Data-Mining Project: BIP

Alberto Parravicini, Simone Ripamonti, Luca Stornaiuolo

22 giugno 2016

Interactive Shiny report at: https://albertoparravicini.shinyapps.io/DM-Report/

Repository at: https://github.com/ripa1993/DM-Project

Forecasts over 10 days: https://github.com/ripa1993/DM-Project/blob/master/Results/PREDIZIONE__FINALE__FUTURE.csv

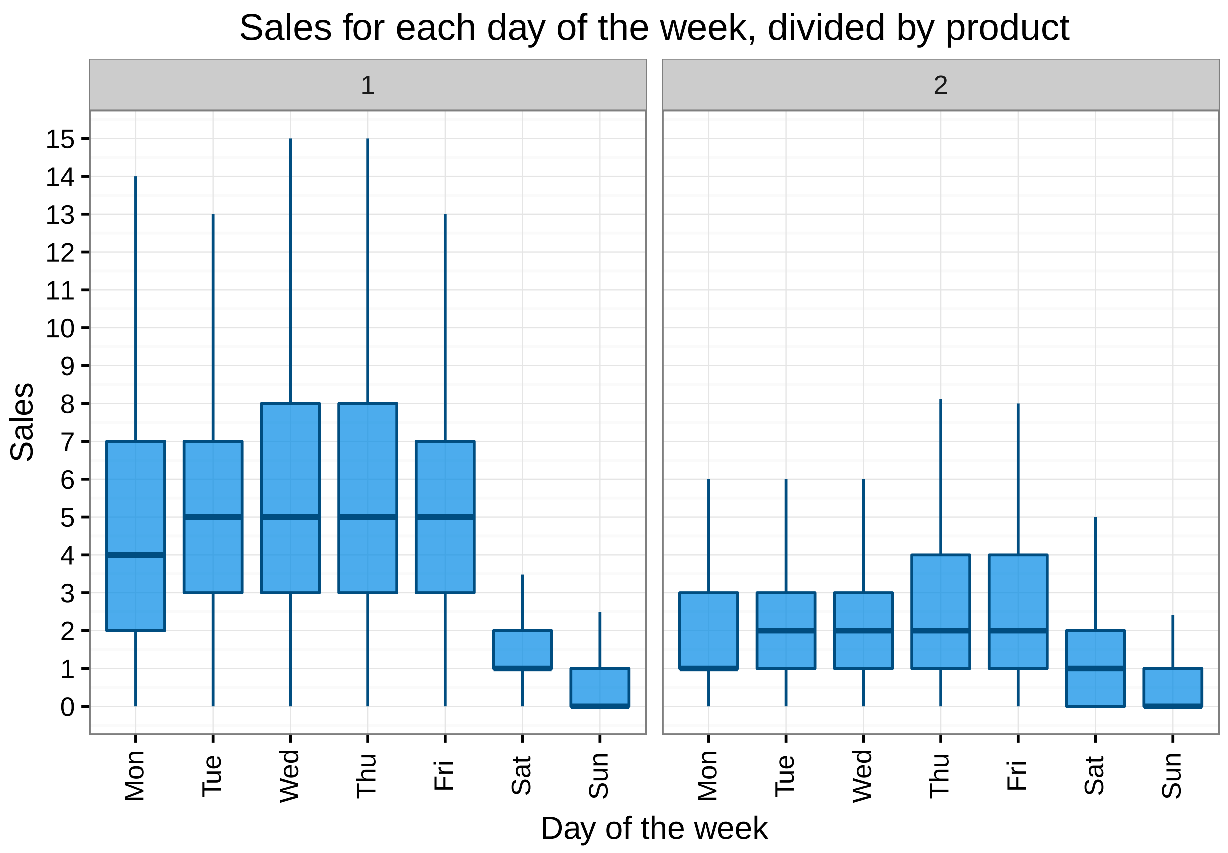
Presentation at: https://albertoparravicini.shinyapps.io/DM-Presentation/

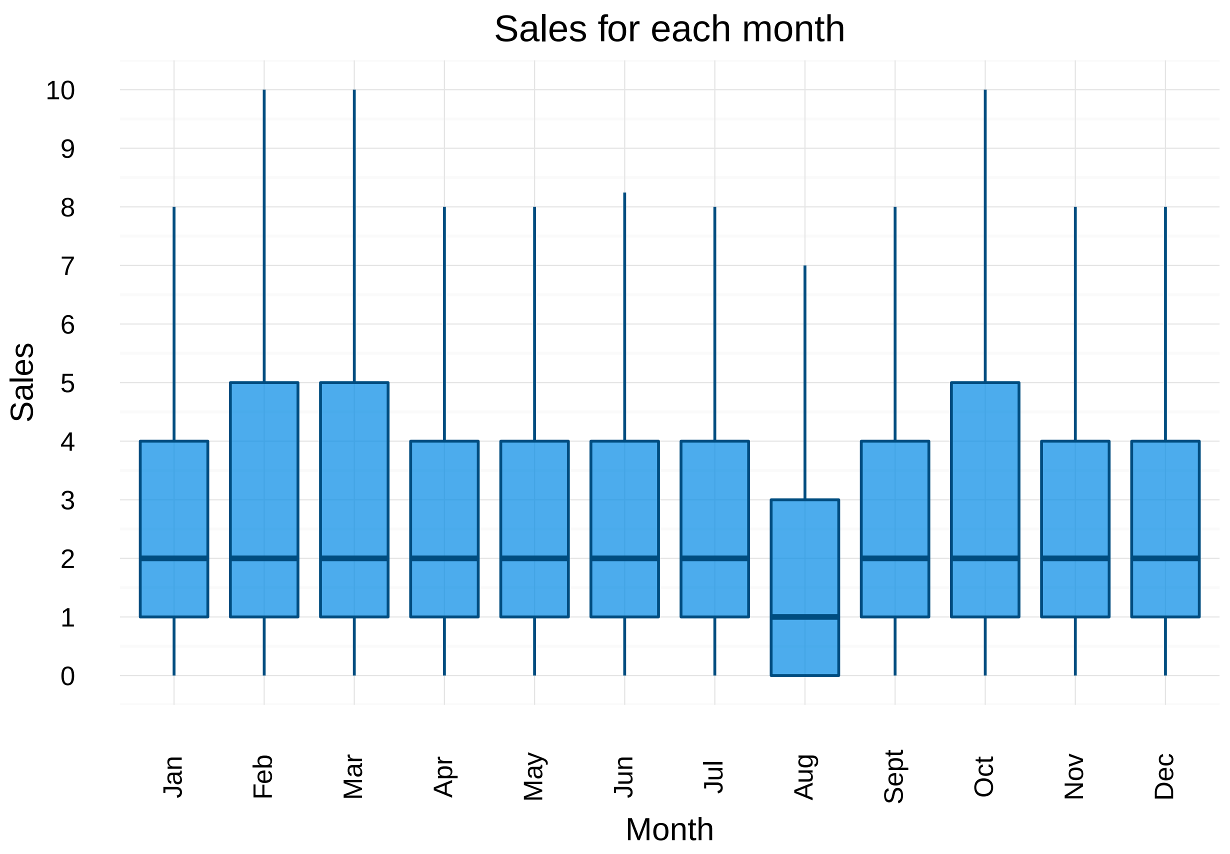
DATA EXPLORATION:

