Time Series Forecasting of sales of an Ecuadorian store chain

Nicklas Prochazka s87375@bht-berlin.de
Data Science
Fachbereich VI
Berliner Hochschule für Technik

Abstract

This paper describes the development of Machine Learning algorithms to be used for predicting future prospects in the *Multiplayer Online Battle Arena League of Legends*, using in-game statistics. The data used is from the *LCK* (Korean league) from 2015-2022, resulting in 904 observations with 20 continuous features. These will be used to predict the MVP points of a player and their respective rank within the league. The features will first be checked for collinearity and importance using the Pearson Correlation Coefficient, Stepwise Regression, the Variance Inflation Factor and LASSO. Based on these results, half the features will be eliminated and the remaining ones used to develop two models for predicting MVP points, one using Random Forest Regression and the other using Support Vector Regression. The RF performs best, with a normalized MAE of 0.1258 and an RMSE 0.1667 (using the LCK test set). Using the models on test data from different leagues, they perform a bit worse. Predicting ranks of players is not feasible as all models fail to consisently predict the top five, let alone the top player.

Keywords: Time Series, Forecasting, EDA, Parallel Hyperparameter Optimization, Store Sales, Exponential Smoothing, SARIMAX, Kaggle, Favorita

1. Introduction

Time series analysis and forecasting is a powerful analytical tool that can provide valuable insights into the future performance of businesses. For store owners, the ability to accurately predict sales trends and anticipate market changes is essential for maintaining profitability and competitiveness. However, the process of time series forecasting can be complex and challenging, requiring expertise in data analysis, statistical modeling, and machine learning.

In this paper, a detailed case study of TS analysis and forecasting is applied to store sales data from the Favorita grocery chain in Ecuador. The analysis encompasses a range of techniques, including exploratory data analysis (EDA), parallel hyperparameter optimization using cross validation (CV) (using a self-written and published Python package), and model selection using popular approaches such as Exponential Smoothing (ExpSm) and SARIMA(X) (trying both univariate and multivariate analysis). These are then used to forecast approximately two weeks.

By following this methodology and applying the principles and practices of TS analysis and forecasting, store owners can gain deeper insights into their sales data and make informed decisions about inventory, pricing, and marketing strategies. This paper serves as a comprehensive guide to the tools and techniques of TS analysis and forecasting and offers practical advice on how to apply them to real-world business problems.

2. Time Series Analysis and Forecasting: Theory and Methodology

Time series (TS) analysis is a prelude to TS forecasting. It uses statistical techniques to evaluate how time-dependent data changes over time. As such, it involves the analysis of data points that are taken at (normally) regular intervals over time, and the aim is to understand the underlying patterns, trends, and relationships in the data. With this understanding, an appropriate model is generated to predict/forecast future data points. This can be done using a univariate (i.e. only using past values of a single series) or a multivariate (i.e. using auxiliary data, namely explanatory variables in the form of additional time series) approach (Chatfield, 2000).

The following steps provide a general framework for conducting a time series analysis and forecast:

1. **Data Preparation**
   The first step in TS analysis is to collect and prepare the data. This involves checking for data quality, removing any missing values, and ensuring that the data is in the correct format. As with any ML task, it is also important to split the data into training (, evaluation) and test sets to evaluate the performance of the models. However, in TS analysis there exists an additional caveat. Since TS data is inherently sequential, the data cannot be randomly split into training and test set as it could lead to data leakage (i.e. using the past to predict the future). Instead, the data must split in a chronological order, with the training set consisting of the earlier observations and the test set consisting of the more recent observations. The same goes when using Cross-validation (CV). Here, a technique called *rolling window* CV is used, where the model is trained on a fixed window of the data and tested on the next window, which is then repeated on the following windows, essentially splitting the data into several non-overlapping buckets.

