The document provides an analysis work on some retail data, with the main goal being to explore 2 models for the target value “Sales” and predict the sales for the following week (week 98).
The document is organized as follows:

1. A first section presents the available tables and explores the data.
2. A second section details the construction of 2 predictive models, based on ARIMA and Random Forest, and their evaluation.
3. A third section suggests directions to further improve the modeling. This latter part evaluates also the relevance of a method from a research paper.
4. A last section presents the results from a [late and quick] LSTM based model.

**Files package description:**

| File Name | Type | Description |
|---|---|---|
| RandomForest_features.csv | CSV | Features for RF model,<br>exported from SQL query |
| Week_serie.csv | CSV | Weekly sum of sales for ARIMA,<br>exported from SQL query |
| Features_extr_for_RF.sql | SQL | Query extracting some features for RF model |
| Quantity0.sql | SQL | Queries highlighting Quantity=0 values in the table |
| Sum_of_sales.sql | SQL | Queries providing weekly sum of sales |
| Preliminary_question.sql | SQL | Queries solving preliminary questions |
| Lstm.py | Python | LSTM model for weekly sum of sales |
| Preliminary_questions.py | Python | Python script to solve preliminary questions<br>(as double check) |
| Arima.R | R | ARIMA model in R |
| RandomForest_modeImpute.R | R | RF model in R, with mode based imputation |
| RandomForest_rfImpute.R | R | RF model in R, with rfImpute function based imputation |

# Contents

# 1. Data Exploration

## 1.1. Data Source description

The dataset consists of 3 CSV tables:

- The main one - *TRANSACTION_DATA* - provides the registered sales transactions from a retail comprising several stores.
- The *DEMOGRAPHIC* table contains the data specific to the customers.
- The *PRODUCT* table contains the data specific to the products available from the store.

TRANSACTION_DATA and DEMOGRAPHIC can be linked (joined) by the PRODUCT_ID.
TRANSACTION_DATA and PRODCUT can be linked (joined) by the HOUSEHOLD_KEY.

At some point, the 3 tables will be used in the study. However, let us first analyze the main one TRANSACTION_DATA, as it contains the data of interest for our predictive modeling: **Sales**.
Beside the SALES_VALUE field, we can find several other related to coupons (COUPON_MATCH_DISC, COUPON_DISC) and discount (RETAIL_DISC). After reading carefully the description, the SALES_VALUE is the real amount received by the retailer. Therefore, we can safely use this field as target variable and disregard the coupon and discounts related fields.

Notice: as opposed to what is written in the description (slide 6), the dataset finishes at week 97, not at week 102.

## 1.2. Preliminary questions

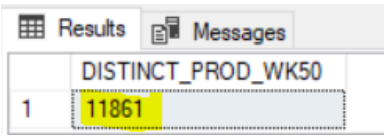
As the 3 CSVs come from a relational database, it makes sense to explore the data and at least answer the preliminary questions using SQL server.
Upon import, it is possible (and recommended) to check the data types and change them if required.
I changed the types according to the table below:

| Column Name | Data Type |
|---|---|
| household_key | int |
| BASKET_ID | bigint |
| DAY | int |
| PRODUCT_ID | int |
| QUANTITY | int |
| SALES_VALUE | float |
| STORE_ID | int |
| RETAIL_DISC | float |
| TRANS_TIME | int |
| WEEK_NO | int |
| COUPON_DISC | float |
| COUPON_MATCH_DISC | float |

**"How many different products are offered by the store in week 50?"**

As I found the question a bit ambiguous, I decided to provide 2 answers resulting from 2 interpretations of the question:

1. First interpretation (most probable to me): How many **DISTINCT products** are offered by the store, (purchased by the customers, according to the data), **during the week 50 ONLY**




| | DISTINCT_PROD_WK50 |
|---|---|
| 1 | 11861 |

2. Second interpretation: How many **DISTINCT products** are overall available from the store AT week 50, **ASSUMING that** the number of product references grows every day in the store. During week 50, the customers did buy only a subset of the complete product list.

| | DISTINCT_PROD_UNTIL_WK50 |
|---|---|
| 1 | 63651 |

These results are available in SQL and Python from the following files:

- *Preliminary_questions.sql*
- *Preliminary_questions.py*

**"What is the best-selling product overall?"**

This question is answered using SQL queries:

A first query, using only TRANSACTION_DATA gives the best-selling PRODUCT_ID in descending order

| | PRODUCT_ID | TOTAL_SOLD |
|---|---|---|
| 1 | 6534178 | 204770364 |
| 2 | 6533889 | 17390532 |
| 3 | 6534166 | 13242006 |
| 4 | 6544236 | 2609757 |
| 5 | 1404121 | 1667186 |
| 6 | 397896 | 1281329 |
| 7 | 1426702 | 453293 |
| 8 | 5703832 | 430940 |