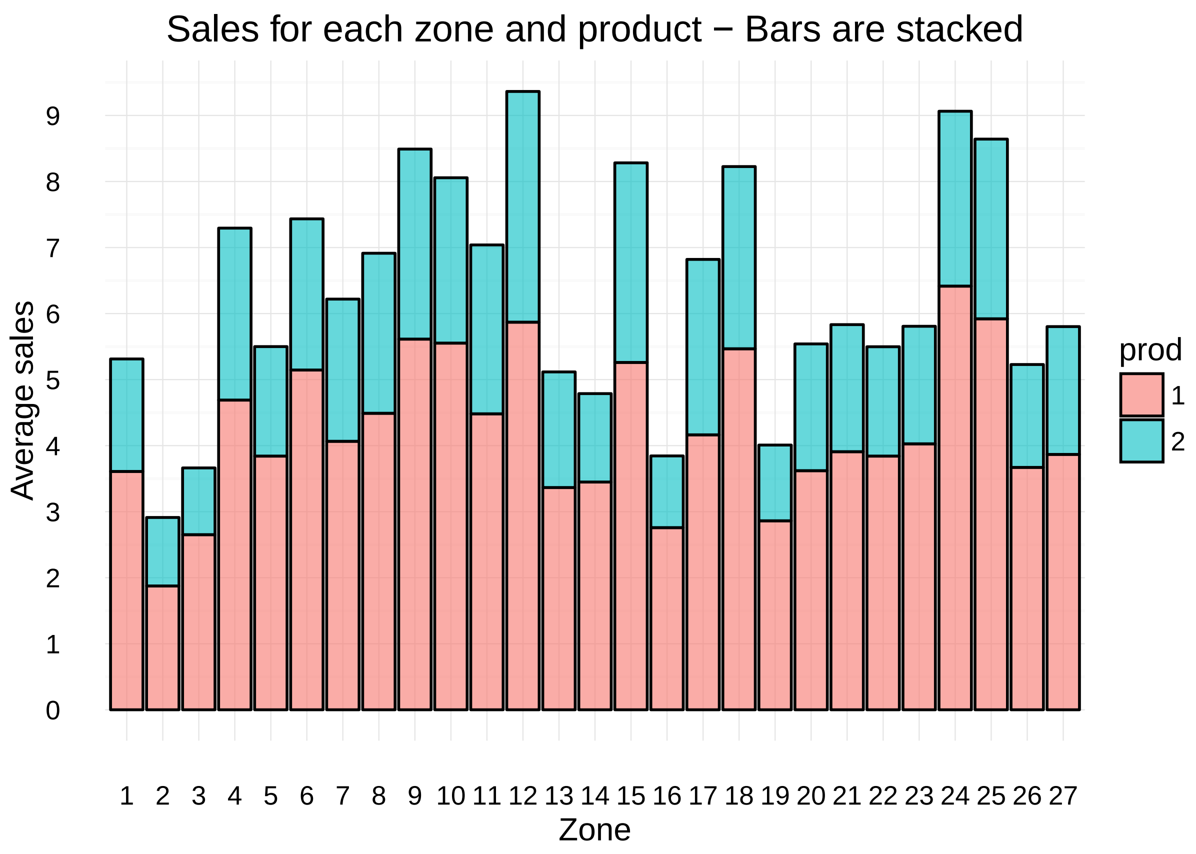
This section contains the preliminary operations we performed on the dataset: data visualization, outlier identification, and in general anything that could help us to better understand the dataset and the information it contains.

Initial exploratory analysis: we identified the *distribution of sales* (cumulative or divided by product) across areas, days of the week, months, years. Tuples in the dataset are uniquely denoted by the key "Product, Subarea, Date".

Among other things, we visualized the sales across the days of the week and across months, and also how sales varied depending on the various geographical areas.
Below, a brief selection of these plots.