2. **Exploratory Data Analysis (EDA)**
   EDA is an important step in TS analysis as it provides insights into the underlying patterns and trends in the data. It includes:

   - Visualizing the data using several TS plots.
   - Analyzing the correlation between a TS and itself if lagged by a certain time period. This is done inspecting the autocorrelation function (ACF) and the partial autocorrelation function (PACF). The former measures the correlation between a TS and all of its lagged values, while the latter measures the correlation between a TS and its lagged values, while controlling for the effects of all shorter lags (i.e. using only the lags up to k as explanatory variables).
   - Decomposing the TS to identify trends, seasonality, and other patterns.
   - Testing for stationarity, i.e. checking if statistical properties such as mean, variance, and autocorrelation structure remain constant over time. One popular test is the Augmented Dickey-Fuller (ADF), taking a slightly different form depending on constant and trend (or their absence). The H_0 says: TS does possess a unit root, i.e. is not stationary. It estimates a lag length, which can be done computationally minimizing the AIC until the last lag is statistically significant (e.g. $p < 0.05$). If the TS is not stationary, techniques/methods such as differencing, ExpSm or SARIMA can be used to achieve (weak) stationarity.

3. **Feature Generation**
   This step involves creating new features from the original TS data or adding explanatory variables to it. These features can include time-specific (one-hot) encoded features, such as *day of the week* and *month of the year*, or lagged values of variables as well as external data such as economic factors. This step is not always necessary, mainly for enriching multivariate

methods, sometimes it is helpful for visualization (e.g. for showing distribution throughout the week/year etc.)[1].

4. **Model Selection and Training**
There are many different models suited for TS forecasting which can be divided into parametric (e.g. ExpSm, AR(I)MA and similar ones) and non-parametric models (Neural Networks, Regression). Many can be used either in a univariate or multivariate manner (Gautam and Singh, 2020). Two will be used and thus introduced here:

- **Exponential Smoothing** ExpSm is a popular method/model that is used to smooth as well as predict future values of a TS. It is a simple, yet effective, method that involves averaging the previous values of the TS to generate predictions for the future. While a simple Moving Average (MA) weighs all past observations equally, ExpSm decreases their weight the further they lie in the past. There are several types, namely Simple, Double, and Triple Exponential Smoothing (also known as Holt-Winters). The first one helps deal with noise (see Equation 1), while the second one additionally helps deal with a trend in the data (see Equation 2), the third additionally helps deal with seasonality (see Equation 3). This is done by introducing the smoothing parameters α, β and γ respectively. The forecast uses all variables estimated previously (see Equation 4) (Hyndman and Athanasopoulos, 2018).

$$l_t = \alpha y_t + (1-\alpha)\hat{y}_{t-1} \tag{1}$$
$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{2}$$
$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{3}$$
$$\hat{y}_{t+h} = l_t + hb_t + s_{t+h-m} \tag{4}$$

where:

y = observed value
$\hat{y}$ = predicted value
t = an index denoting a time period
l = level/noise component
b = trend factor
s = seasonal index
m = seasonal period
h = forecast at m periods ahead
h = number of steps to forecast ahead
α, β, γ = smoothing parameters to be estimated

- **SARIMA(X)** Seasonal Autoregressive Integrated Moving Average models are used for forecasting. It is a univariate method (like ExpSm) that includes the same components as ARIMA models, namely autoregression (AR), differencing (I), and Moving Average (MA), but also incorporates an additional seasonal component (S).

SARIMA models use, additionally to the three regular parameters (lowercase) used in ARIMA models, their seasonal parts (uppercase; with seasonal periods accounted for):

where:

1. This means it can sometimes be useful to take (parts of) this step before EDA.

$$\underbrace{(p,d,q)}_{\uparrow} \qquad \underbrace{(P,D,Q)_s}_{\uparrow}$$

Non-seasonal part of the model — Seasonal part of the model

p/P = number of autoregressive terms
d/D = order of differencing
q/Q = number of moving average terms
s = seasonal periods (e.g. 7 for weekly data)

Adding these to the ARIMA formula yields (with the same notation using lowercase symbols for the non-seasonal and uppercase symbols for the seasonal part):

$$\phi_p(B)\Phi_P(B^s)\Delta^d\Delta_s^D y_t = \theta_q(B)\Theta_Q(B^s)\epsilon_t \tag{5}$$

where:

ϕ = autoregressive operator
B = backshift operator (lag)
∇ = differencing operator
θ = moving average operator
t = time
y = time series (values)
ϵ = (white noise) error term

The (hyper-)parameters above have to be estimated using ACF and PACF (plots) and/or Grid Search over all (feasible) combinations.