A second and more elaborated query, joining TRANSCATION_DATA and PRODUCT tables gives the product description in addition.

| | PRODUCT_ID | TOTAL_SOLD | PRODUCT_ID | MANUFACTURER | DEPARTMENT | BRAND | COMMODITY_DESC | SUB_COMMODITY_DESC |
|---|---|---|---|---|---|---|---|---|
| 1 | 6534178 | 204770364 | 6534178 | 69 | KIOSK-GAS | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |
| 2 | 6533889 | 17390532 | 6533889 | 69 | MISC SALES TRAN | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |
| 3 | 6534166 | 13242006 | 6534166 | 69 | MISC SALES TRAN | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |
| 4 | 6544236 | 2609757 | 6544236 | 69 | MISC SALES TRAN | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |
| 5 | 1404121 | 1667186 | 1404121 | 69 | KIOSK-GAS | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |
| 6 | 397896 | 1281329 | 397896 | 69 | KIOSK-GAS | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |
| 7 | 1426702 | 453293 | 1426702 | 69 | MISC SALES TRAN | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |
| 8 | 5703832 | 430940 | 5703832 | 69 | KIOSK-GAS | Private | COUPON/MISC ITEMS | GASOLINE-REG UNLEADED |

These results are available in SQL from the following file: *Preliminary_questions.sql*

## 1.3. Anomaly to be investigated

Exploring the data with SQL, I noticed the following anomalies:

- There are 13659 transactions where the QUANTITY = 0
- There are 13596 transactions where the QUANTITY =0 AND SALES = 0
- There are 63 transactions (difference of the 2 above) where QUANTITY=0 AND SALES > 0

The queries are available from the file: *quantity0.sql*
These findings should be clarified with the store, in order to better understand these transactions and eventually decide to keep, remove or process these lines for the modeling work.
For this project, as we have only 63 lines with very small values of sales, I decided to keep them for data processing simplicity. However, would they be more numerous and with higher sales values, they should be processed somehow to not biased the models.

## 1.4. Data Generation for modeling

Simple SQL queries are used to compute the sum of sales per WEEK or per DAY, to build the ARIMA model (the feature extractions for the Random Forest model is described in the related section). The results are then exported into a CSV file, that are then loaded in R for further processing.

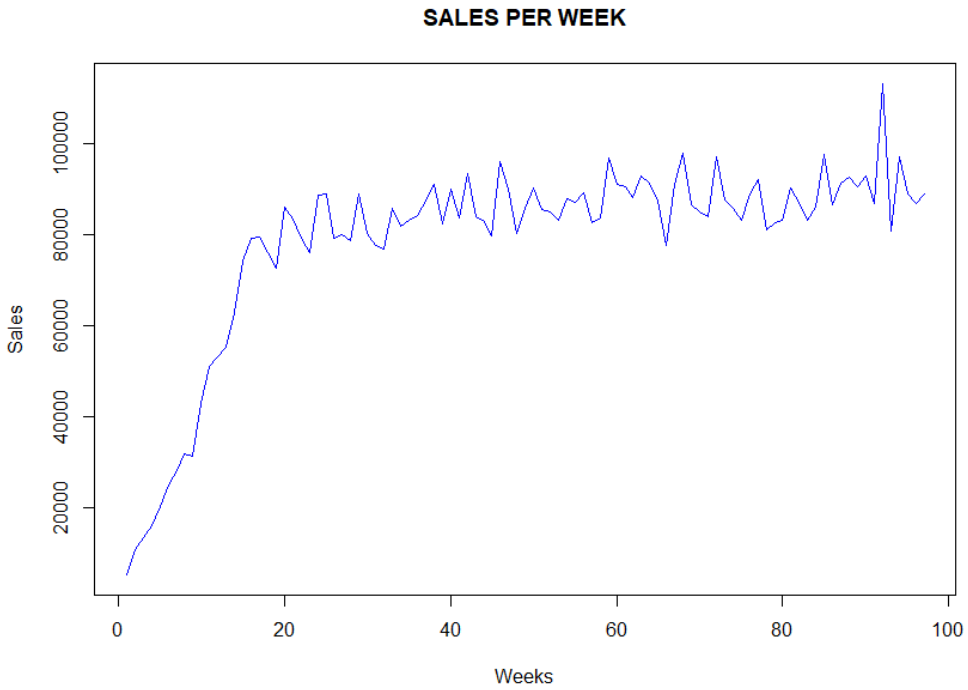
The queries are available from the file: *sum_of_sales.sql*
The CSV files are: *day_serie.csv and week_serie.csv*

Eventually, the daily data is unused in this study, but is mentioned as an alternative in the relevant section.