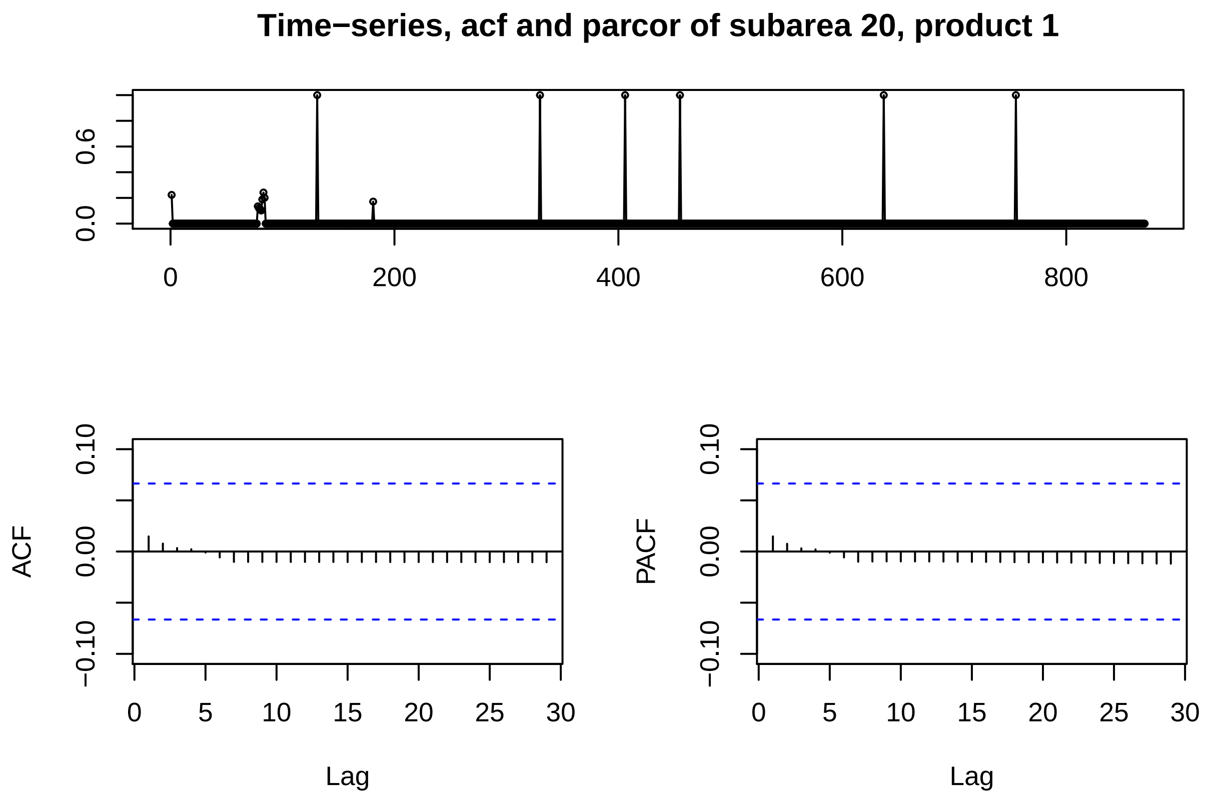


Outlier and missing values: the dataset doesn't explicitely contains missing values (i.e. NA or similar values). There is however one missing subarea (**51**) and a number of outliers; as areas are mapped to an arbitrary number of subareas (and each subarea is mapped to exactly one area), there is no reason to believe that the missing subarea should negatively affect other predictions. On the other hands, the extremely high sales values at *"2014-06-30"*, and the lack of sales at *"2014-01-01"*, and over the period from *"2014-03-19"* to *"2014-03-25"* can be considered outliers and missing values: indeed, these events seems to be purely random and not correlated to any subsequent behaviour of the sales. We decided to replace these values with a prediction (see **"Data preparation"** for further details regarding the metodology).

A number of subareas display unusual sales behaviours: subarea **20**, and subareas **32** and **78** (relatively to product 2) have mostly no sales. We then decided to inspect the time-series of these subareas and of the other subareas, to understand if they displayed a predictable internal structure or if they resembled white noise.
The *Box-Pierce* test statistic examines if a univariate time-series is composed of independent observations: this is the case of white noise or of constant signals, whose best predictor is the mean of the signal itself.
The null hypothesis of the *Box-Pierce* test is that the signal is made of independent observations: a low p-value implies that this isn't the case, and the signal has a more complex internal structure.

```
# [1] "Box-Pierce test for subarea 20, product 1"
#
#   Box-Pierce test
#
# data:  subarea_20_p1$Vendite
# X-squared = 0.057371, df = 1, p-value = 0.8107
```

```
# [1] "Box-Pierce test for subarea 22, product 1, for comparison (this timeseries has a complex structure)"
#
#   Box-Pierce test
#
# data:  filter(dataset_polimi, Sottoarea == "Sottoarea_22", Categoria_prodotto ==     "Prodotto_1")$Vendite
# X-squared = 88.688, df = 1, p-value < 2.2e-16
```



Subarea 20, and subareas 32 and 78 (relatively to product 2) can indeed be considered deterministic signals with mean 0: using the tuples referring to these time-series in the training would not be beneficial, so we removed them from the dataset, and predicted sales equal to 0 *a-posteriori*. By using the geographical coordinates at our disposal we noted that these areas are close to Milan, Lamezia Terme and Potenza: unrelated cities that further confirm the hypothesis that keeping these values in the training set would not be useful.

DATA PREPARATION:

In this section we discuss the process of further cleaning the data, and the feature engineering we applied to the dataset.

Basic feature engineering

- Split the dates into:
  - Day of the week.
  - Day of the month.
  - Day of the year.
  - Month.
  - Week of the year.
  - Year.
- Add a "Weekend" column: is a certain day a saturday or a sunday?
- Add "Holiday": is a certain day an Italian holiday (Christmas, Easter, etc...)?
- Add "First of the month": we noticed that during the first day of the month sales are generally lower. Out of curiosity, we added a boolean variable to explicitely keep track of whether a day is the first of the month or not.

