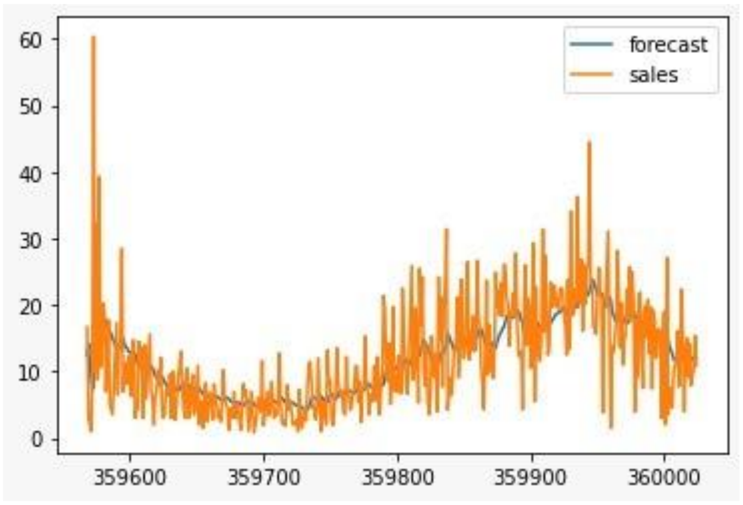
McKinsey
& Company

Índex

1. Problem description

The HackUPC sponsor, **McKinsey**, proposed a really interesting challenge for data scientists that consists in **predicting future sales** data and **extracting insights** from the processed dataset. The given dataset contains over 4.000.000 entries with some missing data. The missing data, identified with NaN values have been interpreted as 0, as, for example, if a particular product has not been sold in a shop at a defined date it can be interpreted that the product is out of stock. The considered attributes during the prediction are: SKU, geoCluster, price, sales.

2. First glance on the data



To know better the nature of the data we make the first exploratory analysis to see if the data follows any particular pattern. To make a clear visualization of the series the product with the most non-missing data was chosen, in particular: bananas. We adapted an ARIMA model with (p,d,q) = (1,1,1) and it results that does follow a pattern with a moving average, but the data has a very high variance. Having made this observation, we supposed that it would be difficult to obtain accurate results. Also, we discarded the option of using such time series models (like ARIMA) for our prediction, first of all because we would have to adapt one model for each product and location and second because for many of these shops we do not have such a long temporal series for us to study (many are full of NaNs or have few sales).

3. Our solutions

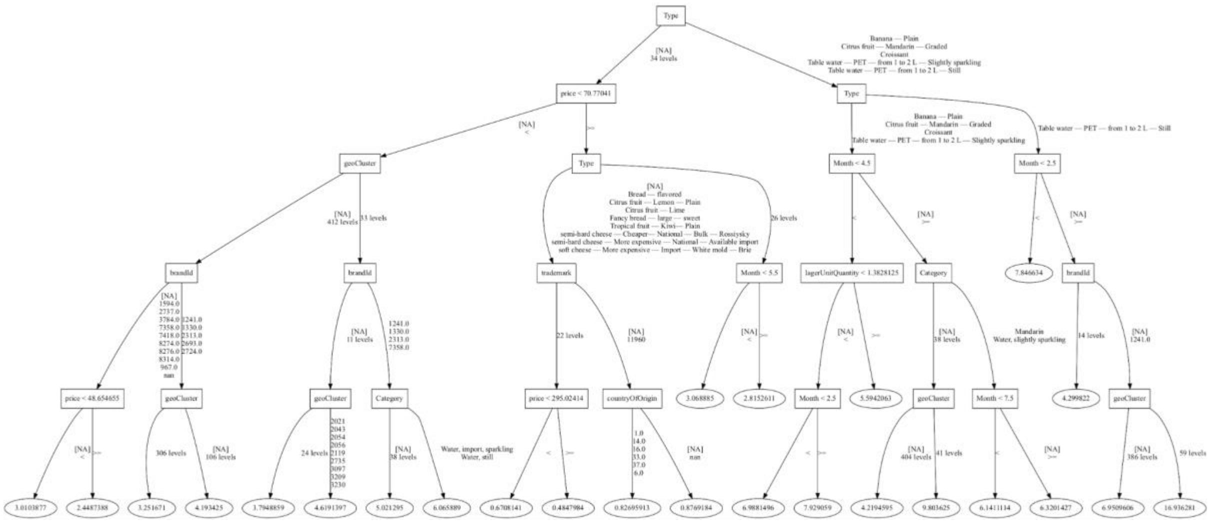
Having considered various options, the models that returned the best results were: Random Forest, Gradient Boosting and Multihead Transformer.