# 2. Predictive Modeling

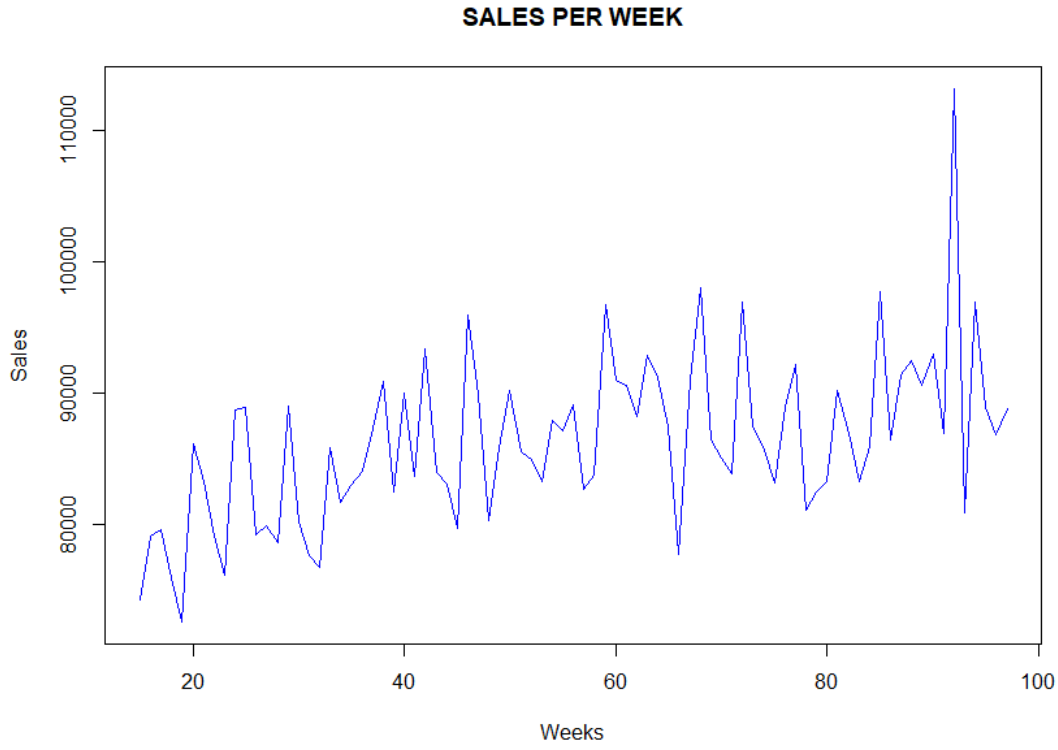
## 2.1. Arima

First-of-all, we may want to plot the weekly sales graph:




We clearly see a ramp in the sales at the beginning, that must correspond to the starting activity when the stores opened. In order to not be biased by this early activity, that does not correspond to a regular activity, the first 14 weekly sales points will be truncated.

After truncation, we end-up with the following sales evolution:



An increasing trend is clearly visible on the time-serie.
The Dickey-Fuller test provides a p-value = 0.23 (H0: the time-serie is non-stationary) that confirms the observation. A differentiation with a lag=1 allows to further work with a stationary time-serie.

R provides an auto-arima function that gives the following result:

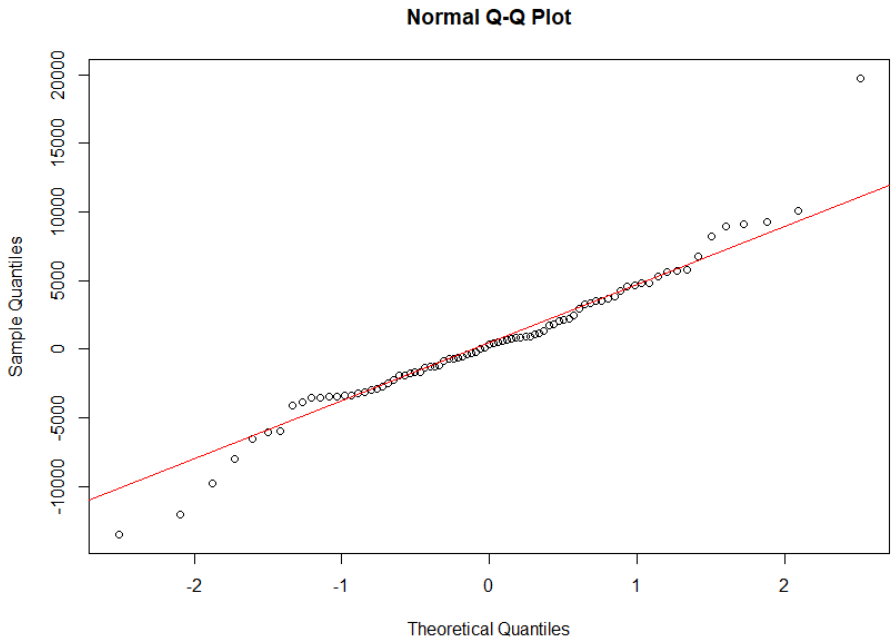
| Auto Arima Model | AIC criterion | BIC criterion | Week 98 Sales prediction |
|---|---|---|---|
| AR p=3, d=1, MA q=3 | 1641.21 | 1658.05 | 88877 |

