Scripts Description

[1. Fraud Detection 1](#_Toc21099681)

[2. Future Sales 1](#_Toc21099682)

[3. Demand Forecasting 2](#_Toc21099683)

1. Fraud Detection

Very first approach to predict if a transaction is a fraud or not. Full\_train- / Full\_test\_data sets were built using joins with train\_- / test\_- id sets. Then the “meaningful” variables (mngful\_variables) have been gathered separately. So were the numeric variables (train\_numeric) – they have been standardized with StandardScaler as well. Log-transformation of the dependent variable (TransactionAmt) has been implemented to assure normal distribution.

Finally, Logistic Regression has been chosen to make first prediction attempt. Its accuracy has been evaluated via:

* Confusion matrix,
* Sensitivity,
* Specificity,
* ROC Curve,
* AuC (Area Under Curve)

1. Future Sales

In the beginning a triple loop (through years, quarters and blocks) has been run to fill the MQ\_Dict dictionary. It will be used to map each month/year to respective quarter (column: date\_quarter) (to check the season impact on sales). The data has been aggregated to build a model that could be later tested on the test set. Test set has been assigned with date\_quarter = 4 and date\_block\_num = 34.

There were 4 regressors used to calculate the predictions:

* Dummy Regressor,
* Linear Regressor,
* Random Forest Regressor,
* ANN Regressor,

The ANN has been visualized using ann\_visualizer library. Later, RMSE has been calculated to compare all of the models. Basic attempt on Grid Search has been taken (with number of estimators). For exploratory analysis an object *single\_values* has been prepared. Uncommenting the XKCD-plot section leads to outputting nicely visualized time series for shop&item pair specified as arguments.

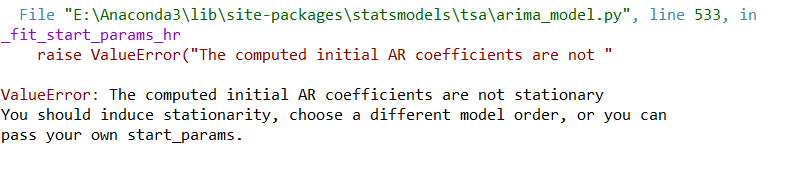
1. Demand Forecasting

First of all, the script show MsgBox with some basic definitions (for now: stationarity of time series). Then it open some links I used as inspiration for further analysis. Then the code is divided into two parts.

First one – *“Dynamic assignment of ARIMA parameters”* loops through all of the unique shop & item combinations and extracts time series for one of unique 441 combinations. Then, Augmented Dickey Fuller test is carried out for regular, differentiated and twice differentiated time series to find out if the series have to be differentiated or not. The autoregressive term (p) is assigned by first value order that gets into the confidence interval of PACF plot, whereas the moving average term (q) is evaluated using the same methodology for the ACF plot. Each p,d,f parameters are stored in ARIMA\_Summary dataframe. Section starts and ends with printing the respective datetimes.

Second part – *“Execution of 3 month predictions for each item&shop pair”* carries out 90 days predictions for all of the time series using previously estimated time series. As it is (currently) quite demanding in computational terms, the predictions are not only saved to the working directory as ARIMA\_forecasts.csv file, but also the predicted end time is calculated by using the average time spent on previous iterations multiplied by remaining ones.

Currently code breaks on the second part, due to following error:



1. Walmart

The script begins with building two dictionaries: DICT\_weekend and DICT\_weekday. Their aim is to either to create a dummy variable (a day is on the weekend or not) or to encode categorical variable weekday. Secondly, all of the products within one transaction (‘ScanCount’ variable] are separated into two columns – ProductsReturned and ProductsPurchased. Moreover, each transaction is assigned with total number of products within one visit (‘ProductsPerVisit’).

The very first attempt uses five variables from train\_columns list:

* Products Purchased,
* Products Returned,
* Weekday,
* ProductsPerVisit,
* DepartmentDescription

and following models:

* Bayes Classifier,
* Random Forest Classifier,
* Logistic Regression Classifier,
* XGBoost Classifier,
* K-Nearest Neighbors Classifier,
* Artificial Neural Networks Classifier
* ~~SVC (Support Vector Classifier)~~

Confusion matrix is built for all of the classifiers and the accuracy object has been built to evaluate correct predictions (sum of all values on the diagonal line divided by sum of all values).

On the bottom an object *count\_exec\_time* has been created. Taking a *cmdstring* (recommended to take it from a text file and assigning to string variable) and optionally *title* as an input it outputs how much time a code has been executing.