The main evaluation criteria of the performance of the model is the **Mean Square Error**. It is calculated with the real sales values and its square difference with the predicted value.

2.1 Random Forest and Gradient Boosting

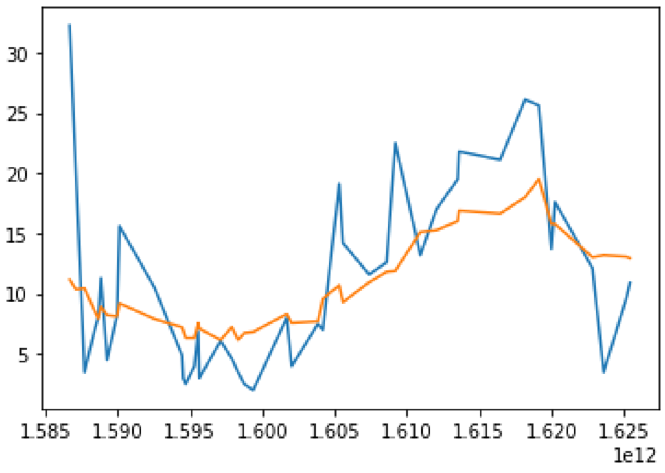
These two first models have been created as a first approach using machine learning techniques. The main advantage of this model is that it does not require a long training time since it does not have to learn the parameters. To create both of these models the library h2o has been used as it allows us to use both categorical and numerical variables.

Random forest has another big advantage which is the fact that it can be easily interpreted by visualizing its trees.



This tree is one of the ones used in our model in order to predict the value of the price. (This image is also available as a .png for better visualization). When a new value is received, the model chooses one branch or another depending on which are the variables of the sale we want to predict. In the first case, for example, the model chooses right if the category of the product is banana, citrus fruit, croissant, sparkling and still water and chooses left otherwise. The branches are created maximizing the separability between each side. This means that all the similar data will go down to the same branch. Once the data arrives at the bottom of the tree, one value is assigned to the variable "sales".
However, as explained above, we use a random forest, not a random tree, this means we repeat this process many times for different trees and average the results in order to obtain a more accurate estimate of the sales.

For each model we have partitioned our dataset into train and test and have obtained some measures describing its performance using the function .model_performance(). The hyperparameters of both models have been adapted in order to maximize the accuracy of the models.
In the case of the random forest model, the parameters that have been fixed are: `ntrees=100,max_depth=100` and `min_rows=50`.
Regarding the Boosting Estimator, we have fixed the parameters of `nfolds=50`.



As the data presents quite high variance, both models, equally, predict a similar value to the mean of the series, although they adapt to the overall trend evolution.

Random forest results:

```
MSE: 11.31
RMSE: 3.36
MAE: 1.67
RMSLE: 0.44
MRD: 11.31
```

Gradient Boosting results:

```
MSE: 11.49
RMSE: 3.39
MAE: 1.72
RMSLE: 0.45
MRD: 11.49
```

As it can be seen from the results extracted after the predicting, both models are really similar, however the Random Forest works out slightly better. **The Random Forest model predicted the data following the distribution of the actual sales discarting the null data and obtained very good results.**

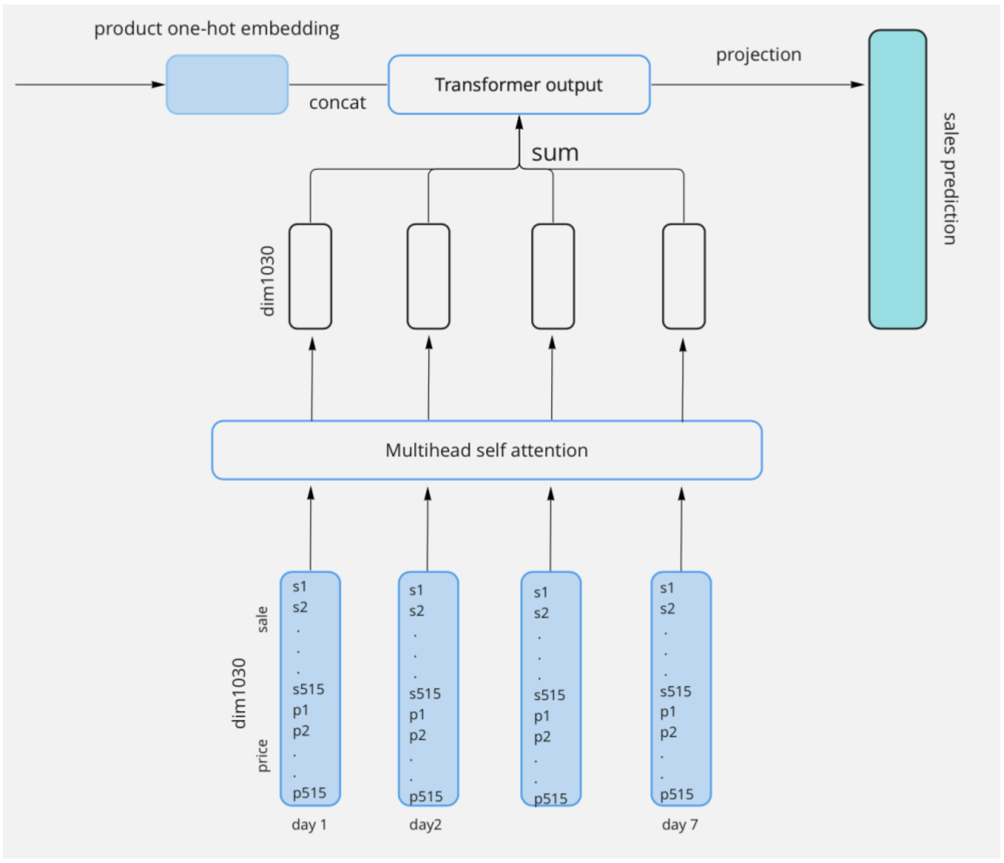
2.2 Transformer

What is a transformer?

Transformers are the latest deep learning models used for prediction with time series that adopt the mechanism of self-attention, differentially weighting the significance of each part of the input data. Due to this fact, a Transformer is a very interesting and promising model for solving the problem. The proposed model is a Multihead Transformer, this means that separate sections of the input can learn different aspects of the data. This particular model is executed with **5 heads, with batch size 2048 and with 3 training epochs.**

How to execute it?

In order to predict the future sales data with the trained model, the algorithm extracts information on sales of the 7 previous days of all the shops and products. Then, the product is specified. Having done this, the neural network will return a vector containing the prediction of all the sales for each location for the next day for that specific product.



There’s two reasons why to return a vector containing a prediction for each location. The first one is that by doing so, cases where almost all the sales of the product are null can be avoided. By returning a vector of size 515 it is less probable to return a set of zeros.The second one is that it will also be easier for the distributor to understand what amount of product he will have to send and thus he will be able to optimize the route to get to all the shops that will require sales.

Since the test document asked for specific products, location and day, we have had to look for the specific location in every vector returned.

To avoid constant filtering the data in big data frames with lots of missing values and save space we created a 4 dimensional matrix with all the needed data that is used as a lookup cheatsheet during the training. The first dimension is time, the second is product, the third geocluster, and the 4rth dimension contains the price and the sale, however the price was not used. This matrix allows saving space by only containing the index of the values in the training data. As the model trains with 7 previous days to the specified date lots of values are repeated. For this reason by saving only the 7 indexes instead of the 7 whole vectors we manage to save space.

In order to make a better prediction for the test values the model was defined to be **autoregressive.** An autoregressive model introduces the predicted values into the dataset from which it extracts the information on previous sales. As the model shows very nice results we can trust the inserted values.

The transformer model resulted to be the one that gives one of best results in terms of loss function.

4. Insights

The Random Forest showed that the most important attribute in the prediction is the **type** of the products as it is the one that branches the most. The next attribute is the **price** of the product, which is a contradictory insight with the Transformer model. These are followed by the **month** of the sale and the **geoCluster**.

During the analysis it was discovered that the data not only had missing values in the descriptive variables but also in the predicted, which really complicated the result extraction. We noticed that every time the sales were missing, the price was also missing. However, we interpreted that these instances indicated that the product was available in the store and did have a price but any unit of that product was sold. Working under this assumption, when training the transformer we have modified the training data. The NaNs of the sales have been changed to zeros and the price of the product has been set equal to the last price that product had in that same location.

Another observed insight is that when adding the price data apart from the sales and geoCluster to the model the obtained results are much worse. That was an unexpected result, however, we can conclude that there is no obvious relation between price and sale value.

It would be great to see at what cities the data was registered, as one alternative hypothesis was that the temperature would affect the sale level of some products (for example ice cream on a hot day) as the city code is not relevant without an actual name of the city.

5. Conclusions

The best results were obtained by two models: Random Forest and Transformer. The main difference between them is that the Transformer model uses complete data with lots of null values and as a consequence predicts values closer to zero. The Random Forest uses only not null data and predicts high values assuming that there is always a sale not taking into account the sequentiality of the data.

As a result we present two variations of the filled data: one with Transformer and one with Random Forest.

For further work it would be great to try another custom loss function and try the Random Forest model with zeros in its data.