However, we may want to do our own analysis, to possibly find a better fit:
Analyzing ACF (tailing off) and PACF (cutting off) schemes, I tend to think that we rather have an AR(p=5) scheme.

| Custom Model | AIC criterion | BIC criterion | Week 98 Sales prediction |
|---|---|---|---|
| AR p=5, d=1, MA q=0 | 1641.72 | 1656.16 | 83599 |

The (5,1,0) model should be a bit better as the BIC criterion is lower. **I will continue with this model.**

The normality of the residuals looks fine from the QQ plot but has some divergence at the extreme ends. We will confirm the normality with other means.

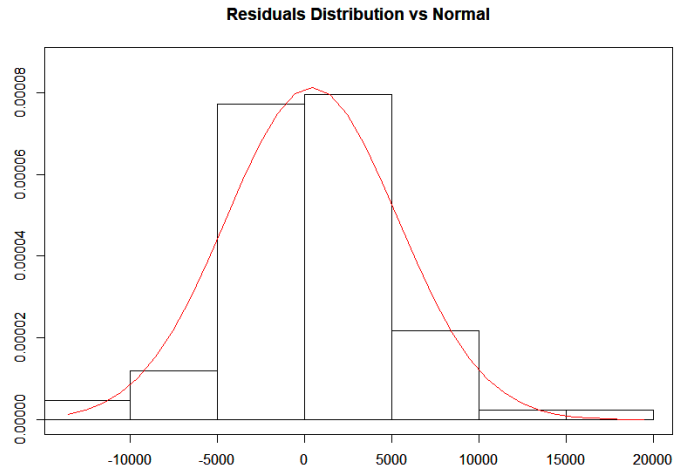



In particular, the Kolmogorov test (H0: the 2 distributions residuals and normal are identical) gives a p-value larger than the significance level (0.05), meaning that we can accept the assumption that the distribution of the residuals is identical to a normal distribution.