SARIMAX (SARIMA with Exogenous Variables) is a multivariate method that can include other predictors/covariates in addition to the time series itself (which can improve the forecasting accuracy). It adds a vector of exogenous variables x to Equation 5.

5. **Hyperparameter Optimization**
Hyperparameter Optimization is, as with almost all ML tasks, an important step in TS forecasting. TS models often have a lot of hyperparameters which can be tuned (in contrast to model parameters, which are normally determined by the model during the training process). This is done either by inspection using mathematical/graphical tools or more often creating a combination of all possible/a range of hyperparameters and running the training process with these, evaluating them against a validation set. Hyperparameters often try to deal with (if necessary) noise, trend and seasonality.

6. **Model Evaluation**
A similar process used for evaluating hyperparameters is used when gauging overall model performance and comparing different models amongst each other. This is done all throughout the process of developing ML models. First, a baseline should be established or found, against which more sophisticated models can be compared. Since the target value in TS analysis/forecasting is continuous, metrics like Mean Absolute Error (MAE) or Root Mean Squared Error (RMSE) are used. The model is trained using training data and evaluated against held-out validation data (potentially using CV).

7. **Forecasting**
   The final step is to do the actual forecasting, i.e. predict future data. This can either be a single-step or multi-step prediction. If it is done against a test set, the result should be evaluated as shown above.

3. Analysis and forecast of store sales

3.1 Introduction to the data and their features

The provided data are daily sales of the individual stores, split by product family, of Favorita, an Ecuadorian chain of general stores (among other business areas), selling e.g. products of daily use and automobile parts. They span a time of about 4.5 years, from 2013 to mid-2017. Other auxiliary data are the count of articles on promotion, holidays and the oil price. Further features can be engineered from the dates, i.e. day of the week, day of the month, day of the year, month etc.

3.2 Exploratory Data Analysis

After loading the data, several plots were used to visualize trends, patterns/seasonality, peculiarities, i.e. summarize their main characteristics.

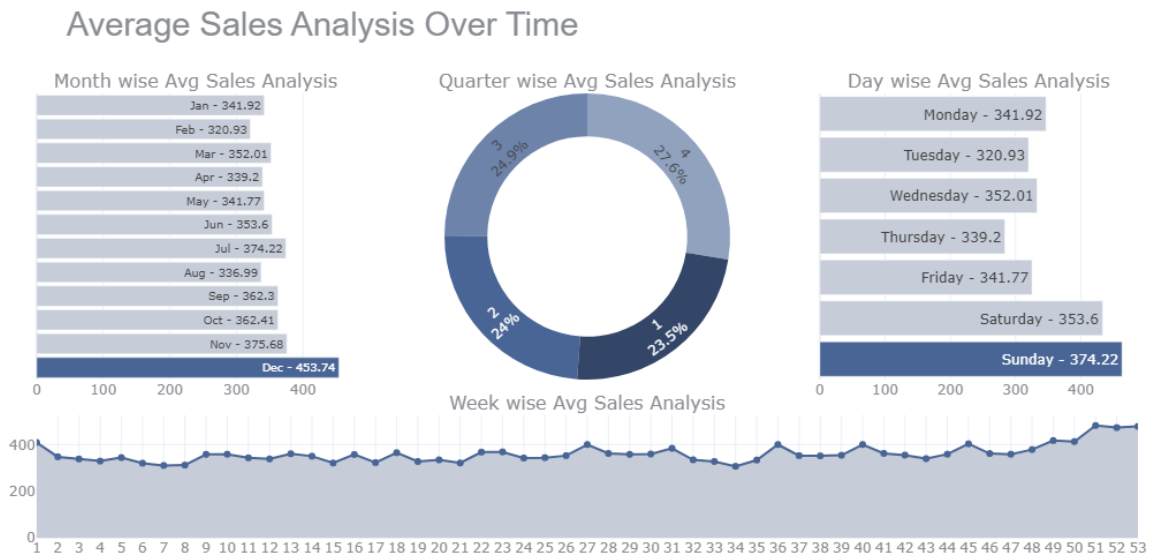

Figure 1: Quarterly, monthly, weekly and daily analysis of aggregated store sales

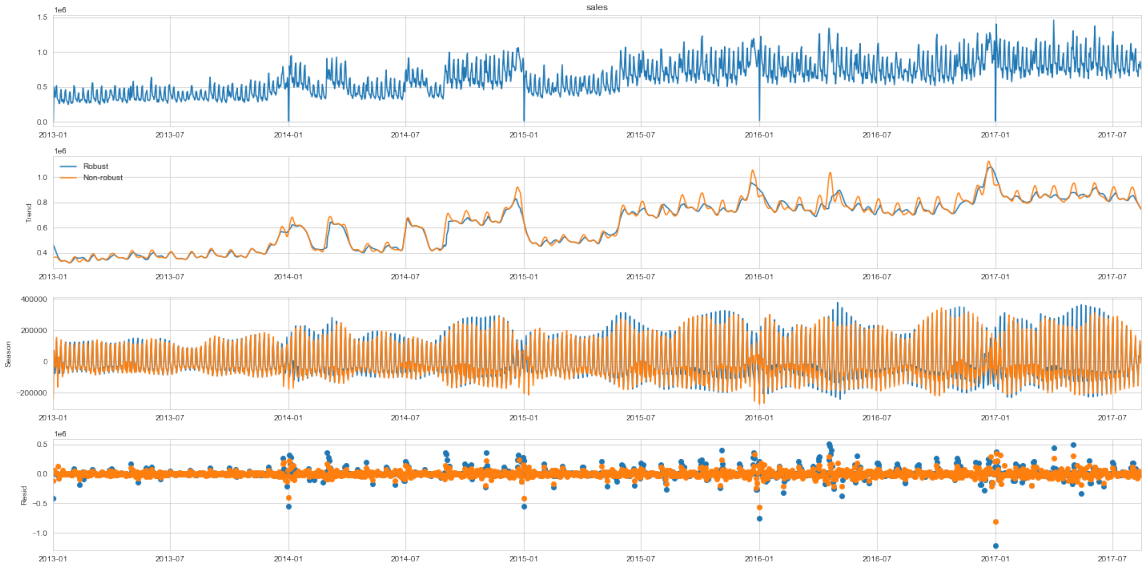

Figure 2: Quarterly, monthly, weekly and daily analysis of aggregated store sales

