# Udacity - Machine Learning Engineer Nanodegree

## Capstone Project: Walmart Store Sales Forecasting

Corentin Mouquet
July 11th, 2020

## I. Definition

### Project Overview

As a Supply Chain Manager in retail, I'm particularly interested in sales forecasting. Indeed, Supply Chain management is all about having the appropriate amount of products, in the right place, at the good time. Too little stock leads to high percentage of out of stock items, on the contrary too much stock leads to overstock: Two major Key Performance Indicators in Supply Chain. We easily understand the positive or negative impact for a company, in term of customer satisfaction, turnover and cashflow.

Better sales forecasts allow a better supply chain management, enabling to order or ship the good quantity of products, at the right time, to the good store.

In this project, I choose to work on sales forecasting for Walmart stores from a Kaggle dataset.

Here is some information about Walmart from Wikipedia:

*"Walmart is an American multinational retail corporation that operates a chain of hypermarkets, discount department stores, and grocery stores (…) As of April 30, 2020, Walmart has 11,484 stores and clubs in 27 countries (…) Walmart is the world's largest company by revenue, with US$514.405 billion, according to the Fortune Global 500 list in 2019"*

### Problem Statement

This is a time-series forecasting project. We get historical weekly sales data for 45 Walmart stores and for each of theirs departments. We will have to project sales for each store and each department the more precisely as possible.

Moreover we will have to deal with usual sales forecasting problems: missing data or short sales history for instance.

In particular, the dataset will be more detailed below, but we will create training and test time series from historical weekly sales. Test time series will be 36 weeks long. This is the same length than original test file from Kaggle competition, but we will be able to calculate ourselves the evaluation metrics and display predicted versus real weekly sales.

After analysing the dataset, the solution will be a trained model that we will use to forecast sales and get better results than our benchmark model given evaluation metrics

**Project design**

1) Load and explore the data

We will first load and save the files, pre-process the data, look at the correlation between features and global sales evolution.

2) Create training and test sets of time series

Then, we will create the appropriate inputs to train a chosen model:
- Benchmark models based on last year or last week sales
- AWS DeepAR model requires a JSON format to pass sales evolution, store and department number and chosen dynamical features
- Scikit-learn Random Forest Regressor model

3) Train the model
4) Deploy (for DeepAR) and get a prediction from the model
5) Evaluate the predictions with our metrics and select the final model

**Metrics**

We should first verify that our data sales evolutions are not following a random walk. Evaluation metrics in this case could seems pretty good but really misleading, as it will be by definition impossible to predict future outcomes. (https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424)

There are lots of evaluation metrics for time series forecasting, for instance:

MSE - Mean Square Error
RMSE - Root Mean Square Error
MAPE - Mean Absolute Percentage Error
SMAPE - Symmetric Mean Absolute Percentage Error
MAE - Mean Absolute Error…

We will use WMAE (weighted mean absolute error), which is the metric used to evaluate the competition by Kaggle.
This seems a good choice because a weight of 5 is set if the week is holiday week, 1 otherwise. We will see that the biggest sales weeks are holiday weeks and it makes sense for Walmart to expect particularly good sales predictions for these weeks.

$$\text{WMAE} = \frac{1}{\sum w_i} \sum_{i=1}^{n} w_i |y_i - \hat{y}_i|$$

where

- n is the number of rows
- $\hat{y}_i$ is the predicted sales
- $y_i$ is the actual sales
- $w_i$ are weights. w = 5 if the week is a holiday week, 1 otherwise

## II. Analysis

### Data Exploration

The data we will use for this project is from Kaggle competition: “Walmart Recruiting - Store Sales Forecasting” (https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting)

The dataset is composed of 3 csv files:

- A ***Train*** file containing weekly sales evolution for each store and department of the dataset from 2010-02-05 to 2012-11-01. With 45 stores and 81 departments the file contains 421570 lines

In this file we find the following information (this part directly from Kaggle)
*- Store: the store number*
*- Dept: the department number*
*- Date: the week*
*- Weekly_Sales: sales for the given department in the given store*
*- IsHoliday: whether the week is a special holiday week*

From this file, we will create our own training and test time series.
(We will then not use the Kaggle available Test file)

*First lines of Train file:*

| | Store | Dept | Date | Weekly_Sales | IsHoliday |
|---|---|---|---|---|---|
| **0** | 1 | 1 | 2010-02-05 | 24924.50 | False |
| **1** | 1 | 1 | 2010-02-12 | 46039.49 | True |
| **2** | 1 | 1 | 2010-02-19 | 41595.55 | False |
| **3** | 1 | 1 | 2010-02-26 | 19403.54 | False |
| **4** | 1 | 1 | 2010-03-05 | 21827.90 | False |

- A ***Features*** files containing 12 features and their evolution through time for each store. The file contains 8190 lines

In this file we find the following information (this part directly from Kaggle)
*- Store: the store number*
*- Date: the week*
*- Temperature: average temperature in the region*
*- Fuel_Price: cost of fuel in the region*
*- MarkDown1-5: anonymized data related to promotional markdowns that Walmart is running. MarkDown data is only available after Nov 2011, and is not available for all stores all the time. Any missing value is marked with an NA.*
*- CPI: the consumer price index*
*- Unemployment: the unemployment rate*
*- IsHoliday: whether the week is a special holiday week*

We will replace IsHoliday True or False (Boolean) values by 5 or 1 (integer) to compute more easily WMAE metrics

During data exploration we will look at correlation between these features and weekly sales. If some features have really low impact on sales, we will not take them into account to train our model. Moreover, if between them, some features are highly correlated we could only take one of them into account to make our model more efficient.

Finally, to make a future prediction, the only certain and reliable features will be Store, Date, IsHoliday and Markdown if Walmart know them in advance. Other features are not precisely predictable (or it's another project) so it will be better to avoid taking them into account.

*First lines of Features file:*

| | Store | Date | Temperature | Fuel_Price | MarkDown1 | MarkDown2 | MarkDown3 | MarkDown4 | MarkDown5 | CPI | Unemployment | IsHoliday |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2010-02-05 | 42.31 | 2.572 | NaN | NaN | NaN | NaN | NaN | 211.096358 | 8.106 | False |
| 1 | 1 | 2010-02-12 | 38.51 | 2.548 | NaN | NaN | NaN | NaN | NaN | 211.242170 | 8.106 | True |
| 2 | 1 | 2010-02-19 | 39.93 | 2.514 | NaN | NaN | NaN | NaN | NaN | 211.289143 | 8.106 | False |
| 3 | 1 | 2010-02-26 | 46.63 | 2.561 | NaN | NaN | NaN | NaN | NaN | 211.319643 | 8.106 | False |
| 4 | 1 | 2010-03-05 | 46.50 | 2.625 | NaN | NaN | NaN | NaN | NaN | 211.350143 | 8.106 | False |

We can see that all values of MarkDown in these five first lines are 'NaN'
If we count and compute the percentage of all missing values for MarkDown1 to 5 we get the result below.

*Percentage of missing values for MarkDown1 = 64.26*
*Percentage of missing values for MarkDown2 = 73.61*
*Percentage of missing values for MarkDown3 = 67.48*
*Percentage of missing values for MarkDown4 = 67.98*
*Percentage of missing values for MarkDown5 = 64.08*

We should be really cautious if we need to take these features into account.

- A ***Stores*** file containing the size and type (A, B or C) of each store. The file contains 45 lines

In the same way as with the features files, we will take into account size and type of store in our model only if this has a clear impact on weekly sales evolution.

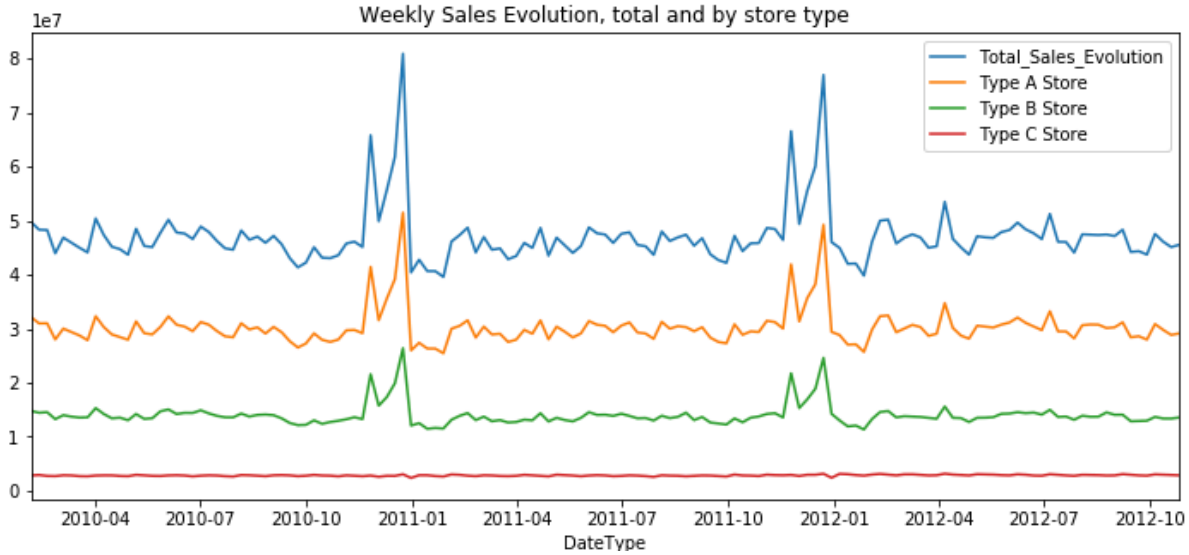
We will have to convert the type as integer if we want to use it in our models

*First lines of Stores file:*

| | Store | Type | Size |
|---|---|---|---|
| 0 | 1 | A | 151315 |
| 1 | 2 | A | 202307 |
| 2 | 3 | B | 37392 |
| 3 | 4 | A | 205863 |
| 4 | 5 | B | 34875 |

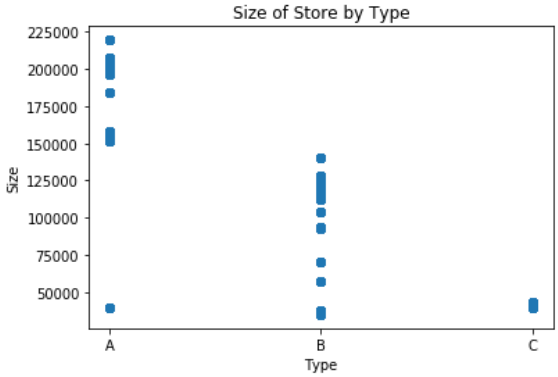
**Exploratory Visualization**

The aim of this project is to forecast weekly sales. We could first take a look at the total sales evolution through time in the dataset.



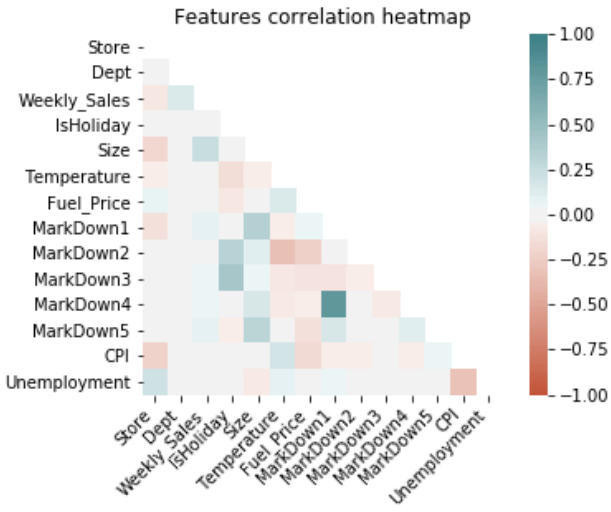

Total sales seems to follow a yearly pattern, with 2 noteworthy peaks during the last weeks of the year, corresponding to Thanksgiving and some weeks later Christmas holidays.

'Type C' stores represent a really small part of global sales. Indeed, there are only 6 'Type C' stores, which are among the smallest.



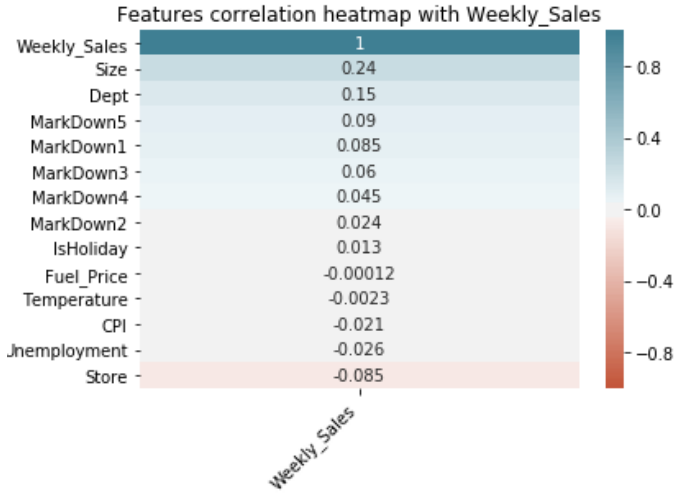

We get a Features file with 12 features. We should first look if there are some correlations between them, and more particularly with weekly sales. After merging Train, Stores and Features in one DataFrame we get this correlation heatmap.

Features seem poorly correlated in general except from MarkDown1 with MarkDown4.

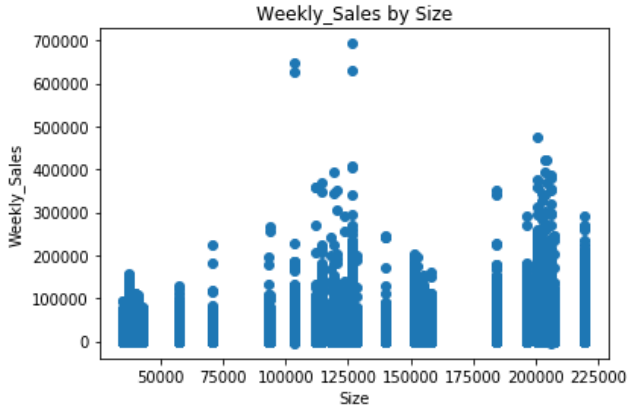


If we look more closely at Weekly_Sales correlations, they are the most correlated with the store's size and departments.

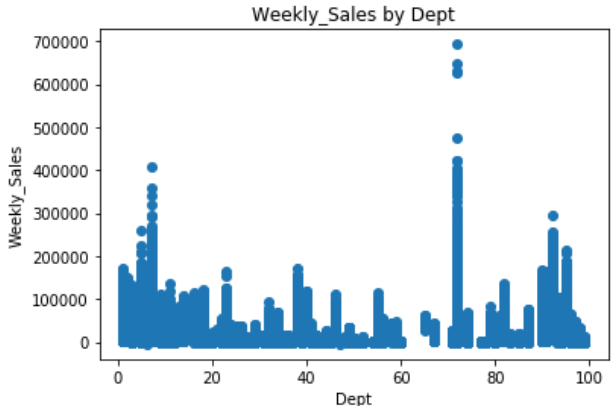
On the other hand, Weekly_Sales are the less correlated with store's number.




We will look at the sales repartition on these 3 features




There are outliers, but we see that in general the larger the store is, the more sales there are.




Sales are very uneven between different departments with some of them with no sales at all.


As expected, the store's number shows no correlation with weekly sales.

This exploratory visualization teaches us that:

- As features of the Features file show no particularly good correlations with weekly sales, they won't be especially useful (except from date, store number and isholiday which are already in the train file).
- On the contrary, weekly sales follow a clear yearly pattern and it will be a good idea to take the timeline into account as well as the store's department number in our model.

## Algorithms and Techniques

We are looking for a supervised learning algorithm to forecast time series. The algorithm should be able to take time series (dates linked with weekly sales) as input to train the model as well as some data characterizing the time series.

### 1 - Amazon Sagemaker DeepAR

Amazon Sagemaker DeepAR forecasting algorithm is one of them using recurrent neural networks. It takes times series as input, which can be associated with categorical features not evolving through time, like store and department number, as well as dynamic features like temperature or fuel price for instance. The algorithm returns forecasts for an asked prediction length.

One interesting point of DeepAR is that time series might contain missing values. We already see that some department have no sales at all and it's possible that some weekly sales for a given store/department contain some missing values.
Furthermore, DeepAR easily take recurrent pattern into account. In our case we get weekly sales with yearly pattern.

All documentation and more details about how Deepar works is available here:
https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html

### 2 - Scikit Learn Random Forest Regressor

Scikit Learn Random Forest Regressor Model is also a supervised learning algorithm that can perform regression task with help of decision trees. As shown in this illustration, random forest combines the predictions of a large number of decision trees, enabling better results than any individual decision tree.

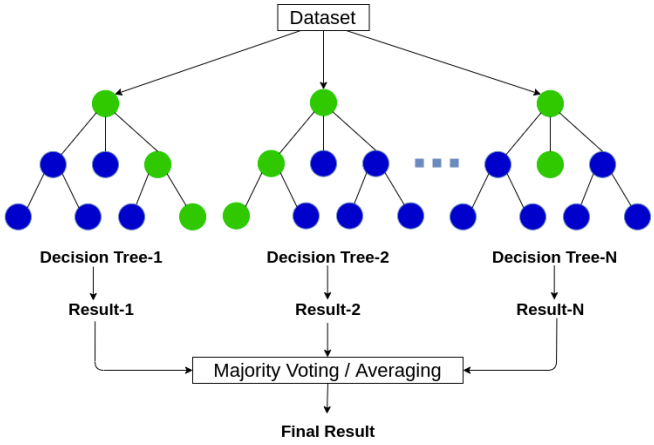



*Image source and explanation about random forest: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/*

In our project each decision tree will be trained with data like week number, department number and with weekly sales as output. It seems to be another good algorithm for our project. The main advantages are that the implementation of the model is relatively easy even with large dataset and should lead to good results.

Here is some well explained information about Decision tree:
https://clearpredictions.com/Home/DecisionTree

Here is the documentation of the scikit learn algorithm we will be using:
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

**Benchmark**

For our benchmark model, we will start with a really naïve model where predicted sales will be the average of all sales, grouped by store and department and compute the WMAE metrics as define previously on the 36 last weeks of data. We get this result.

Predicted sales = all weeks average sales:  WMAE = 2334,34

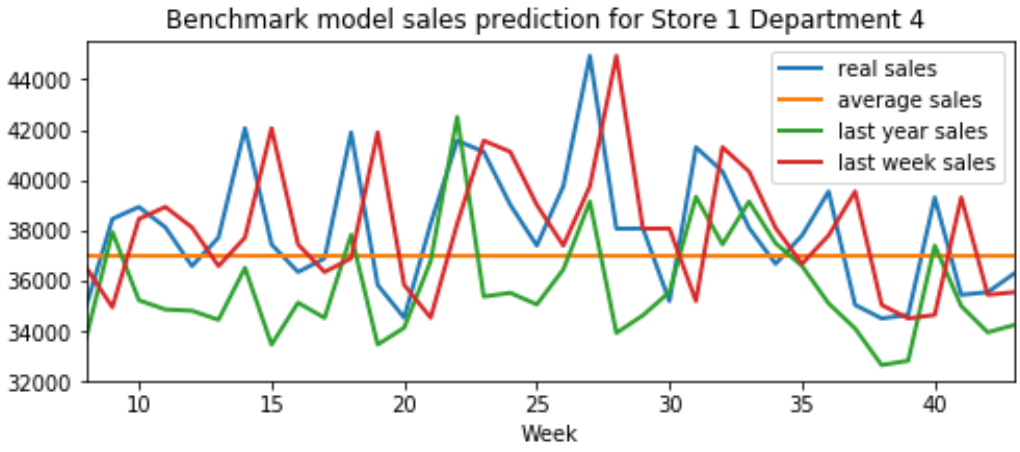
Next, we take more competitive benchmark models, with predicted sales equivalent to last week sales and a last model with predicted sales equivalent to last year sales.

Predicted sales = last week sales:  WMAE = 1685,56

Predicted sales = last year sales:  WMAE = 1789,95

These two models show significant improvements. Indeed we saw previously that weekly sales are relatively stable most of the time (except from lasts weeks of the year) explaining good result for the model with predicted sales equivalent to last week sales.
Moreover weekly sales follow a yearly pattern, explaining a nearly as good result with predicted sales equal to last year sales.

Here is an illustration of our 3 benchmark-models in comparison with real weekly sales for the 36 last weeks of our dataset and for one selected store and department.



# III. Methodology

### Data Preprocessing

There are common parts but we will need to preprocess the data differently for DeepAR and Random Forest algorithm.

- Same pre-processing part

From the train file, we will first replace missing values and negative weekly sales by 0.

To compute our WMAE evaluation metrics easily and in the same way as Kaggle's competition, we will set "IsHoliday" column to 5 if True, 1 otherwise.

- DeepAR preprocessing

As DeepAR is specific to time series forecasting, we will first set the date column as the index of the dataframe and then remove the column.

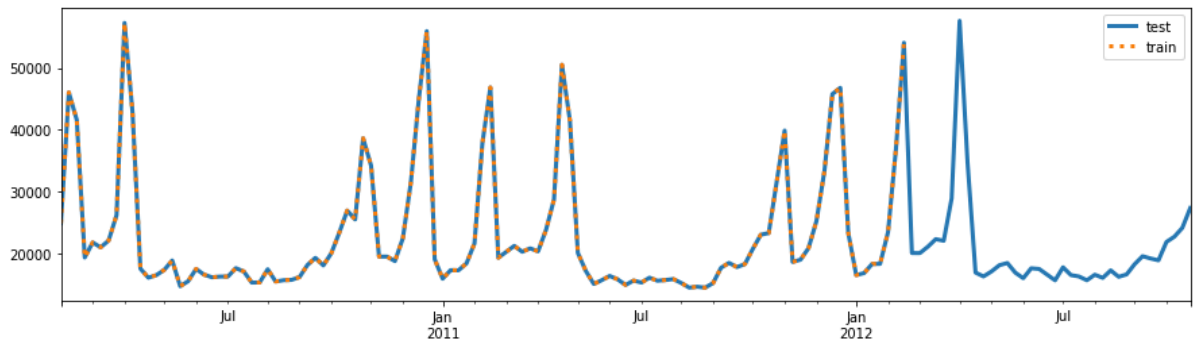
Next we will have to convert Store and Department number to a format convenient with DeepAR, to use them as categorical features. Amazon documentation explains that *categorical features must be encoded as a 0-based sequence of positive integers*. There are missing Department number, and the list start with 1 instead of 0. We convert Store and Dept a list of 45 Stores and another of 81 Departments, each starting by 0.

Data is then ready for us to create complete time series, using Store and Department as categorical features, "IsHoliday" as dynamical feature, and weekly sales evolution as target.

I choose to append the store and department number at the end of each time series to get access to them more quickly later. Of course we will make sure to remove them when we will display the time series or pass them to train the model.

After that, we create training time series by the same process as previously but without taking into account the 36 last weeks of sales.

Example of first test and train time series:



Finally we have to create train and test JSON objects, composed of as much JSON lines as there are time series. Each JSON lines contains start time, the list of weekly sales (target), the store and department number (cat) and the list of IsHoliday value during the time series (dynamical_feat)

Example of JSON Lines format:
{"start": "1999-01-30 00:00:00", "target": [2.0, 1.0], "cat": [1, 4], "dynamic_feat": [[1.3, 0.4]]}

To finish, we save the train.json and test.json file locally then upload them to AWS S3.

- Random Forest preprocessing

Random Forest model is not specific to time series forecasting but we will make it train with time data as input. We will then convert the “Date” column into a “Year” and “Week” columns.

**Implementation**

- DeepAR implementation

Train.json and test.json files were uploaded to S3. We now instantiate a DeepAR estimator and set up hyper parameters: we select a weekly frequency and a context length the same size as prediction length, that’s to say 36 weeks

We launch a training job, asking DeepAR estimator to fit the model with train file

We deploy the trained model, and ask it to predict the 36 last weeks of the dataset for each time series. For this, the JSON lines provided to the estimator will contain only the training weekly sales, but all dynamical features (36 more values than weekly sales). The estimator will understand and forecast the 36 missing weekly sales values. We then decode and save the prediction.

We will next display and compute the evaluation metrics

- Random Forest implementation

With Random Forest, the implementation is much faster. We first create a global Train DataFrame with all data except the 36 last weeks, and a global Test DataFrame with only the 36 last weeks.

With a loop, we then create sub train and test DataFrames for each distinct store/department couple.
Finally we split these 2 DataFrames in two to get xtrain and xtest (containing all data of the store/department couple except weekly sales) and ytrain and ytest (containing only weekly sales for the given store/department couple)

We then use the Scikit Learn Random Forest Regressor model to fit xtrain with ytrain and ask the model a prediction for xtest. (Real sales being ytest)

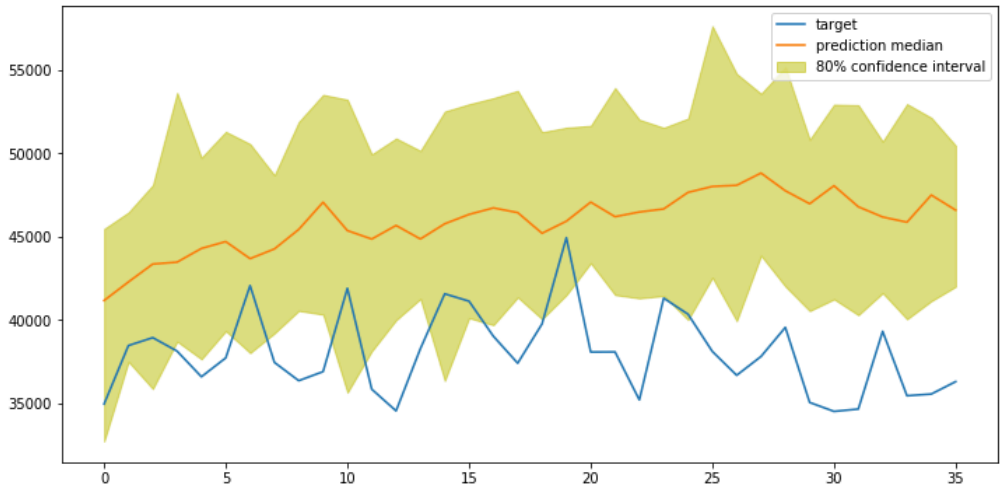
We will next display and compute the evaluation metrics

**Refinement**

- DeepAR Refinement

*Notebook: DeepAR_2stores_train.ipynb*

With our first implementation, the results were really disappointing. When displayed, our forecasted prediction median often doesn't take yearly pattern sufficiently into account and sometimes seems uncorrelated with previous years real sales. During improvement steps, we will look at a same example for Store 1 Department 4 prediction for comparison. Here is the prediction made by our model in first implementation.



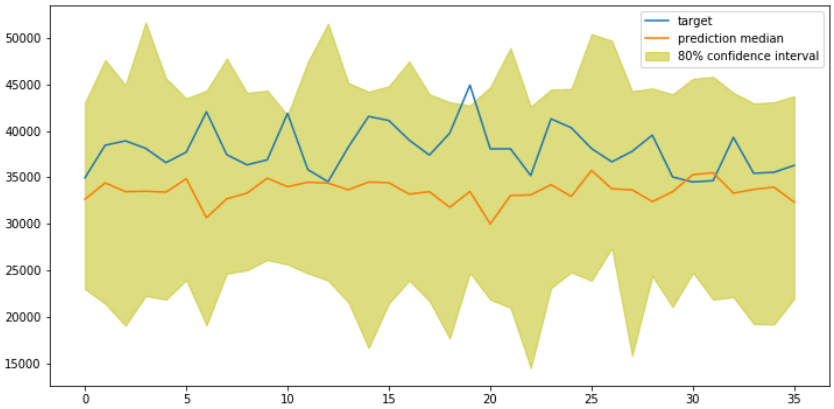

Our evaluation metric has logically pretty bad results: WMAE = 5395,34

*Notebook: DeepAR_2stores_train_2features.ipynb*

I try to improve this result by making some changes in the parameters of the model and in the features:

1. DeepAR algorithm automatically tries to minimize his own metrics, which are the root mean square error or RMSE and the mean quantile loss. As we define our own evaluation metric (WMAE) and modify the IsHoliday values, I set the value back to 1 or 0 (instead of 5 or 1) to avoid disrupting too much the model.
2. I try to set one other dynamical features and select MarkDown1 which was the most correlated among other dynamical features and also correlated with MarkDown4

With these two changes I get a little improvement with WMAE decreasing to 4177 and the prediction approaching the real sales but still less performing than all our benchmark models




After trying these different changes with DeepAR we get results really far from expectations and I check the following DeepAR best practices:
*https://docs.aws.amazon.com/sagemaker/latest/dg/deepar.html#deepar_best_practices*

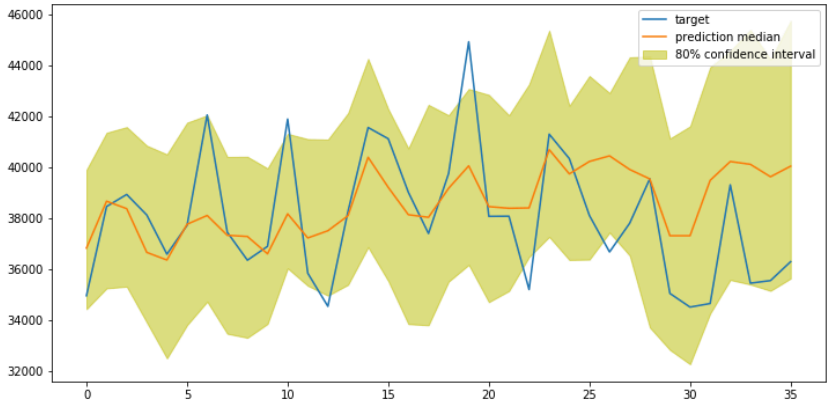
- Provide entire time series (except when splitting for training) to the model
- Create training time series by removing the last prediction points
- Avoid using prediction length > 400
- Set context length to the same value as prediction length

All these previous points were satisfied but a last point of the best practices recommends training the model with as many time series as possible. DeepAR algorithm is especially efficient with hundreds of related time series.

During my first attempts with DeepAR, I indeed get error message “Your invocation timed out while waiting for a response from container model” and as a consequence try to train my model with less data. I was training and asking predictions for only two stores and this could be the reason of bad performances.

So I decided to train my model again with previously seen parameters, but now with all the dataset, and the results were better. Nevertheless, I only achieve to get predictions for 4 stores otherwise get the same error message, when asking for predictions but we can already compute and display results for the first 4 stores.

*Notebook: DeepAR_Final_Model.ipynb*



The prediction now seems really closer to real weekly sales, and WMAE decrease to 2283,4 for the 4 stores. This result is more acceptable as it is better than the average benchmark model, but it still doesn't beat the last year and last week benchmark models.

- Random Forest Refinement

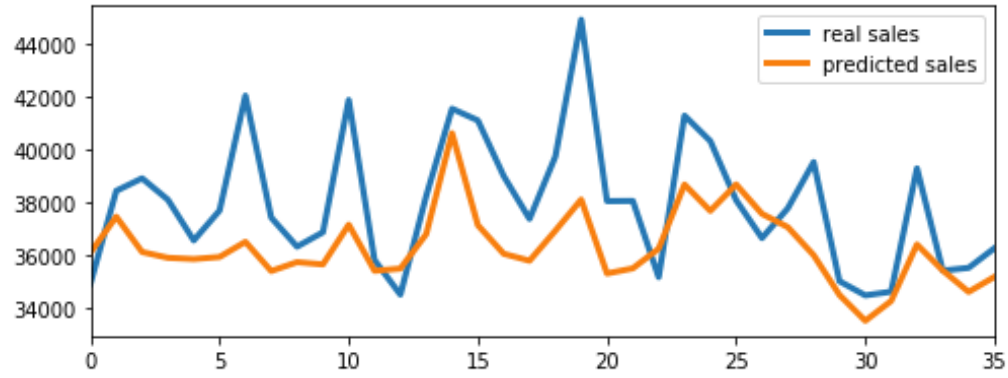
After asking our trained model to predict value for xtest, we plot the predictions and compute WMAE.

Results are better than DeepAR and a little better than our last week and last year benchmark models.

We get a global WMAE of 1555,32 (7,7% improvement from our best benchmark model which had a WMAE of 1685)

Here is the forecast for store 1 department 4 we saw earlier in our benchmark model and with DeepAR model.

```
Weighted mean absolute error (WMAE) for Store 1 Dept 4 = 2091.669527500012
```




To improve this model, I try to add more features than just IsHoliday column but we get a little worse results when adding the MarkDown1 feature, it leads to a global WMAE of 1566,97
Adding the Size feature leads to a global WMAE of 1555,19. This is slightly better than the first model, but with results so close I think better to select the model with less data.

The best solution we get this far is then this Random Forest Regressor Model, trained with previous weekly sales, store, department, IsHoliday, Week and Year

Here are the first lines of the dataframe serving as input to train our model:

| | Store | Dept | Weekly_Sales | IsHoliday | Week | Year |
|---|---|---|---|---|---|---|
| **0** | 1 | 1 | 24924.50 | 1 | 5 | 2010 |
| **1** | 1 | 1 | 46039.49 | 5 | 6 | 2010 |
| **2** | 1 | 1 | 41595.55 | 1 | 7 | 2010 |
| **3** | 1 | 1 | 19403.54 | 1 | 8 | 2010 |
| **4** | 1 | 1 | 21827.90 | 1 | 9 | 2010 |

## IV. Results

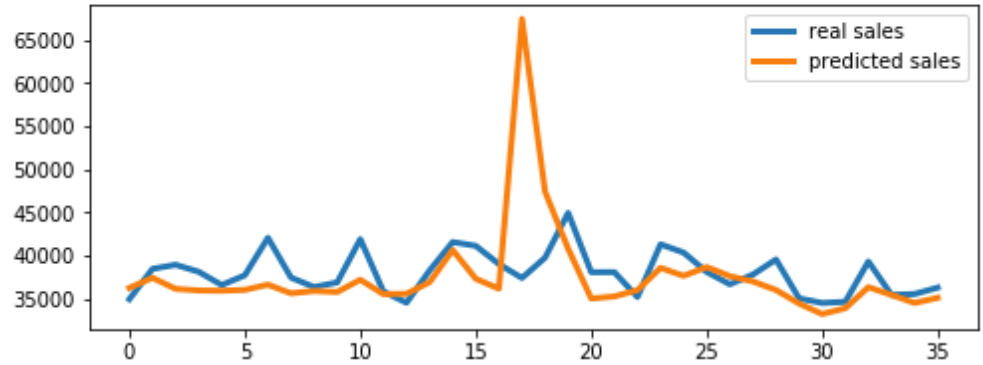
### Model Evaluation and Validation

Our final model is this Random Forest Regressor Model, trained with previous weekly sales, store, department, IsHoliday, Week and Year. We don't make use of the features file provided, as they were no clear correlations with weekly sales. We only keep IsHoliday to compute our evaluation metric. The model meets our expectations to predict 36 last sales weeks of the dataset and is better than the best benchmark model.

To test the robustness of our model we make some changes in the data used to train the model.

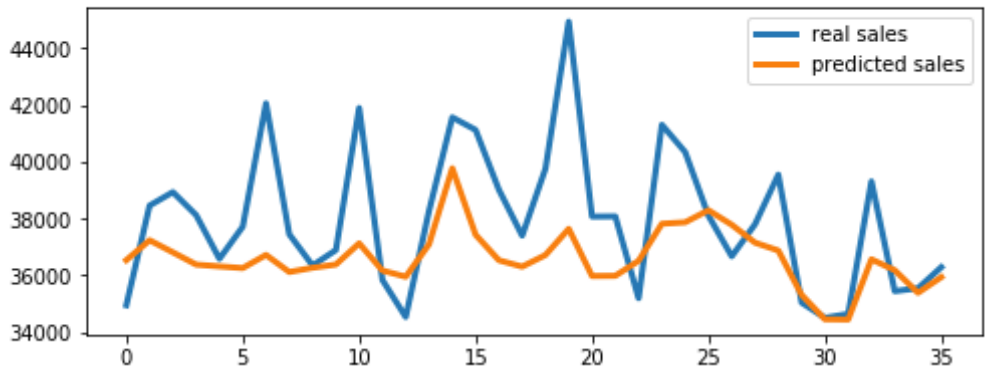
We change manually one week's sales to about 10 times higher than his normal value .The model takes this value into account and we can see the prediction becoming much higher for this particular week.

```
Weighted mean absolute error (WMAE) for Store 1 Dept 4 = 2877.6552150000084
```



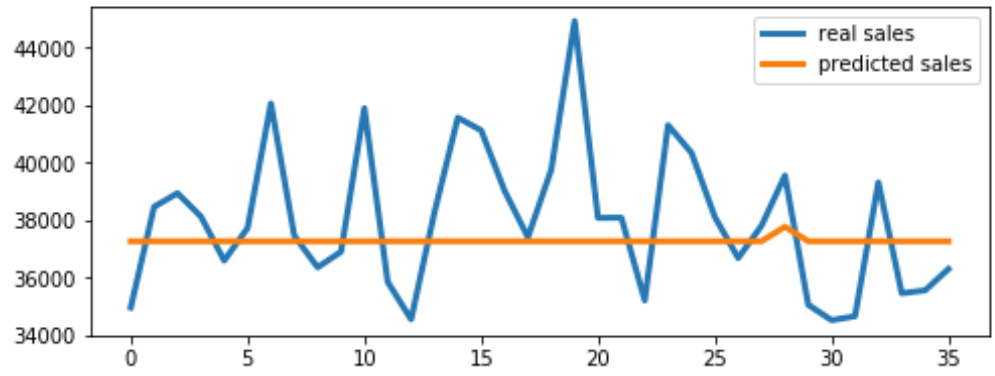

Next we set to zero all the value of 2010 sales for this Store 1 department 4. Predictions stay coherent and don't change too much. In this particular case, we even get better WMAE than with the data of 2010

```
Weighted mean absolute error (WMAE) for Store 1 Dept 4 = 1850.4405175000118
```




Finally we set sales of 2010 and 2011 to zero. The resulting prediction is really poor but take previous weeks of 2012 into account to get a prediction in the average as shown in the next picture.

```
Weighted mean absolute error (WMAE) for Store 1 Dept 4 = 1982.9615125000078
```




Our model is operational but this warns us to make sure there are no wrong or incoherent data leading to wrong forecasts.

Predictions are linked directly with each store/department historical data. If no or few data, we will get poor or bad forecasts.

**Justification**

Here is the summary table of results

| Model | Evaluation metrics: WMAE (weighted mean absolute error) |
|---|---|
| Benchmark 1: All weeks average | 2334,34 |
| Benchmark 2: forecast = Last week sales | 1685,56 |
| Benchmark 3: forecast = Last year sales | 1789,95 |
| AWS Sagemaker DeepAR Final Model | 2283,4 |
| **Scikit Learn RandomForestRegressor** | **1555,32** |

As we saw in exploratory visualization, Sales follow a yearly pattern at the end of the year and otherwise are relatively stable. Our benchmarks model based on last week and last year sales were already providing relatively good results.

A WMAE decreasing of 7,7% in comparison to Benchmark model 2 is then a significant and sufficient improvement.

## V. Conclusion

### Reflection

In this project we had to create a model to forecast sales for 45 Walmart stores and for each of theirs departments. We get historical weekly sales data that we split into train and test sets, and try to make our forecast for the 36 last weeks the more accurate as possible.

After exploratory visualization we understand that all the features were not much correlated with weekly sales evolution and I found that part interesting because I intuitively though it would be useful to take as much data as possible.

After selecting 2 algorithms efficient for time series forecasting, we define as well benchmark models and pre-process the data to make it usable as input by our models.

We train them and then ask predictions before computing our evaluation metric.

I found it interesting to see impacts in results by changing some of the parameters. However it may take longer than expected because each time we have to train (and deploy for DeepAR) before getting results.

We finally select our best model: The Random Forest Regressor and make some changes in the input data to see if the reactions of the model were suitable.

### Improvement

To further improve this model, and even if I already try a lot of parameter changes, it might be possible to use a hyperparameter-tuning job.

Moreover it could be possible to try other kind of time series forecasting algorithms like Seasonal Autoregressive Integrated Moving-Average (SARIMA) or Vector Autoregressive Model (VAR) for instance and my next steps will definitely be to learn more and look at examples of these methods.