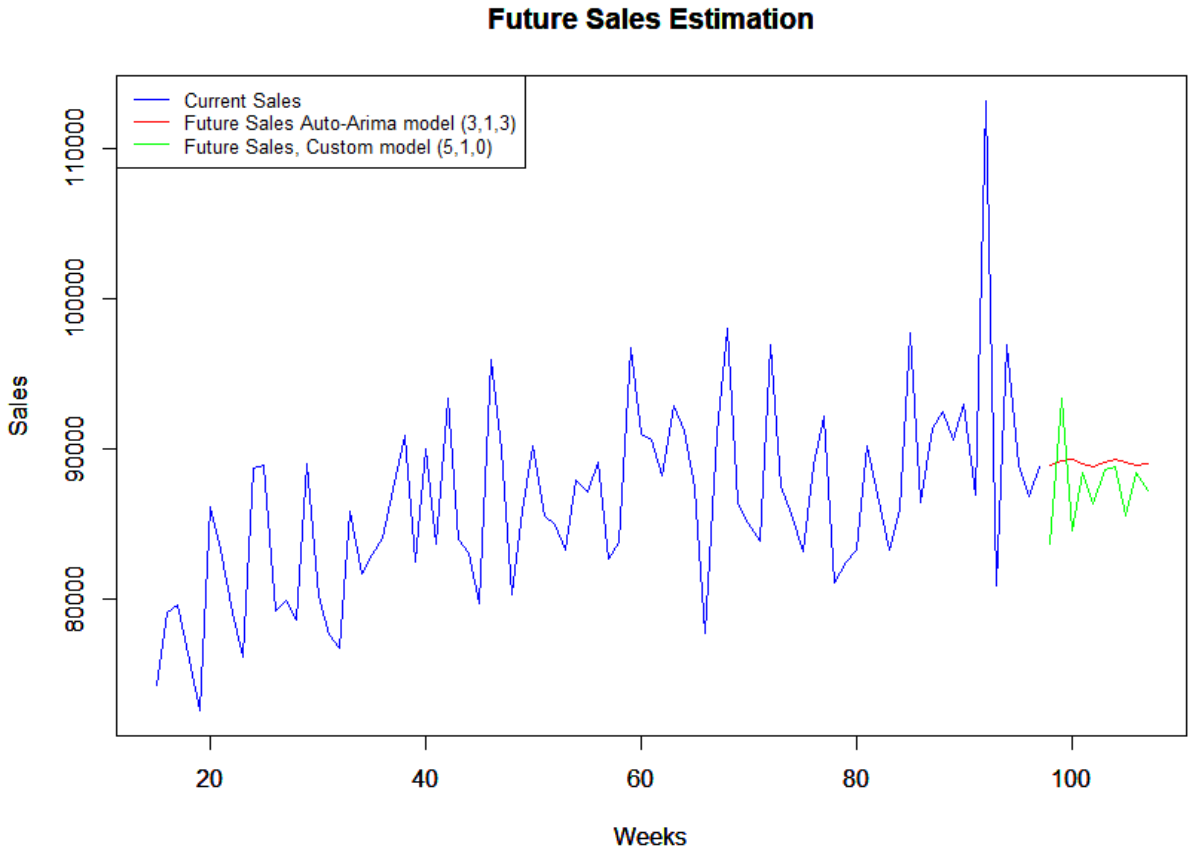
```
        One-sample Kolmogorov-Smirnov test

data:  unique(my_fit$residuals)
D = 0.1008, p-value = 0.3446
alternative hypothesis: two-sided
```

Moreover, plotting the residuals distribution together with the normal one shows that we are fine: we haven't overlooked any trend or seasonality.




Finally, we can plot and compare the predictions on 10 weeks ahead.
To me, the custom ARIMA (5,1,0) looks more realistic as the variance is larger than the other proposal, reproducing the variations of the current sales better.



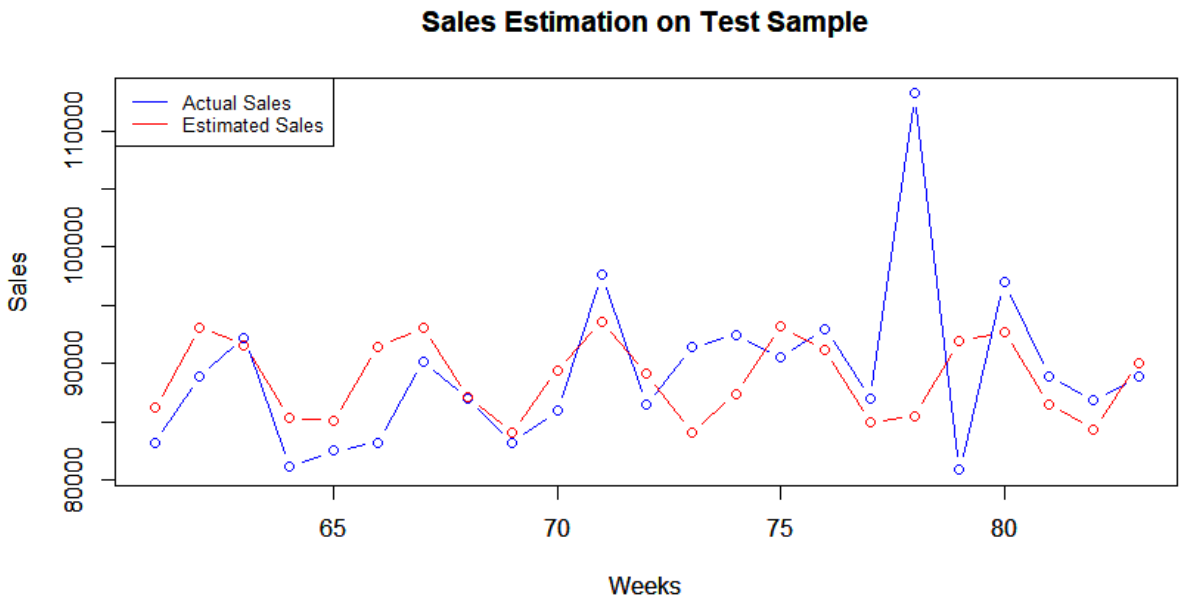
In order to compare this model with the next one (RF), we can:

1. Split the 83 data points into a 70% train (60 points) and 30% test (23 points),
2. Predict 23 points ahead
3. Compute the MAPE
4. Plot the predicted values against the true ones

| Arima Model | MAPE | MAPE Max | MAPE StdDev |
|---|---|---|---|
| AR p=5, d=1, MA q=0 | 4.5% | 22.14% | 0.0484 |
| AR p=3, d=1, MA q=3 | 4.9% | | |

As we can see from the table above, the custom model has a slightly lower error than the auto-arima one, leading to a slightly better fit.

