Demand Forecast Early Research Project

Pletnev Alexander

December 2018

Contents

1 Executive Summary

In this research project we investigate the novel method of predicting the required quantity of item using additional variable: future flag. We apply 8 state-of-the-art methods (Adaboost; Kernel Regression; Lasso; Neural Network;

Poisson Regression; Random Forest Regression; Ridge; SVR), 3 types of transformation (no transformation, min-max scaling, log transformation) and 2 types of fitting (all items and item by item).

We measure the resulting accuracy of prediction using SMAPE and then generate the feature importance for given methods.The tests were run on 2 given data sets.

We have found out that Kernel Regression, Random Forest (one by one fitting, log transformation) and Neural Network (all items fitting, log transformation) perform better among 8 applied methods.

2 Introduction

Demand planners in the industry use commonly tools such as Microsoft Excel to forecast. Although Excel has many strengths and is a flexible tool, it is not well-suited whenever dealing with large amounts of time series. Demand planners struggle reviewing and manually adjusting those forecasts. As a result, they rely on aggregate level adjustments. In many cases, the aggregation is prone to mistakes and poorly reflects the underlying statistical process. Data Science can mitigate this process and make the prediction more precise.

Usually, the demand forecasting setting is to estimate q(t,i) - the needed amount of a particular item i by the end of month t. For many supply chain problems, the definition is extended to estimate q(t,i,f) - the total quantity to be delivered by the end of month t through orders made f months in advance

3 Methodology

3.1 Experimental/sampling design

The study was performed on the data from commercial partner of the university. Several data sets were analyzed, which are labeled by "stage 1" and "stage 3". The incoming data sets had following columns:

- rpd - represents the demand date of each item. Currently we aggregate the daily demand into monthly demand. In stage 1 it ranges from 1 to 45, in stage 3 - from 108 to 151.
- item - represents the index of the items. There are 857 items in stage 1 and 836 items in stage 3.
- quantity - the actual quantity of required item in particular month
- future flag - represents the observed demand by the end of (rpd - future flag) month.

3.2 Data analysis

The following 8 methods were chosen to predict future value of the data: Adaboost; Kernel Regression; Lasso; Neural Network; Poisson Regression; Random Forest Regression; Ridge; SVR.

The prediction works as follows: firstly, we fit the incoming x months. Then we predict the following $x + 1$ month and compare it to the actual value. Afterwards we increase the x by 1 and iterate until it reaches 45.

To determine the starting period from which we need to predict he data, the following plot was generated. Please see Figure 1 to see an example of generated plot

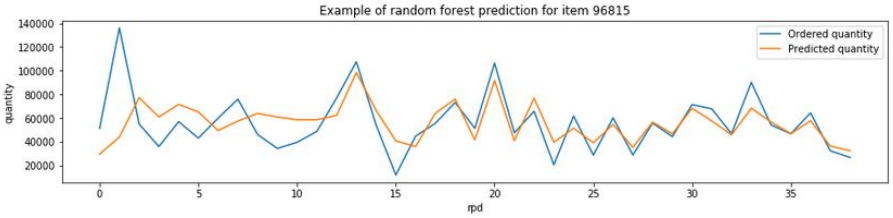

Figure 1: Example of predicting the future demand with future flag = 0

As we can notice, at the beginning of the period, the difference between actual value and prediction is quite big. It is gradually reduced when we increase the predicting month, thus increasing the amount of fitting data. It was determined, that we start to predict the data starting from month 34.

3.3 Measurement

We choose Symmetric Mean Absolute Percentage Error (SMAPE) to evaluate the result. Let y_{it} and $\hat{y_{it}}$ be the actual and predicted quantity of item i during time window t. The SMAPE is defined as:

$$\text{SMAPE} = \frac{1}{\text{n}} * \sum_{\text{i=1}}^{\text{n}} \frac{2 * |y_{it} - \hat{y}_{it}|}{|y_{it}| + |\hat{y}_{it}|} \ (\text{if } |y_{it}| + |\hat{y}_{it}| != 0)$$

4 Results

Let's analyze the key results that has been accomplished:

1. Provided the necessary functioning code to use diagonal feeding to train any method implementing the scikit-learning api.