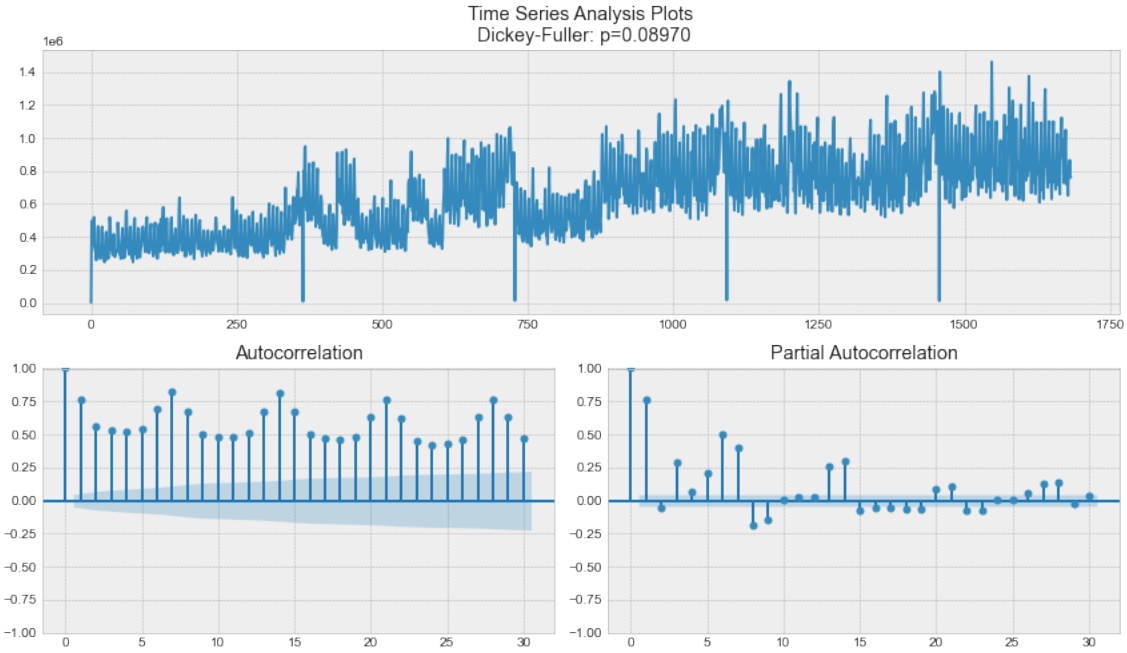

Figure 3: Quarterly, monthly, weekly and daily analysis of aggregated store sales

4. Results and discussion

5. Creating a Python package for parallel Hyperparameter Optimization using CV for TS

It often makes sense to use parallel programming (i.e. leveraging multiple processors/cores) when optimizing hyperparameters since it is computationally intensive, especially with CV. There exist libraries to do parallel optimization like Ray Tune[2], though they are not easily adaptable to all models and none were found to be suited (well) for optimizing hyperparameters of TS models, particularly for ExpSm and SARIMA. Furthermore, parallelization in Jupyter Notebooks is often cumbersome due to most parallelization packages requiring the function run in parallel to be imported from outside the Notebook.

All of this meant it was easier to construct something own. This led to the creation of a small convenience module which takes the data and a combination of all specified hyperparameters (among others) as input and then splits the data using CV (with regard to the special constraints for TS) and evaluates each split, using the RMSE. Where useful, this process can be repeated for each split to deliver more accurate results (e.g. in the case of ExpSm, SARIMA(X) contains no stochastic components/is deterministic). The output consists of the (sorted) results for all combinations of hyperparameters tested. So far, ExpSm and SARIMA(X) are supported, but the module can be easily extended to use other methods.

The package looks as follows:

```
def optimize_hyperparams(hyperparams: list, data, func: str, n_steps=1, n_splits=10,
    runs_per_split=10):

    data = data.reset_index(drop=True)
    tscv = TimeSeriesSplit(n_splits = n_splits, test_size=n_steps)

    rmse_split = list()
    for train_index, test_index in tscv.split(data):
        cv_train, cv_test = data.iloc[train_index], data.iloc[test_index]
        rmse = list()
        for i in range(0, runs_per_split):
            try:
                preds = globals()[func](hyperparams, cv_train, n_steps)
            except:
                print(hyperparams)
                traceback.print_exc()
                return

            true_values = cv_test.values
            rmse.append(sqrt(mean_squared_error(true_values, preds)))

        rmse_split.append(np.mean(rmse))

    rmse_all = dict()
    rmse_all[str(hyperparams)] = round(np.mean(rmse_split), 2)
    return rmse_all
```

It is used as follows:

```
from ts_hyperparam_opt import parallel_hyperparameter_optimization as pho

if __name__ == '__main__':
    freeze_support()
```

2. https://docs.ray.io/en/latest/tune/index.html

```
    results_exp_smoothing = pho.sort_results(process_map(partial(pho.
    optimize_hyperparams, data=df_sales_train, func="exp_smoothing", n_steps=15,
    runs_per_split=10), params_exp_smoothing))
```

Producing output looking like this:

```
[{"{'trend': 'add', 'damped_trend': True, 'seasonal': 'mul', 'periods': 7, '
    use_boxcox': True, 'remove_bias': True}": 108003.45},
 {"{'trend': None, 'damped_trend': False, 'seasonal': 'mul', 'periods': 7, '
    use_boxcox': True, 'remove_bias': True}": 108611.14},
 {"{'trend': 'mul', 'damped_trend': True, 'seasonal': 'mul', 'periods': 7, '
    use_boxcox': True, 'remove_bias': False}": 108993.4}, [...] ]
```

The package has been published on GitHub (https://github.com/nick2202/ts-hyperparam-opt) and PyPI (https://pypi.org/project/ts-hyperparam-opt/) and can be installed using

```
$ pip install ts-hyperparam-opt
```

6. Conclusion

References

Chris Chatfield. *Time-series forecasting*. CRC press, 2000.

Anjali Gautam and Vrijendra Singh. Parametric versus non-parametric time series forecasting methods: A review. *Journal of Engineering Science and Technology Review*, 13:165–171, 06 2020. doi: 10.25103/jestr.133.18.

Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

Appendix

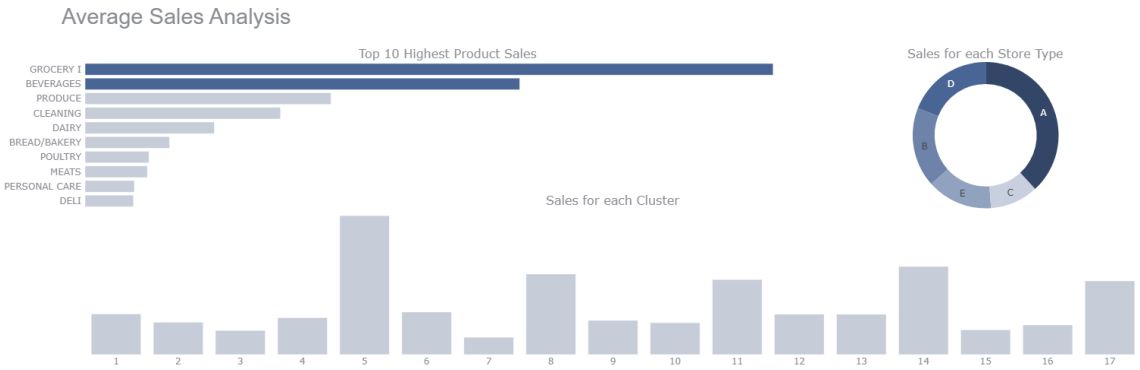

Figure 4: XXX

Figure 5: XXX

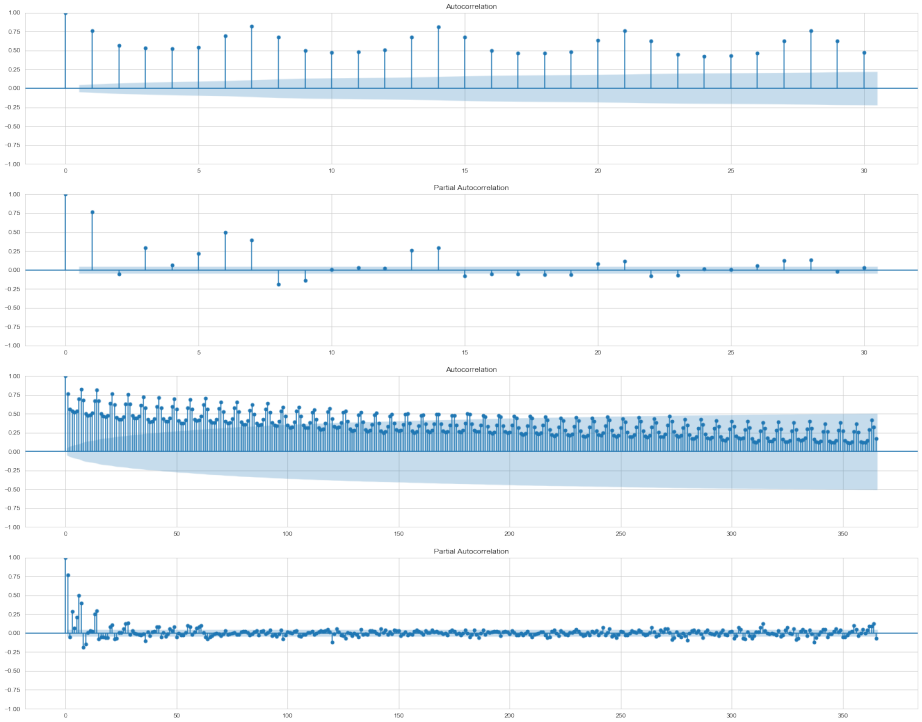
Autocorrelation
Partial Autocorrelation
Autocorrelation
Partial Autocorrelation

Figure 6: XXX

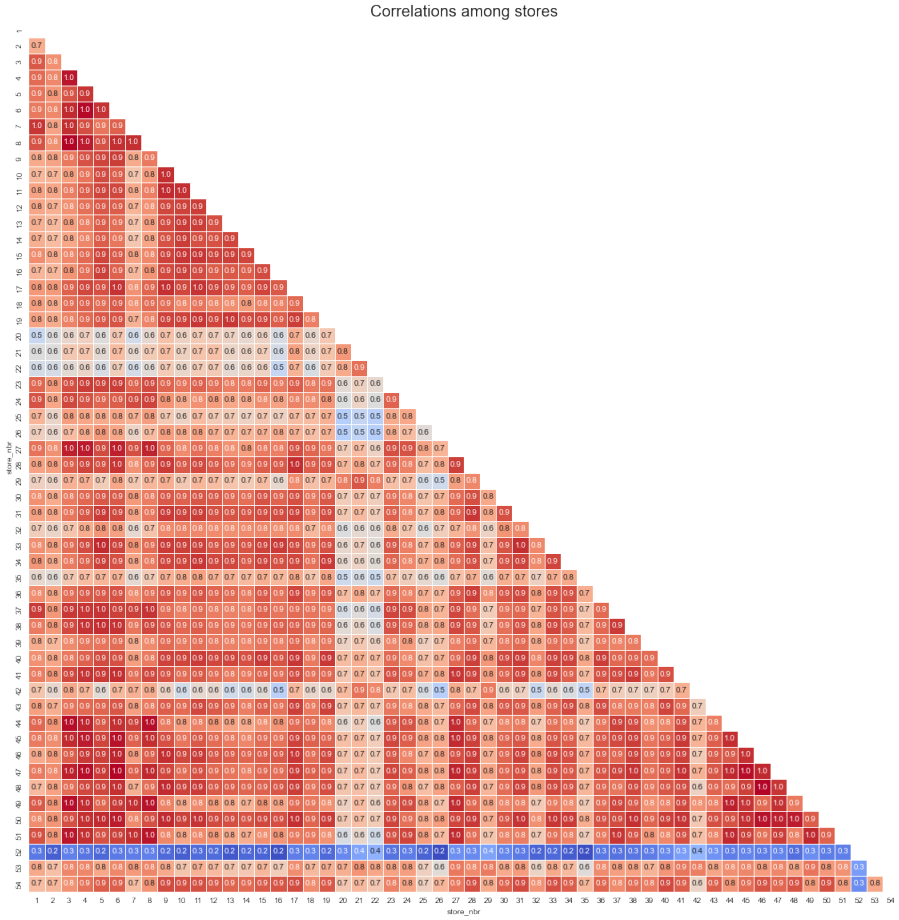

Figure 7: XXX