The scheme below depicts the predicted values against the true ones on the test sample.



## 2.2. Random Forest

### 2.2.1. Features extraction

The transactions and the demographic tables look interesting to extract some features in order to predict sales. Indeed, the sales may be related to the population characteristics visiting the store such as income, household size, age. For example, large families with high income may spend more money in the shop than the others, leading to higher sales.
The product table – as opposed – does not look so interesting as it contains only the product definition.
Using SQL, the TRANSACTION_DATA and DEMOGRAPHIC tables are (left) joined and the following columns are consolidated:

- Week
- Household ID
- Age Class
- Income Class
- Household size Class
- Sales

Marital and homeowner status are not kept

1. To reduce the complexity
2. It might not be too relevant for a first approach

Household composition and kid category are co-linear to household size, so they are removed and only household composition (family size) is kept.

The generated SQL table is depicted below. It is exported to a CSV file.

| | WK | HH | AGE_DESC | INCOME_DESC | HOUSEHOLD_SIZE_DESC | SALES |
|---|---|---|---|---|---|---|
| 1 | 1 | 681 | NULL | NULL | NULL | 61.75 |
| 2 | 1 | 361 | 45-54 | 50-74K | 1 | 42.29 |
| 3 | 1 | 343 | NULL | NULL | NULL | 13.94 |
| 4 | 1 | 1490 | NULL | NULL | NULL | 32.69 |
| 5 | 1 | 2483 | 45-54 | 75-99K | 1 | 5 |
| 6 | 1 | 1590 | NULL | NULL | NULL | 44.7 |
| 7 | 1 | 2324 | 35-44 | 50-74K | 2 | 63.08 |
| 8 | 1 | 68 | NULL | NULL | NULL | 8.13 |

Notice: the number of distinct household IDs in the TRANSACTION_DATA table is much higher (2500) than in the DEMOGRAPHIC table (801). This will lead to missing data in the table. They will be imputed in R.

Additional SQL queries provide the number of lines that will require imputation:

| | |
|---|---|
| Total number of lines | 117480 |
| Number of lines with missing values | 63496 |
| Number of lines that are complete | 53534 |

These queries are available from the file: *features_extr_for_RF.sql*

The idea is to build a new matrix featuring the number of distinct “families” who did their shopping in the stores, the frequency of their different classes of Age, Income, Household size, PER WEEK.
The target variable is the sum of sales during the WEEK.

This leads to a table containing 1 line per week as below:

| | week | n_hh | A1 | A2 | A3 | A4 | A5 | A6 | I1 | I2 | I3 | I4 | I5 | I6 | I7 | I8 | I9 | I10 | I11 | I12 | P1 | P2 | P3 | P4 | P5 | Sales |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15 | 15 | 1156 | 25 | 80 | 120 | 843 | 42 | 46 | 23 | 26 | 39 | 15 | 6 | 515 | 49 | 6 | 103 | 283 | 50 | 41 | 147 | 873 | 67 | 32 | 37 | 74261.29 |
| 16 | 16 | 1244 | 33 | 93 | 128 | 962 | 44 | 49 | 26 | 29 | 124 | 17 | 7 | 339 | 60 | 8 | 120 | 480 | 57 | 42 | 171 | 987 | 74 | 35 | 42 | 79135.01 |
| 17 | 17 | 1262 | 38 | 100 | 130 | 712 | 45 | 346 | 26 | 26 | 191 | 23 | 7 | 10 | 95 | 426 | 327 | 132 | 63 | 45 | 176 | 1039 | 76 | 34 | 46 | 79556.35 |
| 18 | 18 | 1266 | 27 | 94 | 139 | 835 | 164 | 54 | 25 | 25 | 49 | 22 | 7 | 748 | 47 | 8 | 123 | 146 | 70 | 43 | 177 | 980 | 75 | 37 | 44 | 75968.18 |
| 19 | 19 | 1268 | 28 | 96 | 135 | 916 | 41 | 52 | 21 | 28 | 44 | 20 | 8 | 4 | 48 | 12 | 126 | 840 | 68 | 49 | 182 | 932 | 79 | 32 | 43 | 72536.88 |
| 20 | 20 | 1278 | 32 | 97 | 124 | 961 | 50 | 55 | 28 | 29 | 51 | 20 | 9 | 13 | 58 | 13 | 116 | 871 | 60 | 51 | 177 | 992 | 74 | 34 | 42 | 86108.90 |
| 21 | 21 | 1278 | 29 | 90 | 143 | 652 | 350 | 48 | 24 | 31 | 50 | 21 | 7 | 754 | 44 | 12 | 116 | 138 | 69 | 46 | 175 | 977 | 79 | 37 | 44 | 83262.33 |