   Please see Appendix A for the implementation of the functioning code.

```
df = pd.read_csv("demand_out_encoded_stage_3.csv",
    header=0, index_col=None)
exp_X,exp_Y = create_X_y(df)
exp_X = exp_X.applymap(int)
exp_Y = exp_Y.applymap(int)
regr = RandomForestRegressor(max_depth=3,n_estimators
    =100,n_jobs = -1)
y_predicted_rf_all, y_true_rf_all=
    prediction_all_values(exp_X,exp_Y,regr)
```

Example of usage of this function

2. Applied diagonal feeding and generated predictions using 8 methods: Adaboost; Kernel Regression; Lasso; Neural Network; Poisson Regression; Random Forest Regression; Ridge; SVR (see Appendix B for an example of using this methods).

3. Applied diagonal feeding and generated predictions using 3 types of transformation: No transformation; min-max scaling; log transformation.

```
y_predicted, y_true= prediction_all_values(exp_X,
    exp_Y,ab)
y_predicted, y_true= prediction_all_values_minmax(
    exp_X,exp_Y,ab)
y_predicted, y_true= prediction_all_values_logtransf(
    exp_X,exp_Y,ab)
```

Example of generating the predictions using no transformation, min-max scaling, log transformation using Adaboost method

4. Applied diagonal feeding and generated predictions using 2 types of fitting the input: Fit and predict item by item; fit all items and predict item by item.

```
ada_tree_backing = DecisionTreeRegressor(max_features
    ='sqrt', splitter='random', min_samples_split=3,
    max_depth=3)
ab = AdaBoostRegressor(ada_tree_backing,
    learning_rate=0.1, loss='square', n_estimators
    =100)
y_predicted, y_true= prediction_all_values(exp_X,
    exp_Y,ab)
y_predicted, y_true= prediction_item_by_item_fit(
    exp_X,exp_Y,ab)
```

Example of generating the predictions using "Fit and predict item by item", "Fit all items and predict item by item" using Adaboost method

5. Generated histogram with SMAPE distribution (Appendix 14) and prediction scatter plot (Appendix 15) for foregoing 48 models

6. Applied feature importance generation for foregoing 48 models (Figure 2).

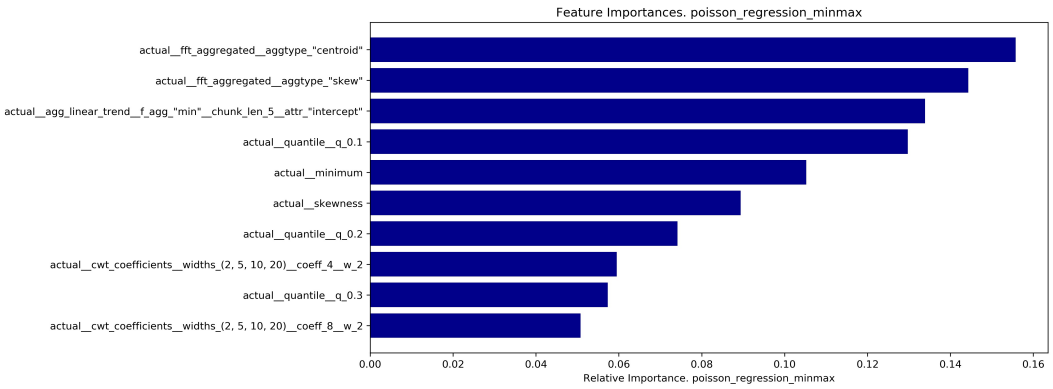

Figure 2: Example of generated feature importance for particular model

Please see Appendix 16 for a scatter plot of the generated feature importance for particular model