Other feature engineering

- We added various features that keep track of cumulatives sales over certain periods:
  - Cumulative sales, divided by product and subarea, over each week, month and year (i.e. how many product the company sold over a certain period, in a certain area).
  - Aggregate sales of the entire company, day by day, divided by product and cumulative (i.e. how many product the company sold in a certain day).

It is not possible to compute directly these values for dates above the one in the dataset: to overcome this issue, we built various *Sarima* models that predicted each of these time-series, over the desired prediction period.
The idea is that using a *Sarima* model built specifically for a certain time-series (e.g. the aggregate sales) can lead to a highly accurate prediction that can be beneficial to the other models. The cumulative sales didn't prove to be very useful, but the predicion of the aggregate sales improved sightly our results.

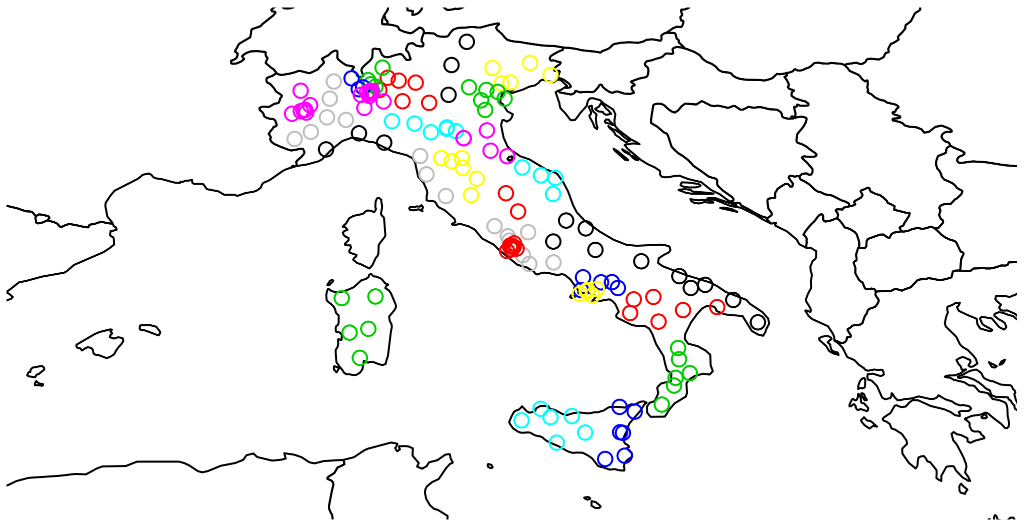
Note that to perform predictions over test and validation sets we substituted their real values of cumulative sales with our predictions: if we used the real values, the results would be faked, as we'd be employing information unavailable in a real prediction.

- We replaced the outliers and the missing values identified before with a prediction found through a *Random Forest* model: As these values are located in the middle of the dataset, instead that at the end, a *Sarima* model wouldn't be as effective, as it could employ fewer consecutive data for training.
  The values replaced are the ones at *"2014-01-01"*, *"2014-06-30"*, and from *"2014-03-19"* to *"2014-03-25"* (in the 2 following years, sales at *"01-01"* are usually rather low, but never completely 0 as in 2014)

Coordinates clustering

As we were provided the GPS coordinates of each subarea, we plotted their latitude and longitute on a map, grouping them by zone and giving each zone a different color. We noted how the division by zones/areas/subareas was partially consistent with what we expected (division by regions, etc...), but it wasn't as *smooth* as one might desire. Morevoer, the division by zones was already quite fine grained (about 20), so we decided to build a higher level hierarchy, to keep into account other geographical subdivision of Italy.

Original division by zone

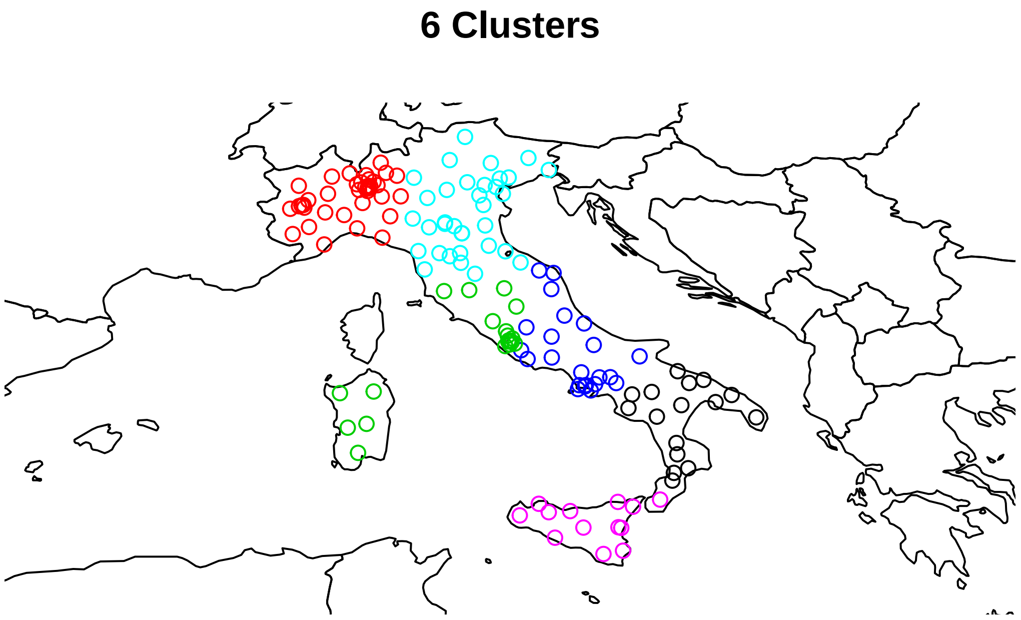


We decided to perform a **Knee-Elbow** analysis to understand how the subareas could be grouped together in a different way than the one provided by zones and areas. First of all we built a **hierarchical clustering**, and from its results, we plotted WSS and BSS errors.

Then we decided to group subareas according to three different **k-means** runs:

- 3 cluster grouping, to distinguish between North, Center and South Italy.
- 6 cluster grouping, according to the value suggested by the **Knee-Elbow** analysis.
- 20 cluster grouping, to distiguish the different Italian region.

We moved from hierarchical clustering to **k-means** as with the latter we would easily be able to assign new points to existing clusters, through the **clue** package.



MODEL BUILDING:

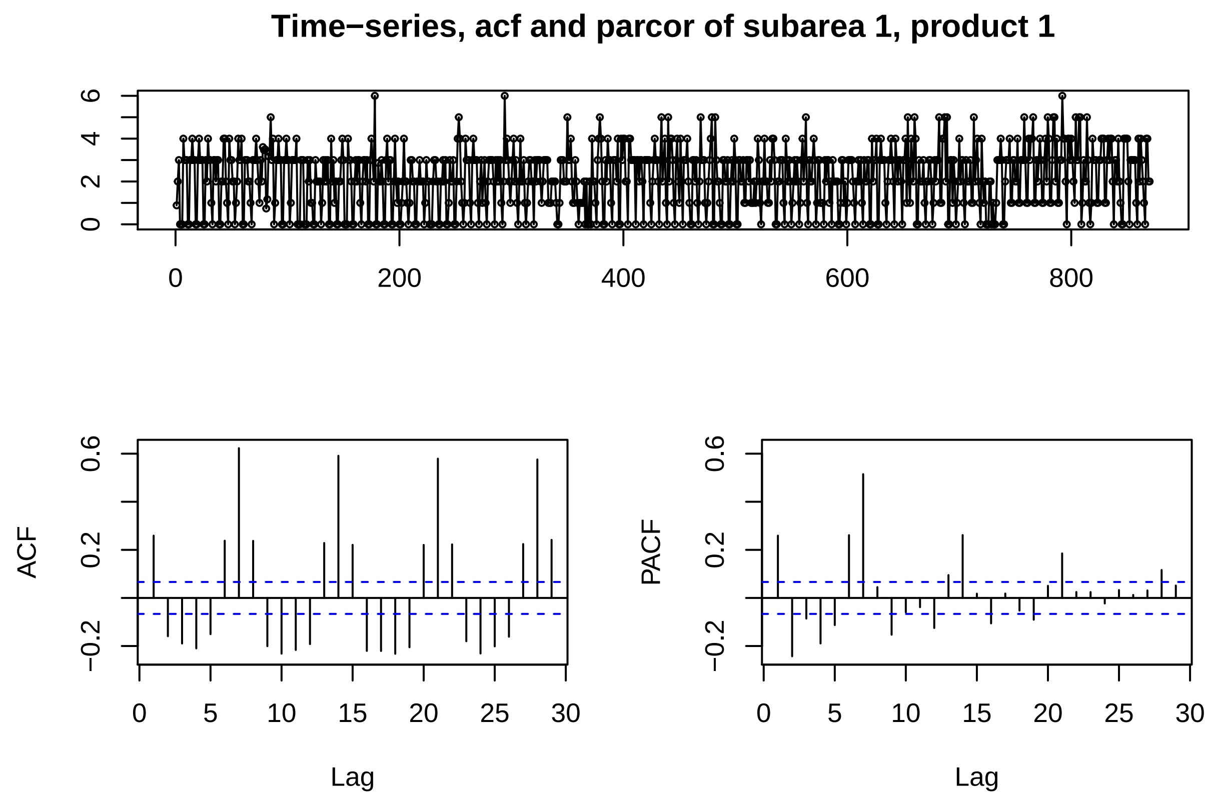
Seasonal ARIMA:

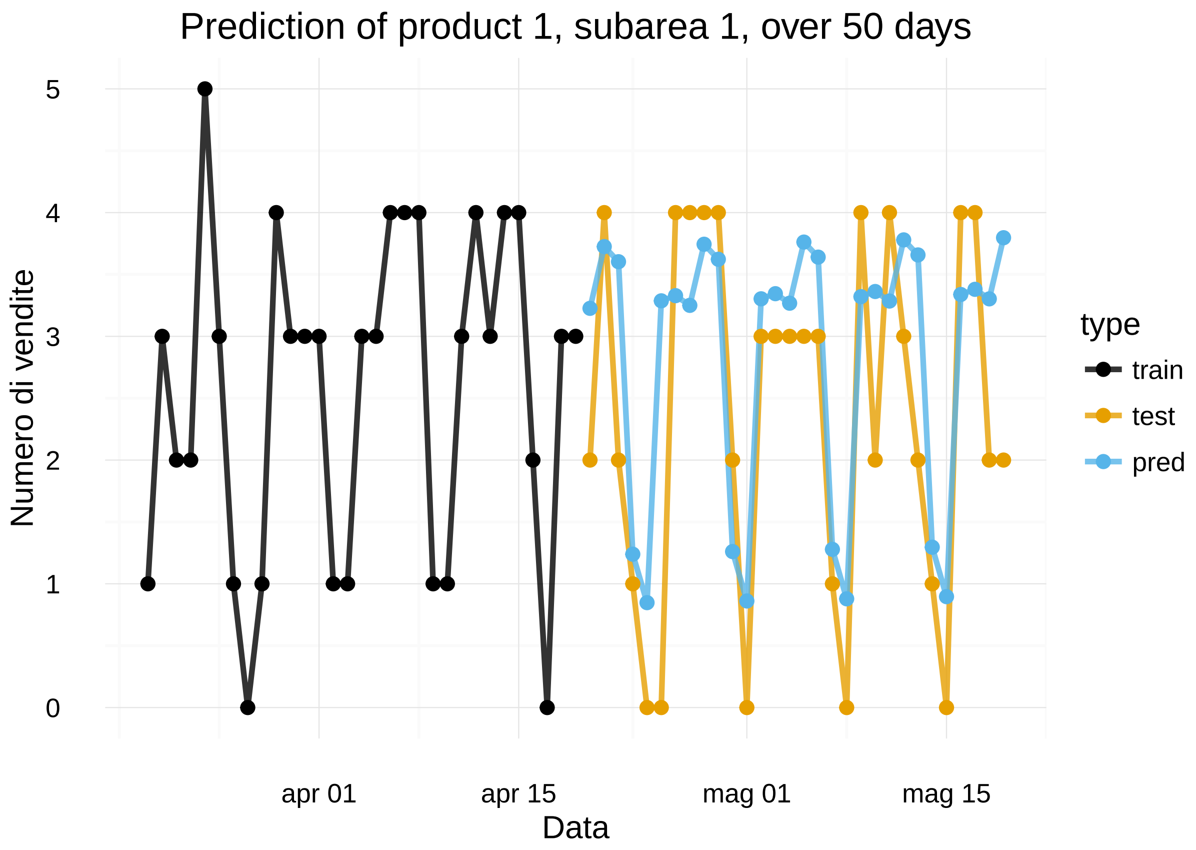
Our first approach to the prediction of sales was to use **Seasonal ARIMA** models (Seasonal Auto-Regressive Integrated Moving Average, or **Sarima**): these models are in general capable of fitting the structure of very complex time series, even those with seasonalities or trends like in our case. Furthermore, finding the right order of a *Seasonal ARIMA* is something that can be done (with a certain degree of approximation) by applying mathematical and statistical analysis to the time-series themselves.

We built a different *Seasonal ARIMA* for each subarea and product, under the assumption that all the time-series could be fitted with a model of the same order (but different parameter values, of course). This hypothesis seems to be true for the majority of cases, with a few exceptions where the other models we built (with Random Forest and XGBoost) outperforms Sarima by a wide margin.

It is interesting to note that *Seasonal ARIMA* uses only the values of the time-series itself for its forecast: it doesn't use any other feature, and every subarea/product combination is completely independent from the others. Despite these strong limitations, *Sarima* managed to perform almost as well as the other models, and in some cases even better (see the specific section for results).

As an example, **Auto-correlation**, **Partial auto-correlation**, to understand the **MA** and **AR** orders, or the **Neyman Smooth Tests of Normality** to check if the residuals of the training can be considered **white noise**.





From a simple inspection of the *autocorrelation function*, we can see how the time-series has a clear weekly seasonality. After various trials, by making use of metrics such as the *mean squared error*, the *AIC*[1], and the estimated variance of the

[1]http://www4.ncsu.edu/~shu3/Presentation/AIC.pdf

parameters, we opted for a $(2,1,1)(4,1,4)_7$ model.
The residuals of this *Seasonal ARIMA* can be considered white noise (low p-value: very likely to be white noise).

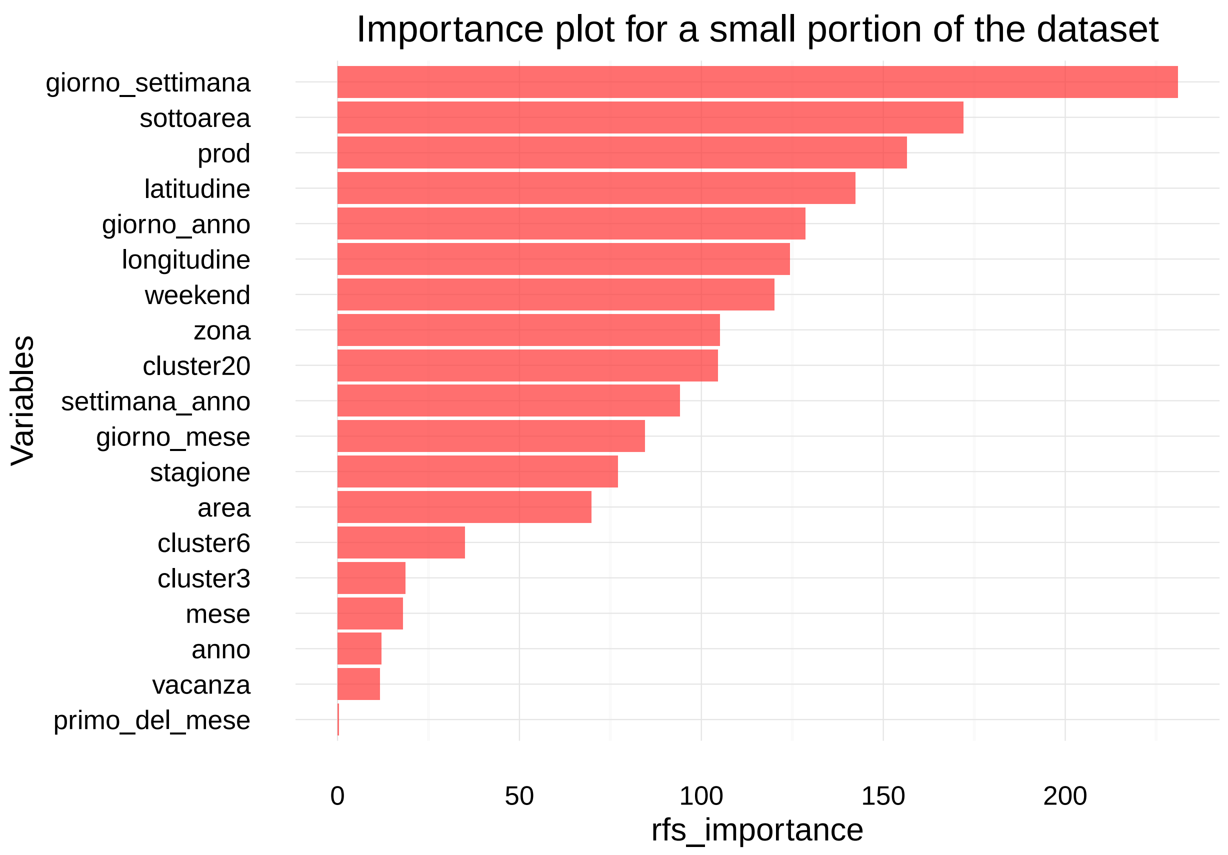
```
nortestARMA(res_tot, fit$sigma2)
#$pvalsdmuest
#[1] 4.927925e-06
```

We also used the overall sales of the company, aggregated on a daily basis, as exogen input of the model: this lead to a slight improvement of the quality of the model, with respect to all the metric considered.

R offers to methods to optimize the *Seasonal ARIMA* models: **CSS-ML** usually offers better results, but sometimes it fails to converge; **CSS**, doesn't have this problem, but it is in general less precise. We used the former to predict the aggregated sales, and the latter for the single subareas.

Random Forest:

The following approach to prediction was to make use of **Random Forests**: as explained before, *Sarima* cannot make use of all the information available in our dataset, so we naturally moved to a model capable of exploting them with ease.
Random Forest proved to be quite flexible and robust, and provided us results that often exceeded the capabilities of *Sarima*; another advantage of *Random Forest* is the ability to compute the **variable importance plot**, to get a better insight of what are the most relevant features used by the model.



We tried two approaches with **Random Forest**: we built a single model that uses the whole dataset, and a set of models, one for each subarea/product combination. At first the second approach seemed to yield better performances, but tuning the parameters of the model resulted in the single model to be superior. In details, the single model requires a higher number of trees that each of the *multiple model* forests, probably because of the higher number of features that must be kept into account when splitting (product number, location information such as coordinates and clusters).

The **out of bag** error was the main metric used to compare different random forests model, as it is in general considered a good estimate of the real predictive power of the model; we also employed *mean square error*, *MAPE* and *MAX APE*, computed on a test set. These last metrics were also useful to compare the forests to the *Sarima* models.

We used the **ranger** package for R, as it is much faster than the commonly used **randomForest**, and can make use of multi-threading.
The final random forest used 1200 trees. Using more trees didn't reduce by a significant amount the *out of bag* error and the *MSE* on the test set, and proved to be computationally prohibitive.

```
rfs_model <- ranger(train_set,
                   formula=vendite ~ (zona + area + sottoarea  + prod +
                    giorno_mese + giorno_anno + settimana_anno + mese + anno  +
                    stagione + primo_del_mese + cluster3 + cluster6 + cluster20 +
                    latitudine + longitudine + giorno_settimana+weekend+vacanza),
                   num.trees = 1200, write.forest = T, verbose = T, num.threads = 4, ...)
```

XGBoost:

eXtreme Gradient Boosting is a classification and regression algorithm based on gradient boosted trees that is widely used due to some winning features:[2]

- Efficiency: automatic parallel computation on a single machine and possibility to be run on a cluster
- Accuracy: good results for most of the data sets, even with heterogeneuos data types
- Feasibility: customized objective function, evaluation metrics and lots of tunable parameters

As with **Random Forest** we tried two different approaches at modeling the dataset using XGBoost: creating one single model for the whole dataset and creating one model for each combination of subareas and products. We splitted the dataset into two different parts: one test set containing last 10 or 50 days of the available sales, and a training set containing the remaining instances.

We run XGBoost's crossvalidation function to get accurate insights of the quality of the two predictive models. We tested and tuned both models by selecting different combinations of attributes and by trying different values for the parameters **nrounds** (number of iterations), **eta** (learning rate), **gamma** (minimum loss reduction to partition on a leaf node of the tree) and others.

We used builtin XGBoost metrics **root mean square error** (RMSE), **logarithmic loss** (logloss) and **mean average precision** (MAP) to evaluate the quality of the model while it was building.

```
xgb.cv(data=xg_train, nrounds = n_rounds, nthread = 4, watchlist=list(train=xg_train), eta = 0.1,
                      nfold=10, eval.metric=c("logloss","rmse","map"), tree_method="exact")
```

We used XGBoost's watchlist and evaluation metrics to keep an eye on the training and test error and perform *early stopping* to prevent overfitting of the test data. This helped us to improve the accuracy of the prediction on the test set.

```
xgb_model <- xgb.train(data=xg_train, nrounds = n_rounds, nthread = 4, eta = eta_value,
              watchlist = list(train=xg_train, test=xg_test), early.stop.round = 5, maximize = F)
xgb_pred <- predict(xgb_model, xg_test)
```

Once the prediction was ready we applied also mean absolute percentage error (MAPE[3]) and max absolute percentage error. These metrics helped us to understand that the single model was performing slightly better than the multiple one, so we decided to focus our efforts on the former.

Features and parameters for the final model:

- features: *zona, area, sottoarea, prod, giorno_settimana, giorno_mese, giorno_anno, settimana_anno, mese, anno, weekend, primo_del_mese, vacanza, stagione, cluster3, cluster6, cluster20, latitudine, longitudine*
- label: vendite
- nrounds: 900
- eta: 0.1
- gamma: default

---

[2]http://www.slideshare.net/ShangxuanZhang/xgboost
[3]https://en.wikipedia.org/wiki/Mean_absolute_percentage_error#Alternative_MAPE_definitions

RESULTS

Among the three models (*Sarima*, *Random Forest*, *XGBoost*), **XGBoost** proved to be the most powerful one; however, by looking at the **mean square error** computed over the various product/subareas combinations of the test set, we noticed how in some occasion *Sarima* and *Random forest* would provide better predictions.