| Week Num | Number of distinct household | Ax (x=1..6) Age class frequency | Iy (y=1 ..12) Income class frequency | Pz (z=1 .5) Household size frequency | Target variable Sales |
|---|---|---|---|---|---|

To be consistent and to allow a fair comparison between the 2 models ARIMA and RF, the 14 first points are removed as depicted and explained in the ARIMA section.

### 2.2.2. Imputation

As the Random Forest algorithm handles missing values, I initially started without imputation at all. This method leads to a MAPE score of the test sample of 6%.
However, imputation using the mode (most frequent value) of the classes gives a significantly better result (4.4%):

1. for each main feature (age, income, household size), the frequency of the different classes is computed over the complete dataset
2. the missing values are replaced by the mode of each class

Other techniques may be used, such as *rfImpute* from R using Random Forest, to find some feature similarities among the complete lines. In that case, it may be necessary to add more features (re-consider “Marital” and “Homeowner” state for example) and/or build new ones.
I explored imputation using *rfImpute* with the limited number of features I had, and obtained a slightly higher MAPE value, although it remains close to the mode-based imputation. To me, this relatively bad results is due to the lack of features used to impute AGE, INCOME and HH_SIZE.

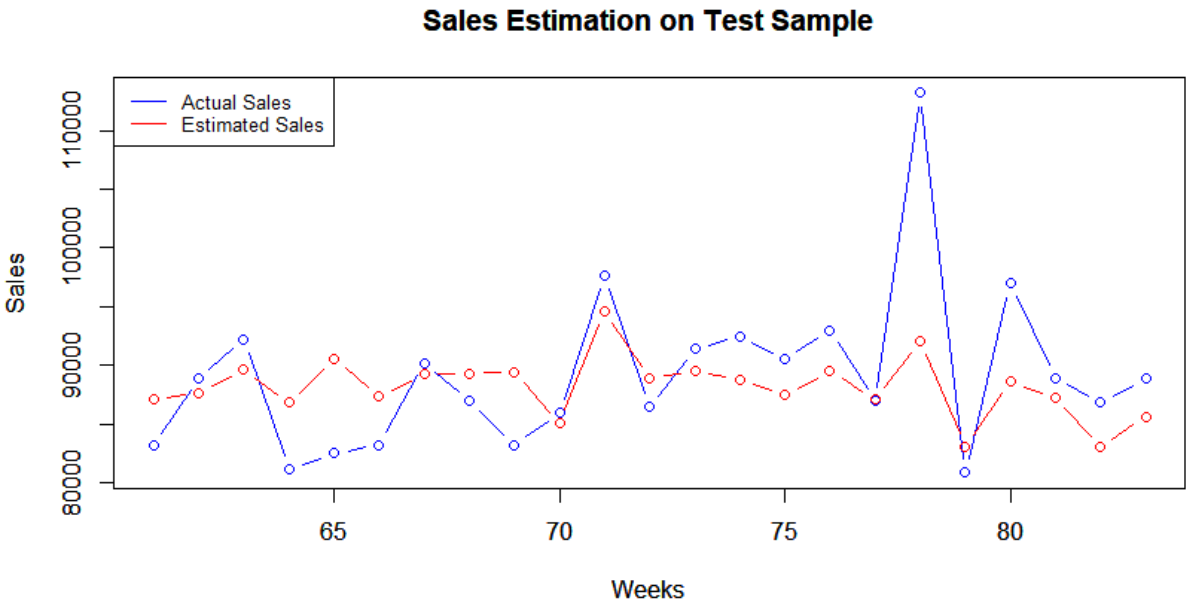
### 2.2.3. Results

The features of week98 are generated using the mean value of the past values for each feature, but a regression-based method could have been used as well.

| Model | Week 98 Sales prediction |
|---|---|
| Random Forest w/ mode imputation | 86610 |
| Random Forest w/ rfImpute | 87469 |