7. Excel file with comparison of performance measure of models for different data sets (Figure 3).

Mean SMAPE

Type of fit	Type of transformation	Adaboost	Kernel	Lasso	Neural Network	Poisson	Random Forest	Ridge	SVR
One item by item	No transformation	0.47	0.44	0.37	0.67	0.37	0.37	0.37	0.55
	Min-max scaling	0.46	0.42	0.59	0.51	0.37	0.37	0.42	0.6
	Log transformation	0.43	0.46	0.38	0.73	0.37	0.34	0.4	0.54
All items	No transformation	1.1	0.4	0.36	1.13	0.4	0.62	0.36	0.86
	Min-max scaling	1.1	0.38	1.14	0.95	0.39	0.61	0.43	1.73
	Log transformation	0.55	0.42	0.41	0.37	0.35	0.37	0.4	0.56

INPUT	
Future flag	0
Stage	3

Figure 3: Generated dashboard with mean SMAPE for each method, when future flag = 0 and for the data set from stage 3

5 Discussion and key findings

1. For foregoing models prediction of only future flag = 0 shows appropriate SMAPE ($> 50\%$).

Figure 4 shows the comparison of dashboards with different future flag. Figure 5 shows the distribution of SMAPE for various future flags.

Mean SMAPE

Type of fit	Type of transformation	Adaboost	Kernel	Lasso	Neural Network	Poisson	Random Forest	Ridge	SVR
One item by item	No transformation	0.65	0.77	0.73	0.81	0.69	0.65	0.73	0.63
	Min-max scaling	0.65	0.65	0.71	1.05	0.68	0.64	0.65	0.72
	Log transformation	0.62	0.76	0.66	0.84	0.66	0.63	0.69	0.63
All items	No transformation	1.16	0.99	0.78	1.2	0.69	0.84	0.78	0.91
	Min-max scaling	1.13	0.64	1.2	0.74	0.71	0.84	0.75	1.75
	Log transformation	0.89	0.7	0.68	0.65	0.61	0.6	0.68	0.71

INPUT

Future flag 1

Stage 3

Figure 4: We can notice that the resulted SMAPE for all models with future flag = 1 is greater than 50%.

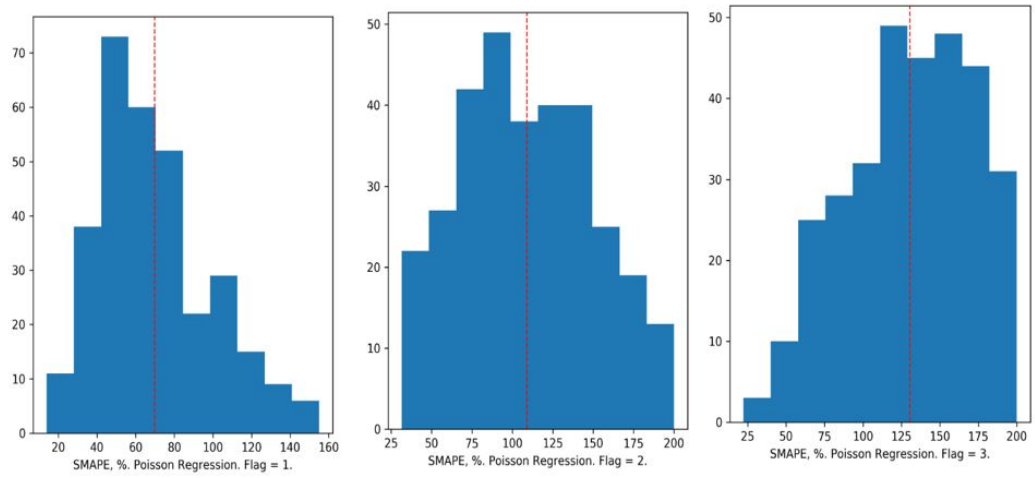

Figure 5: If we increase future flag, the resulting SMAPE will also increase for all models.

2. When we fit all items without the transformation the prediction seems to be heavily shifted to make predictions with larger value.

   Figure 6 gives the example of scatter plot of prediction vs actual, which captures this trend.

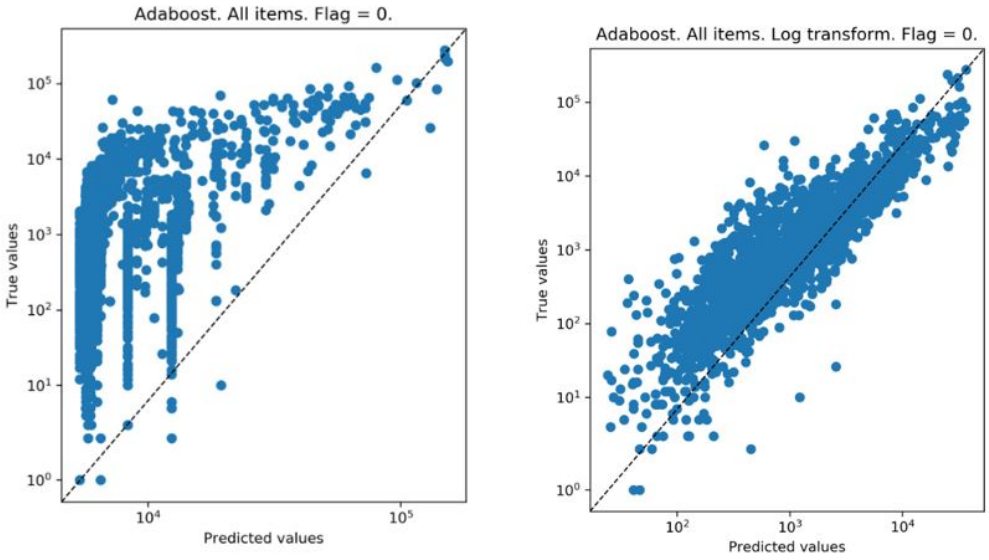

Figure 6: Example of this occurrence. Same phenomenon can be traced in other methods

Possible explanation is that there are items with large volume which shift the prediction in this direction. We can mitigate this my

3. Only Kernel method shows relatively low volatility when estimating SMAPE for different data sets. Other methods show spread around 15%. Also, using Log transformation decreases the volatility (Figure 7).

Δ Mean SMAPE for 2 Stages

Type of fit	Type of transformation	Adaboost	Kernel	Lasso	Neural Network	Poisson	Random Forest	Ridge	SVR
One item by item	No transformation	0.19	0.04	0.2	0.17	0.15	0.14	0.2	0.52
	Min-max scaling	0.2	0.05	0.27	0.24	0.15	0.14	0.21	0.16
	Log transformation	0.1	0	0.12	0.06	0.1	0.12	0.08	0.11
All items	No transformation	0.17	0.02	0.25	0.16	0.15	0.18	0.25	0.21
	Min-max scaling	0.14	0.03	0.15	-0.38	0.16	0.2	0.23	0.09
	Log transformation	0.04	0	0	0.02	0.12	0.06	0.01	-0.12

INPUT	
Future flag	0
Original Stage	1
Compared Stage	3

Figure 7: When we compare mean SMAPE from different stages we can notice that only when we use Kernel Regression the delta is relatively small. We can pay heed to log transformation, where difference is smaller

4. Usually log transformation shows better SMAPE compared to no transformation or min-max scaling (Figure 8).

Mean SMAPE

Type of fit	Type of transformation	AVERAGE	Adaboost	Kernel	Lasso	Neural Network	Poisson	Random Forest	Ridge	SVR
One item by item	No transformation	**0.65**	0.66	0.48	0.57	0.84	0.52	0.51	0.57	1.07
	Min-max scaling	**0.65**	0.66	0.47	0.86	0.75	0.52	0.51	0.63	0.76
	Log transformation	**0.54**	0.53	0.46	0.5	0.79	0.47	0.46	0.48	0.65
All items	No transformation	**0.83**	1.27	0.42	0.61	1.29	0.55	0.8	0.61	1.07
	Min-max scaling	**0.92**	1.24	0.41	1.29	0.57	0.55	0.81	0.66	1.82
	Log transformation	**0.45**	0.59	0.42	0.41	0.39	0.47	0.43	0.41	0.44

INPUT
Future flag 0
Stage 1

Figure 8: We can notice that the average taken SMAPE is greater when we use log transformation. In the highlighted cells the log transformation performed better.

5. Adaboost, Ridge, Neural Network, Lasso, Poisson Regression don't show appropriate SMAPE ($> 50\%$). An exception is fitting all items with log Transformation, where mean SMAPE is around 45% (Figure 9).

Mean SMAPE

Type of fit	Type of transformation	AVERAGE	Adaboost	Kernel	Lasso	Neural Network	Poisson	Random Forest	Ridge	SVR
One item by item	No transformation	**0.65**	0.66	0.48	0.57	0.84	0.52	0.51	0.57	1.07
	Min-max scaling	**0.65**	0.66	0.47	0.86	0.75	0.52	0.51	0.63	0.76
	Log transformation	**0.54**	0.53	0.46	0.5	0.79	0.47	0.46	0.48	0.65
All items	No transformation	**0.83**	1.27	0.42	0.61	1.29	0.55	0.8	0.61	1.07
	Min-max scaling	**0.92**	1.24	0.41	1.29	0.57	0.55	0.81	0.66	1.82
	Log transformation	**0.45**	0.59	0.42	0.41	0.39	0.47	0.43	0.41	0.44

INPUT
Future flag 0
Stage 1

Figure 9: Highlighted cells show, that this particular model performed with SMAPE greater than 50%.

6. Kernel regression shows consistent good results when using all types of transformation and fitting (Figure 10).

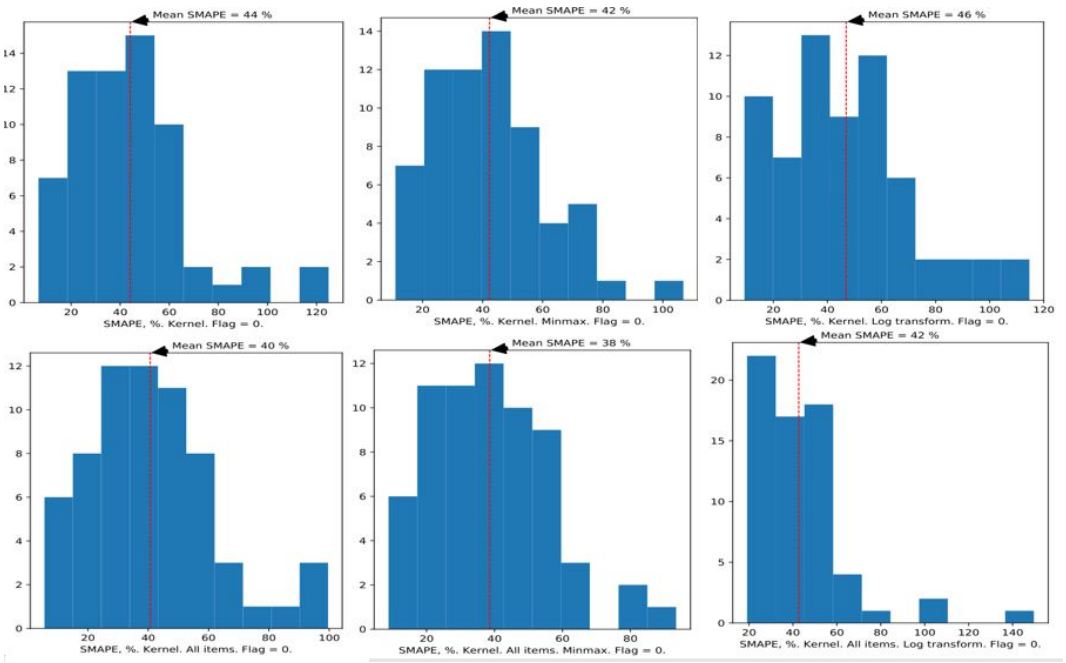

Figure 10: Stage 3. All types of Kernel model show good SMAPE

7. "All items fit" has greater SMAPE by around 5% (Figure 11).

Mean SMAPE

Type of fit	Type of transformation	AVERAGE	Adaboost	Kernel	Lasso	Neural Network	Poisson	Random Forest	Ridge	SVR
One item by item	No transformation	**0.45**	0.47	0.44	0.37	0.67	0.37	0.37	0.37	0.55
	Min-max scaling	**0.47**	0.46	0.42	0.59	0.51	0.37	0.37	0.42	0.6
	Log transformation	**0.46**	0.43	0.46	0.38	0.73	0.37	0.34	0.4	0.54
All items	No transformation	**0.65**	1.1	0.4	0.36	1.13	0.4	0.62	0.36	0.86
	Min-max scaling	**0.84**	1.1	0.38	1.14	0.95	0.39	0.61	0.43	1.73
	Log transformation	**0.43**	0.55	0.42	0.41	0.37	0.35	0.37	0.4	0.56

INPUT	
Future flag	0
Stage	3

Figure 11: Highlighted cell shows, where SMAPE is lesser("item by item fit" vs "all items fit"). One can see the prevalence of highlighted cells in the section of "item by item fit"

8. Random Forests Regression shows good results only when using one item fitting or employing log transformation (Figure 12).

Mean SMAPE

Type of fit	Type of transformation	Random Forest Stage 1	Random Forest Stage 3
One item by item	No transformation	0.51	0.37
	Min-max scaling	0.51	0.37
	Log transformation	0.46	0.34
All items	No transformation	0.8	0.62
	Min-max scaling	0.81	0.61
	Log transformation	0.43	0.37

Figure 12: Table with results of running Random Forest regression on different data sets

9. Most methods, which perform well, depend on very similar values (Figure 13). They are:
   - quantiles / min / median of actual time series
   - centroid / skew derived from FFT for actual time series

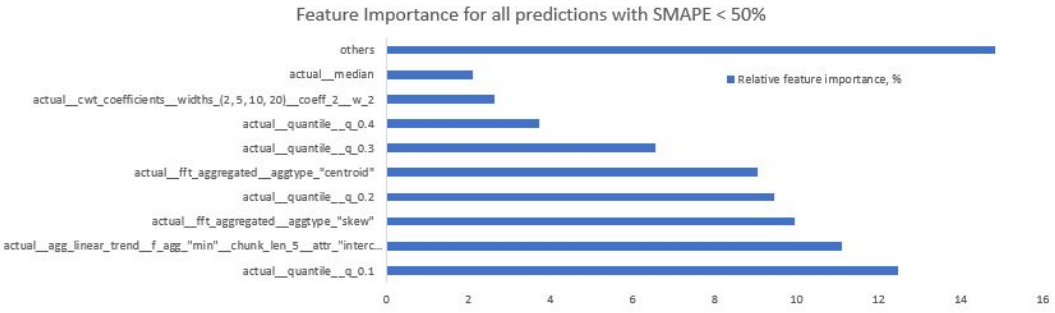

Figure 13: Top 10 features stand out for 84% of all relative feature importances.

6 Conclusions

The initial goal was to:

- Implement functioning code to use diagonal feeding. This task was completed.
- Choose 5 models, apply diagonal feeding and compare results with Ensemble of Gradient Boosting, Random Forest, SVM Regression, ARIMA, ARIMAX, Kernel Regression. We have chosen the following 4 state-of-arts models to compare results to: Adaboost, Lasso, Ridge, Neural Network.

Also the feature importance for given methods was generated as well as applied different techniques of transformation and types of fitting the data. Given more time I would like to focus on:

- Dedicate more time to deeply investigate, why certain methods perform well and others don't. For this I need to thoroughly understand the method and the appropriate applications of the given method
- It would be interesting to see analyze which methods are more appropriate when we now the features of given item. Thus, we can utilize the best performing method and improve the accuracy of prediction

It would be great that in further analyzes I would be able to cover this topics

7 Recommendations

We have determined that Kernel Regression and Random Forest perform better compared to other analyzed methods. Thus, it is recommended to chose them for further analysis. Additionally, when employing Random Forest it is advisable to fit item by item and use log transformation in order to reach the best possible prediction. Neural Network with log transformation and fitting all items at once performs well as well.

Also in order to improve the accuracy of prediction, we can predict items, which have the resembling same quantiles, min, median, centroid derived from FFT, skew derived from FFT as those items, which are given in stage 1 and 3.

Appendices

A

Example of implemented function, which uses diagonal feeding and predicts given items by fitting all items at the same time and uses api from scikit-learning

```
def prediction_item_by_item_fit(exp_X_trunc,exp_Y_trunc,
    regrr):
     y_predicted = defaultdict()
```

```
    exp_Y_trunc = exp_Y_trunc.reindex(columns=['rpd','
        item','␣1-0','␣2-0','␣3-0','␣4-0','␣2-1','␣3-1','␣
        3-2','␣4-1','␣4-2','␣4-3'])
    y_true = defaultdict()
    for it in set(exp_X_trunc.item):
        y_predicted[it] = [[],[],[],[],[],[],[],[],[],[]]
        y_true[it] = [[],[],[],[],[],[],[],[],[],[]]
    for i in range(33,45):
        for ii in range(1,11):
            for it in set(exp_X_trunc.item):
                regr = regrr
                X = exp_X_trunc.loc[exp_X_trunc['rpd'].
                    isin(list(range(1,i+1)))]
                X = X.loc[X['item']==it].iloc[:,2:12]
                Y = exp_Y_trunc.loc[exp_Y_trunc['rpd'].
                    isin(list(range(1,i+1)))]
                Y = Y.loc[Y['item']==it].iloc[:,ii+1]
                regr.fit(X,Y)
                X_test = exp_X_trunc.loc[exp_X_trunc['rpd
                    '].isin(list(range(i+1,i+2)))]
                X_test = X_test.loc[X_test['item']==it].
                    iloc[:,2:12]
                if (X_test.size==0):
                    continue
                y_predicted[it][ii-1].append(regr.predict
                    (X_test)[0])
                true_ = exp_Y_trunc.loc[exp_Y_trunc['rpd'
                    ].isin(list(range(i+1,i+2)))]
                true_ = true_.loc[true_['item']==it].iloc
                    [:,ii+1].values[0]
                y_true[it][ii-1].append(true_)
    for x in y_predicted.keys():
        y_predicted[x] = pd.DataFrame(np.array(
            y_predicted[x]).clip(min=0).transpose())
        y_true[x] = pd.DataFrame(np.array(y_true[x]).
            transpose()).applymap(float)
    return y_predicted,y_true
```

B

Example of generating the predictions for these 8 models

```
ada_tree_backing = DecisionTreeRegressor(max_features='
    sqrt', splitter='random', min_samples_split=3,
    max_depth=3)
```

```
ab = AdaBoostRegressor(ada_tree_backing, learning_rate
    =0.1, loss='square', n_estimators=100)
y_predicted, y_true= prediction_item_by_item_fit(exp_X,
    exp_Y,ab)
kr = KernelRidge(alpha=1.0)
y_predicted, y_true= prediction_item_by_item_fit(exp_X,
    exp_Y,kr)
ls = Lasso(alpha=0.1)
y_predicted, y_true= prediction_item_by_item_fit(exp_X,
    exp_Y,ls)
rg =Ridge(alpha=1.0)
y_predicted, y_true= prediction_item_by_item_fit(exp_X,
    exp_Y,rg)
svrr = SVR(C=1.0, epsilon=0.2)
y_predicted, y_true= prediction_item_by_item_fit(exp_X,
    exp_Y,svrr)
nn = MLPRegressor(activation= 'tanh', hidden_layer_sizes=
     (20,), solver= 'lbfgs')
y_predicted, y_true= prediction_all_values(exp_X,exp_Y,nn
    )
pr = xgboost.XGBRegressor( params={"objective": "count:
    poisson"},num_boost_round=100000,
              early_stopping_rounds=5,verbose_eval=False,
                  n_jobs = -1)
y_predicted, y_true= prediction_all_values_logtransf(
    exp_X,exp_Y,pr)
regr = RandomForestRegressor(max_depth=3,n_estimators
    =100,n_jobs = -1)
y_predicted, y_true= prediction_item_by_item_fit(exp_X,
    exp_Y,regr)
```

C

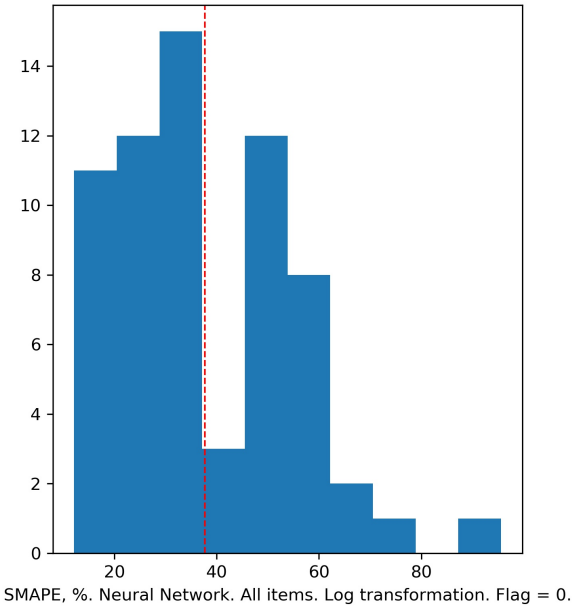

Figure 14: Example of generated histogram with SMAPE distribution. Here we can see, that the chosen method performs quite well with mean SMAPE = 38%

D

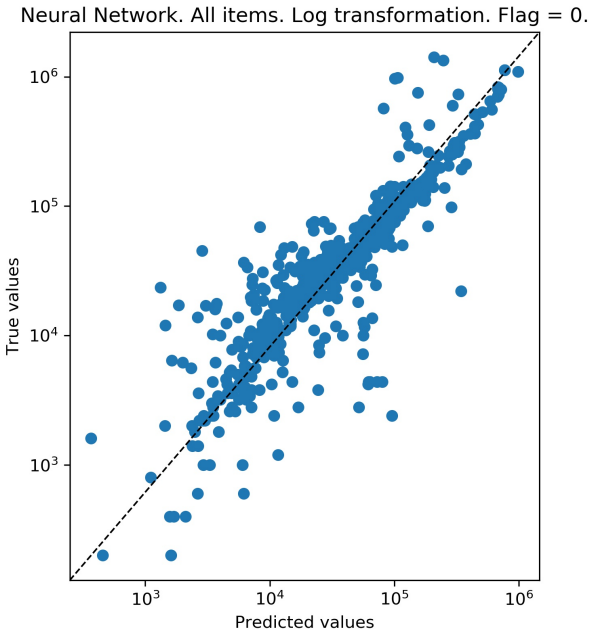

Figure 15: Example of generated scatter plot

E

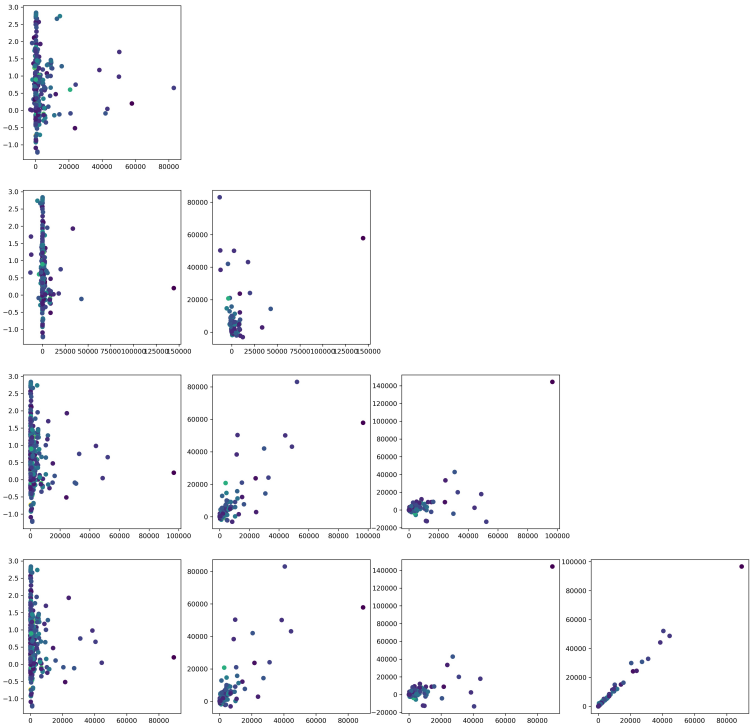

Figure 16: Example of generated feature importance for particular model