We decided to keep this result into account, and build an ensemble model that would use the predictions of the three models, and weight them accordingly to the estimated *mean square error* of each product/subareas combination in the test set.

For each prediction, we considered the *mean square error* estimated over a test set by the three models, for the same subarea and product of the prediction that is being made.

$$prediction = prediction_{xgboost} \cdot w_{xgboost} + prediction_{forest} \cdot w_{forest} + prediction_{sarima} \cdot w_{sarima}$$

With:

$$w_{xgboost} = \frac{mse^{-1}_{xgboost}}{mse^{-1}_{xgboost} + mse^{-1}_{forest} + mse^{-1}_{sarima}}$$

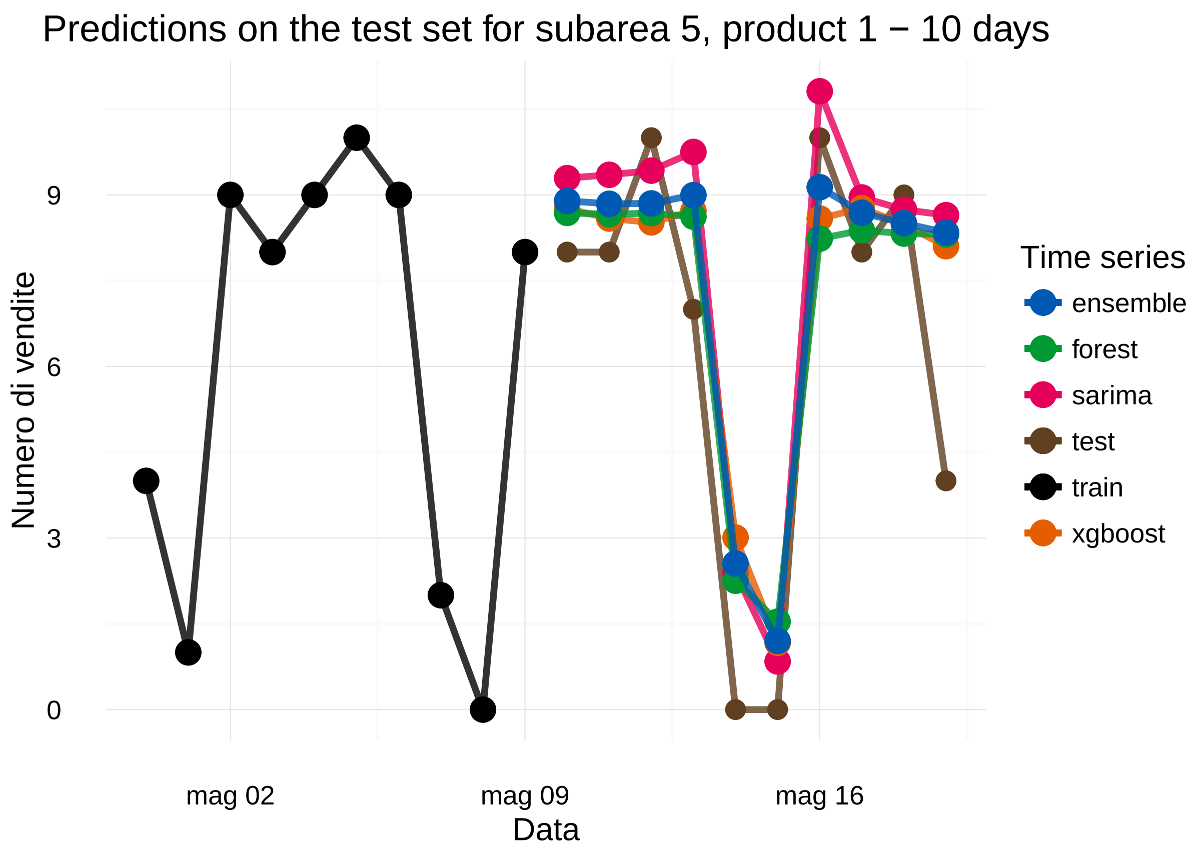
$$w_{forest} = \frac{mse^{-1}_{forest}}{mse^{-1}_{xgboost} + mse^{-1}_{forest} + mse^{-1}_{sarima}}$$

$$w_{sarima} = \frac{mse^{-1}_{sarima}}{mse^{-1}_{xgboost} + mse^{-1}_{forest} + mse^{-1}_{sarima}}$$

Note that the weighting was done without using the predictions for the outlier subareas (the ones for which we predicted 0 sales over the period), to avoid skewing the results with handmade predictions.

The results below list the performances of the various models, with **MSE**, **MAPE**, **MAXAPE** being computed on a 10-days length test set, and *Out of bag*, *Logloss* being computed by the Random forest training and the Xgboost cross-validation, respectively. In all cases, we also considered the subareas for which we manually predicted 0 sales.

MODEL	MSE	MAPE	MAXAPE	OTHER
Sarima	2.146694	0.2788613	3.521273	-
Random Forest	2.148978	0.2865883	4.247568	*Out of bag:* 1.771598
XGBoost	1.964951	0.2705864	4.012398	*Logloss:* -29.58445
Ensemble	1.876493	0.2614705	3.876331	-


Predictions on the test set for subarea 5, product 1 − 10 days
Numero di vendite
0
3
6
9
mag 02
mag 09
mag 16
Data
Time series
ensemble
forest
sarima
test
train
xgboost