The results of RadomForest modeling are summed up in the table below:

| Model | MAPE | MAPE Max | MAPE StdDev |
|---|---|---|---|
| Random Forest w/ mode imputation | 4.4% | 18.72% | 0.0394 |
| Random Forest w/ rfImpute | 4.9% | 19.14% | 0.0411 |
| Random forest w/ no imputation | 6.0% | - | - |



## 2.3. Models evaluation and comparison

At this point, we can see that the RF Model is slightly better than the ARIMA one.

| Model | MAPE | MAPE Max | MAPE StdDev |
|---|---|---|---|
| Random Forest | 4.4% | 18.72% | 0.0394 |
| ARIMA (5,1,0) | 4.5% | 22.14% | 0.0484 |

However, this is achieved knowing that:

- A limited number of features were exploited
- The imputation is naive and could be improved with more elaborated techniques
- The hyper parameters of the random forest algorithm have not been explored (ntree, mtry)

So, not only the model is better at this stage, but it also has a greater potential to be refined and the accuracy to be improved.

Regarding the WEEK98 prediction, although the values provided by the different models are quite close, it is impossible to state on the accuracy of the prediction as the true value is unknown.

A few remarks may be worth mentioning about the 2 models:

- RF requires some feature extraction that may be complex and tedious
- RF may be more accurate (thanks to the averaging on ntree) but it is also more complex, hard to visualize, compute expensive
- With a reduced number of features, a regular CART tree may give good results and is easier to explain/visualize.
- ARIMA is quick and efficient and can provide predictions on n points ahead.
- The more there are points ahead to be requested on ARIMA, the higher the confidence interval.

The high sales at week 78 remain difficult to model as we could expect it, as nothing in the data permit to anticipate such a peak. It probably comes from an exceptional sales event. If it is a seasonal event, it may be modelled more accurately with a dataset covering a larger time period including more occurrences of this event.

# 3. Alternatives

Beside the improvements mentioned in the section above on RandomForest specifically, there are other methods and techniques that could be explored to further improve the modeling of this timeseries:

- Use the daily data instead of the weekly data
- A previous project on a timeseries led me to experiment techniques such as MLPRegressor from SciKitLearn, Radial Based Function Networks (RBFN) or Generalized Regression Neural Networks (GRNN). Please refer to https://github.com/remihardy/DSTI-ArtificialNN-Project
- Long Short Term Memory Network (LSTM), built in Keras for example, are becoming popular to model a time-serie.
- A Bayesian Neural Net as described in the link below may also be worth experimenting https://eng.uber.com/neural-networks-uncertainty-estimation/?fbclid=IwAR0y0jRZLU_EfWR1o8vThUpm7qmoGpuFonqD78Y-ETS9qs8QwNWeljpglg0

Also, we can discuss the article proposed in the project presentation:

The article presents a method to achieve probabilistic forecasting for “thousands or millions” timeseries. Strictly speaking, the article does not fully apply to the goal of our study:

- Probabilistic forecasting is interested in predicting the probability distribution of the timeseries, rather than single points in time.
- We have only one individual (sales) time-serie to model

In the article, the model learns from a great number of related time series and some co-variates, and forecasts a similarly great number of time series together.

While the method may seem over-kill at first, adapting our problem may allow us to apply it, assuming it can scale down from “millions” to “hundreds” (the article mentions in its conclusion that it may be “*applicable to medium-size datasets containing only a few hundred time series*”).

Regarding the first, point, a probability distribution may be as interesting as points in time. For the second point, an idea could be to use the sales of the individual stores or even the sales from the individual households.

Eventually, our predicted individual time series would have to be consolidated - as opposed to the problem description in the article – hoping for a better accuracy.

## 4. LSTM based model

I lately and rapidly explored an RNN based solution as suggested above.
The file is available as *lstm.py* and uses Keras.

I ended-up with a MAPE = 4.6% which is close but not better than the solutions that were worked out previously in the document. My assumption is that I do not have enough points (not enough data) using weekly sales points. Having more weekly sales points or using daily sales data instead may bring better results.

Screen snapshot of Keras Training on the train sample, and the error on the test sample:

```
IPython console
Console 1/A
Epoch 91/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0214
Epoch 92/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0218
Epoch 93/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0187
Epoch 94/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0194
Epoch 95/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0198
Epoch 96/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0188
Epoch 97/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0183
Epoch 98/100
40/40 [==============================] - 1s 20ms/step - loss: 0.0199
Epoch 99/100
40/40 [==============================] - 1s 19ms/step - loss: 0.0205
Epoch 100/100
40/40 [==============================] - 1s 19ms/step - loss: 0.0180
MAPE = 0.04649171343558134
```

Predicted sales on the test sample: