M5 Forecast - Accuracy

Big Data Analysis Exam

Date: 10th June 2020

Maria, Mads, Andy and Emil

All group members have contributed equally to the project.

M5 Forecasting - Accuracy

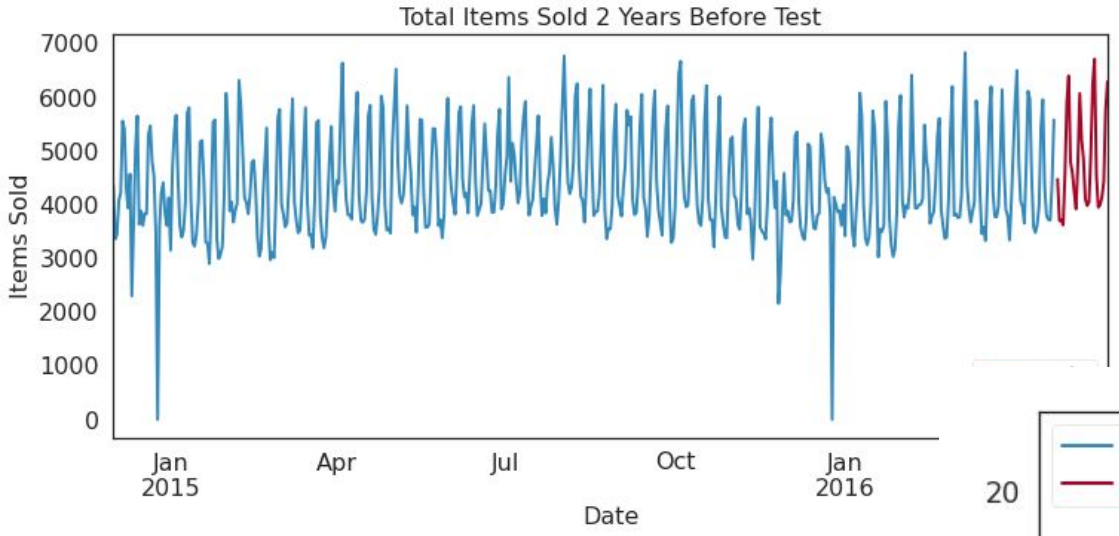
Objective: Predict 28 days of item sales into the Future for 3049 items

Data:

- Historical data for past 1941 days.
- Calendar
  - Public holidays
  - SNAP days
  - etc.
- Prices
  - Week to week prices for all items.

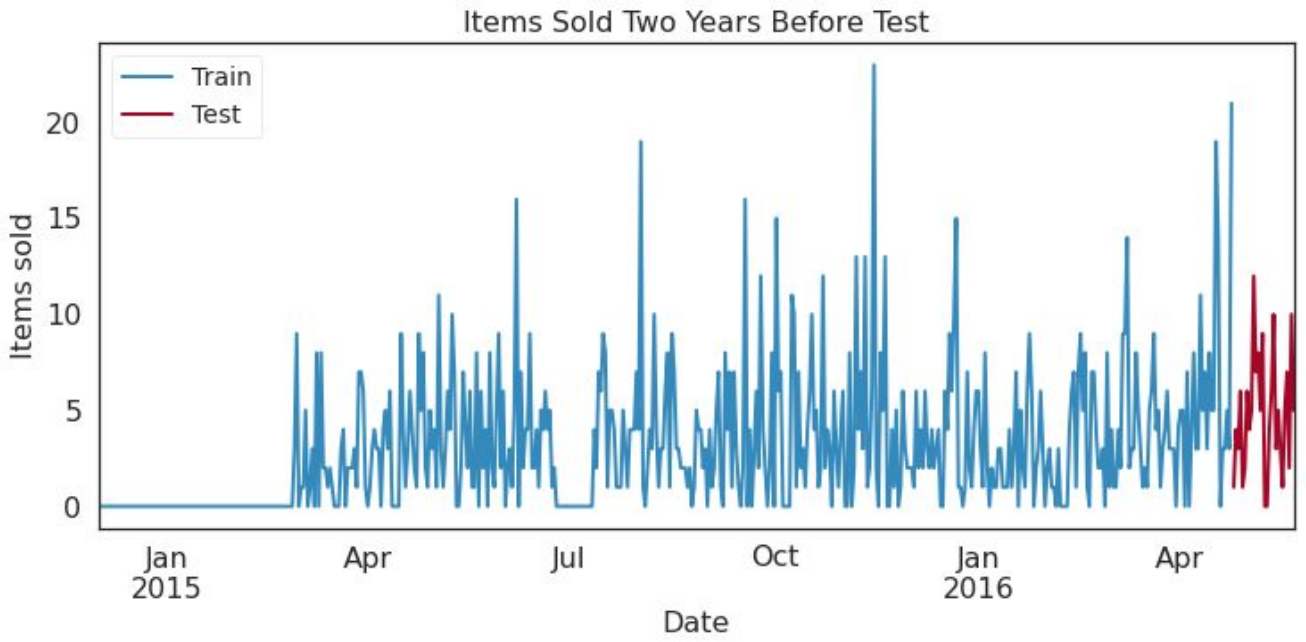
Motivation: 50.000$

M5 Forecasting - Accuracy



Store level:
Can observe seasonality which we can ideally reproduce when summing all individual time series

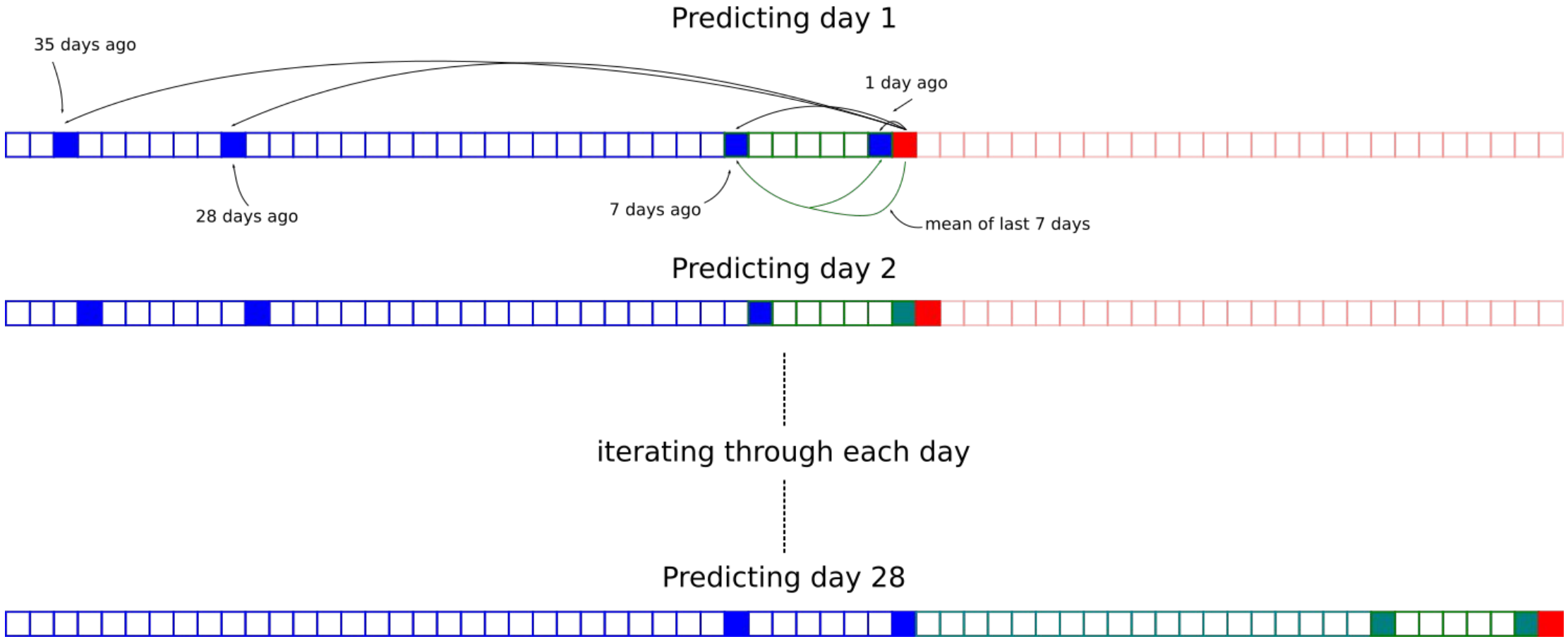
Item level:
time series of individual items
Noisy and low counts
product sold out?



Chosen Models

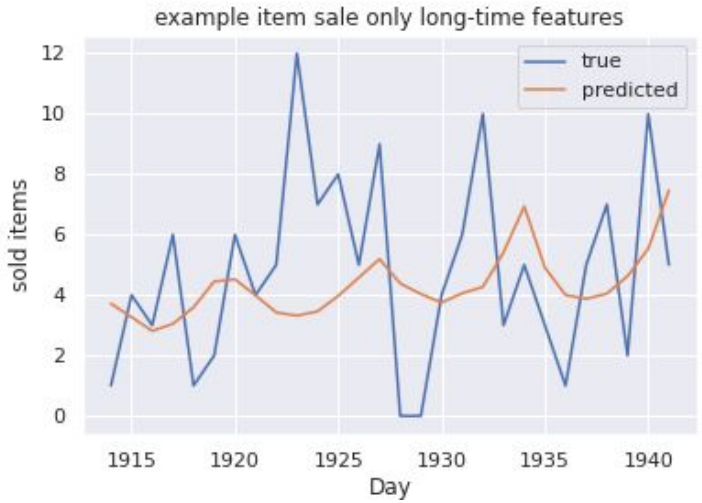
- Gradient Boosted Trees (LightGBM)
- Long Short-Term Memory (Keras)
- Playtime - Graphs!!

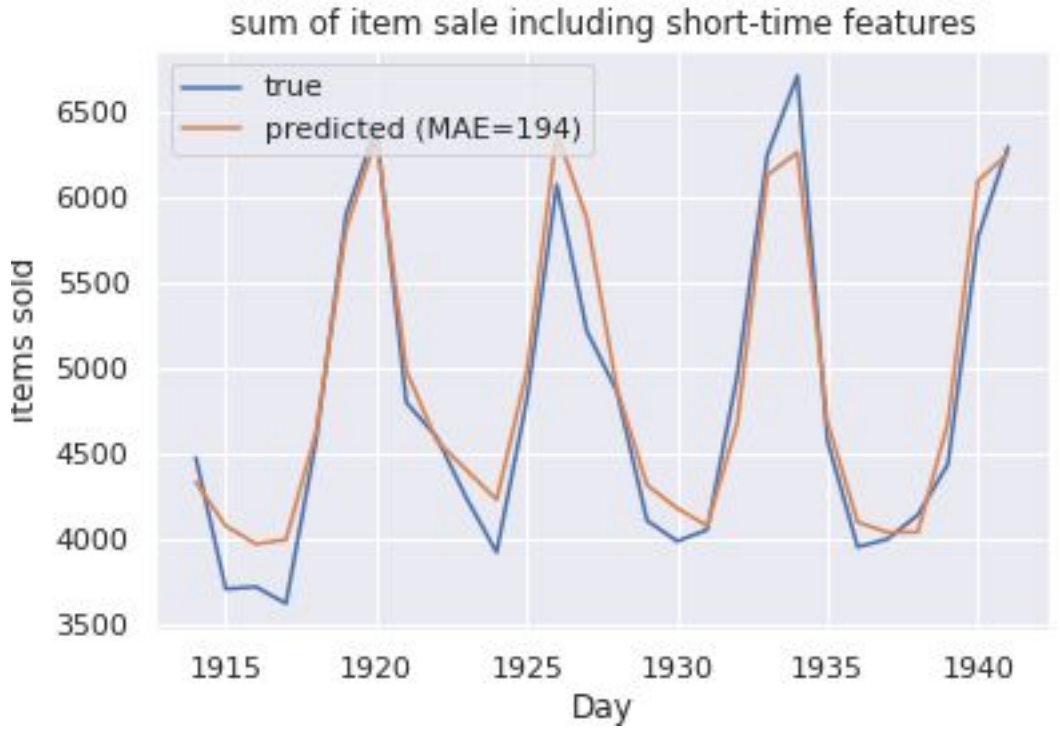
Tree Based - feature engineering (LightGBM)



Tree Based - use predictions in features

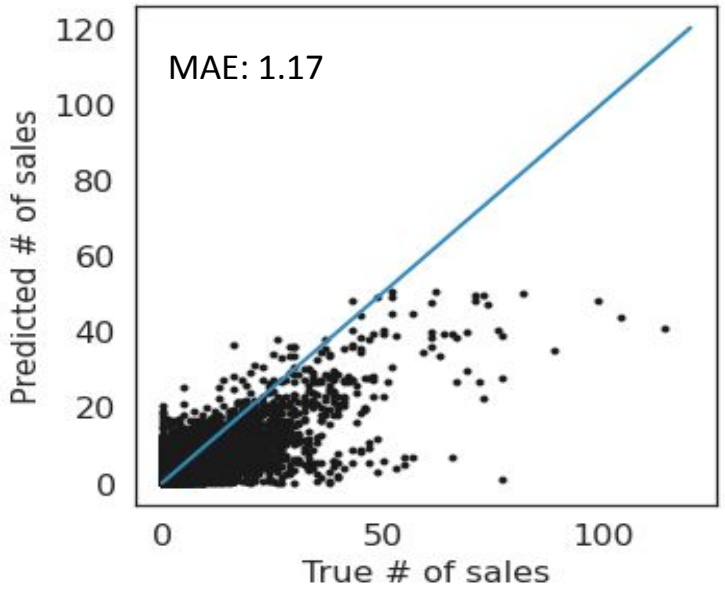
simple rolling mean of last 28 days give:
item level: MAE = 1.17
store level = MAE = 721

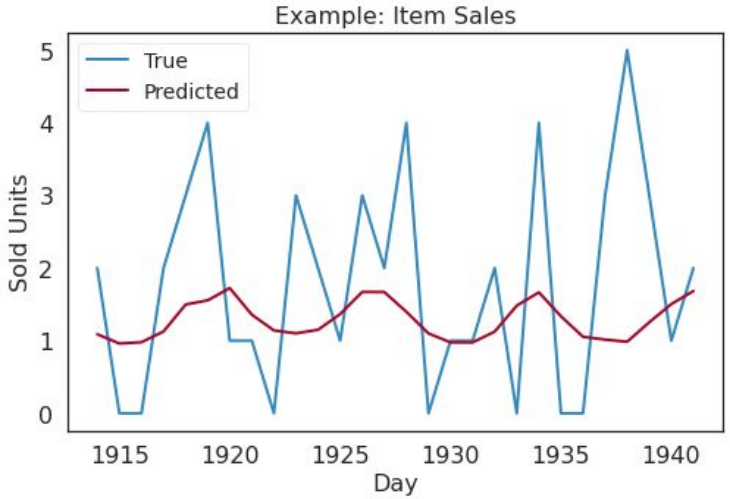


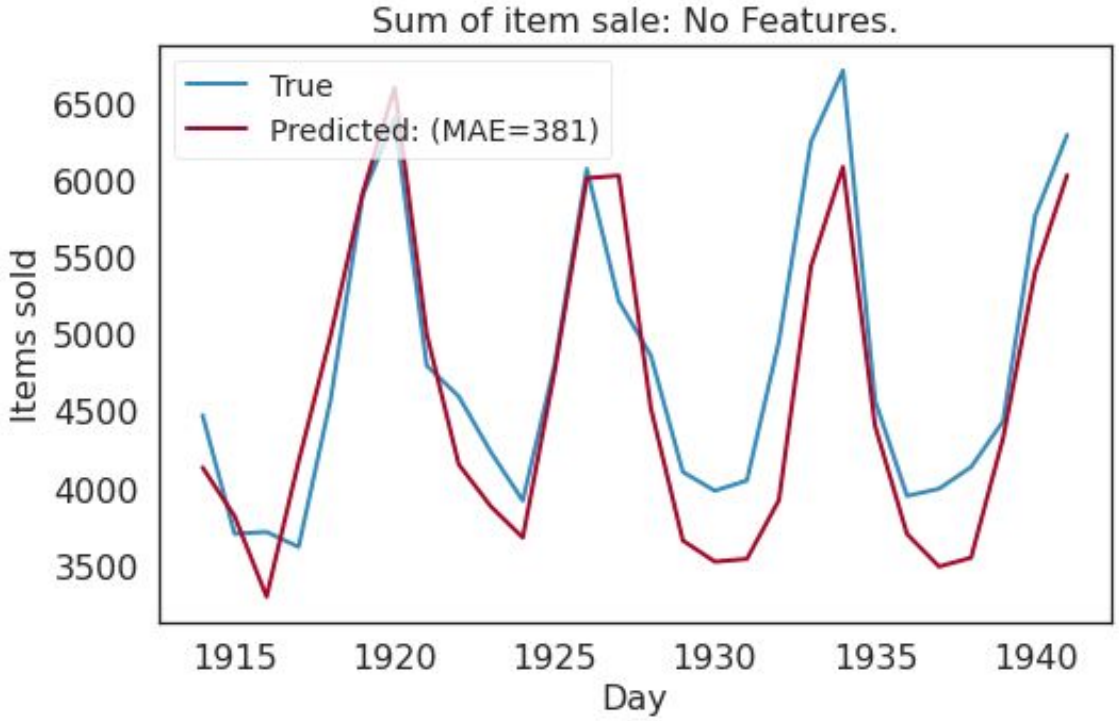


Long Short-Term Memory (LSTM)

simple rolling mean of last 28 days give:
item level: MAE = 1.17
store level = MAE = 721







64% of the sales data is 0 (sparse time series).

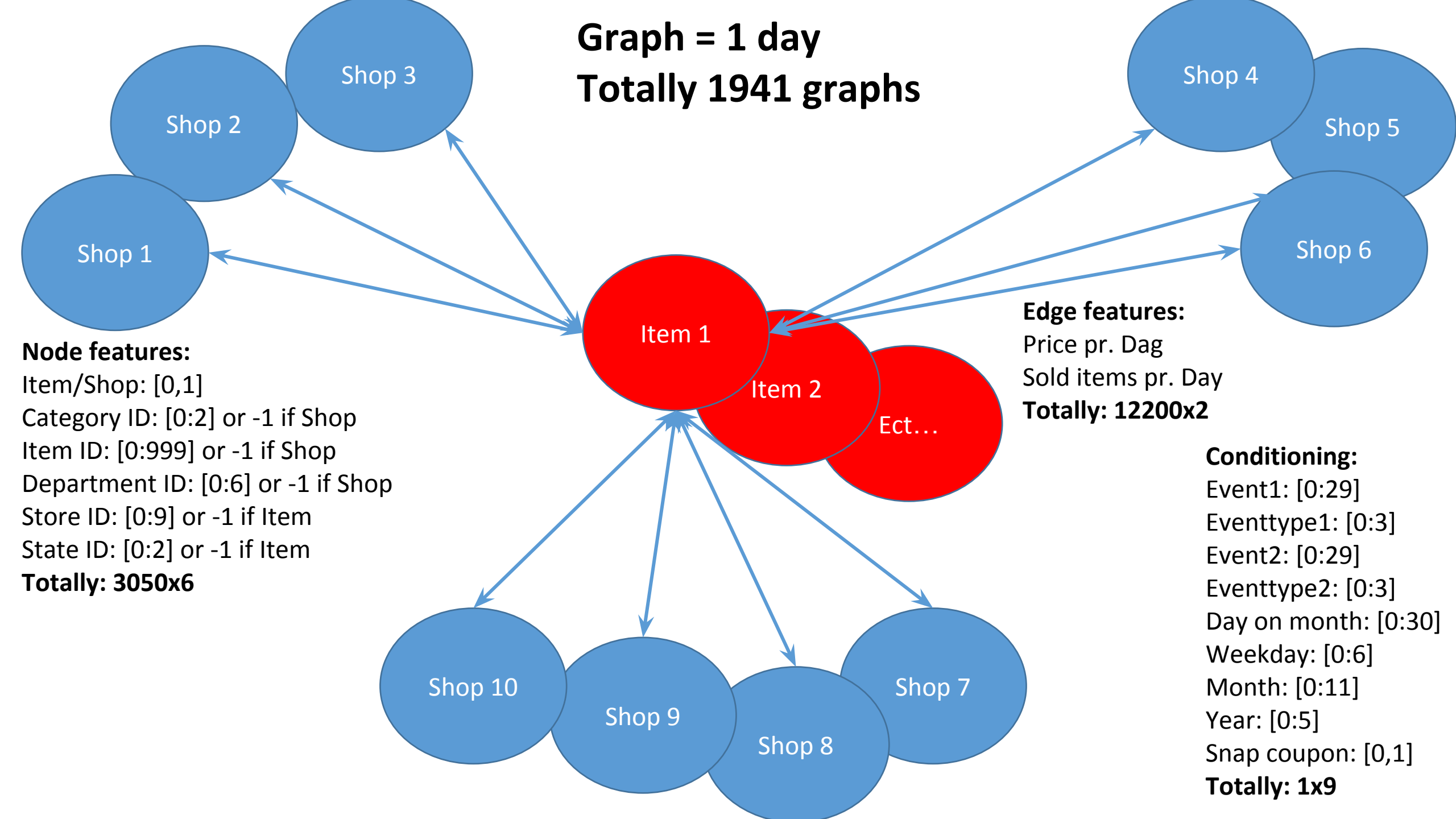
Problems and Outlook

1. Items sales seem random.

   The total sale is ok. Perhaps, unsupervised learning can cluster the time series.

2. Models are underestimating top-selling items.

   The data is unbalanced, more weight on the top-selling items.


Graph = 1 day
Totally 1941 graphs
Shop 1
Shop 2
Shop 3
Shop 4
Shop 5
Shop 6
Item 1
Item 2
Ect…
Shop 7
Shop 8
Shop 9
Shop 10
Node features:
Item/Shop: [0,1]
Category ID: [0:2] or -1 if Shop
Item ID: [0:999] or -1 if Shop
Department ID: [0:6] or -1 if Shop
Store ID: [0:9] or -1 if Item
State ID: [0:2] or -1 if Item
Totally: 3050x6
Edge features:
Price pr. Dag
Sold items pr. Day
Totally: 12200x2
Conditioning:
Event1: [0:29]
Eventtype1: [0:3]
Event2: [0:29]
Eventtype2: [0:3]
Day on month: [0:30]
Weekday: [0:6]
Month: [0:11]
Year: [0:5]
Snap coupon: [0,1]
Totally: 1x9

Graph Structure

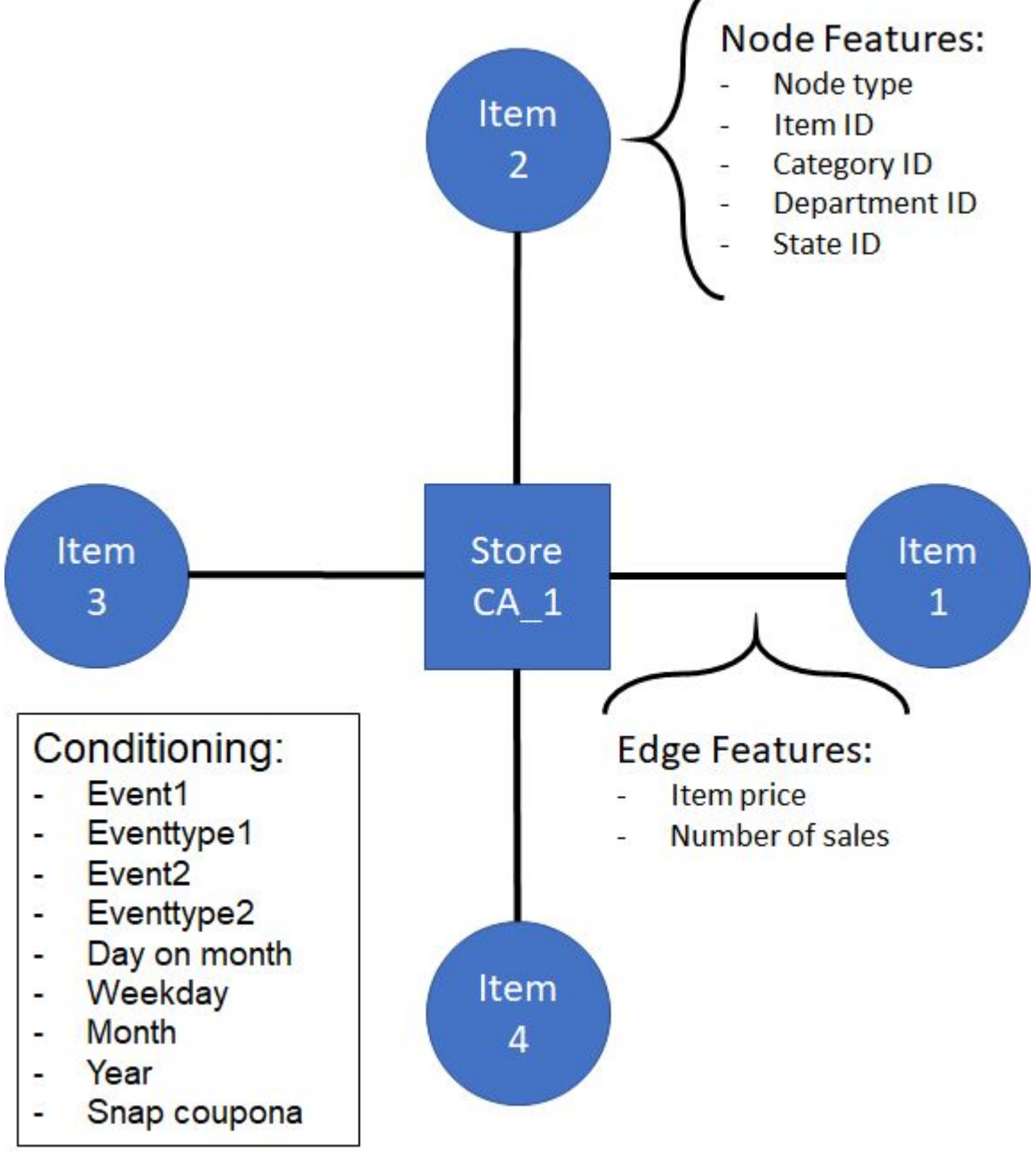
1 graph = 1 day

Only total sales prediction

Good for structuring complex data

Consists of:

- Node Feature Matrix
- Edge Feature Matrix
- Adjacency matrix



Graph Neural Network

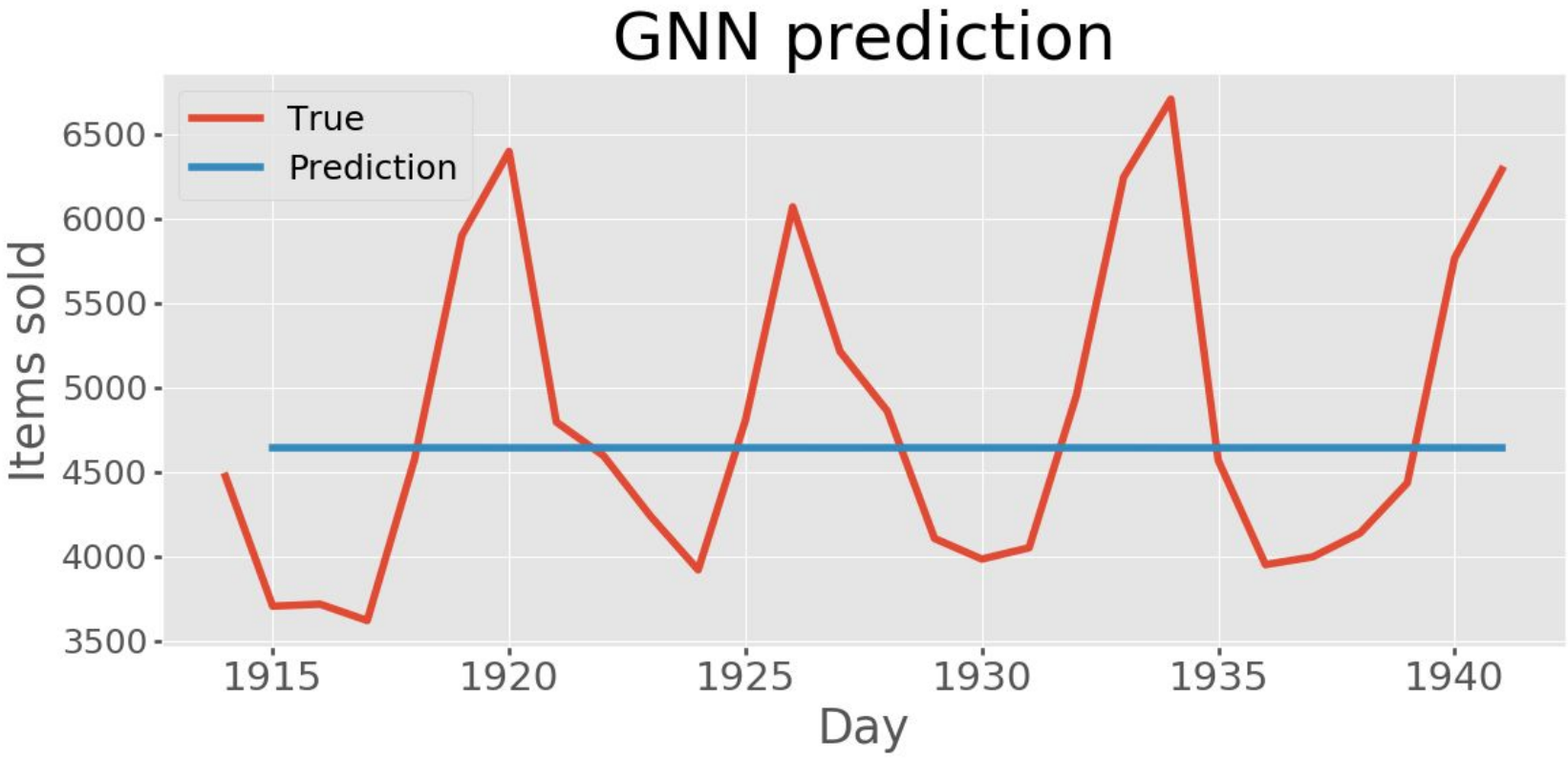
Prediction from 1 day

Last 28 day prediction

No conditioning

Only using prices and sales

MAE = 753



Conditioned GNN

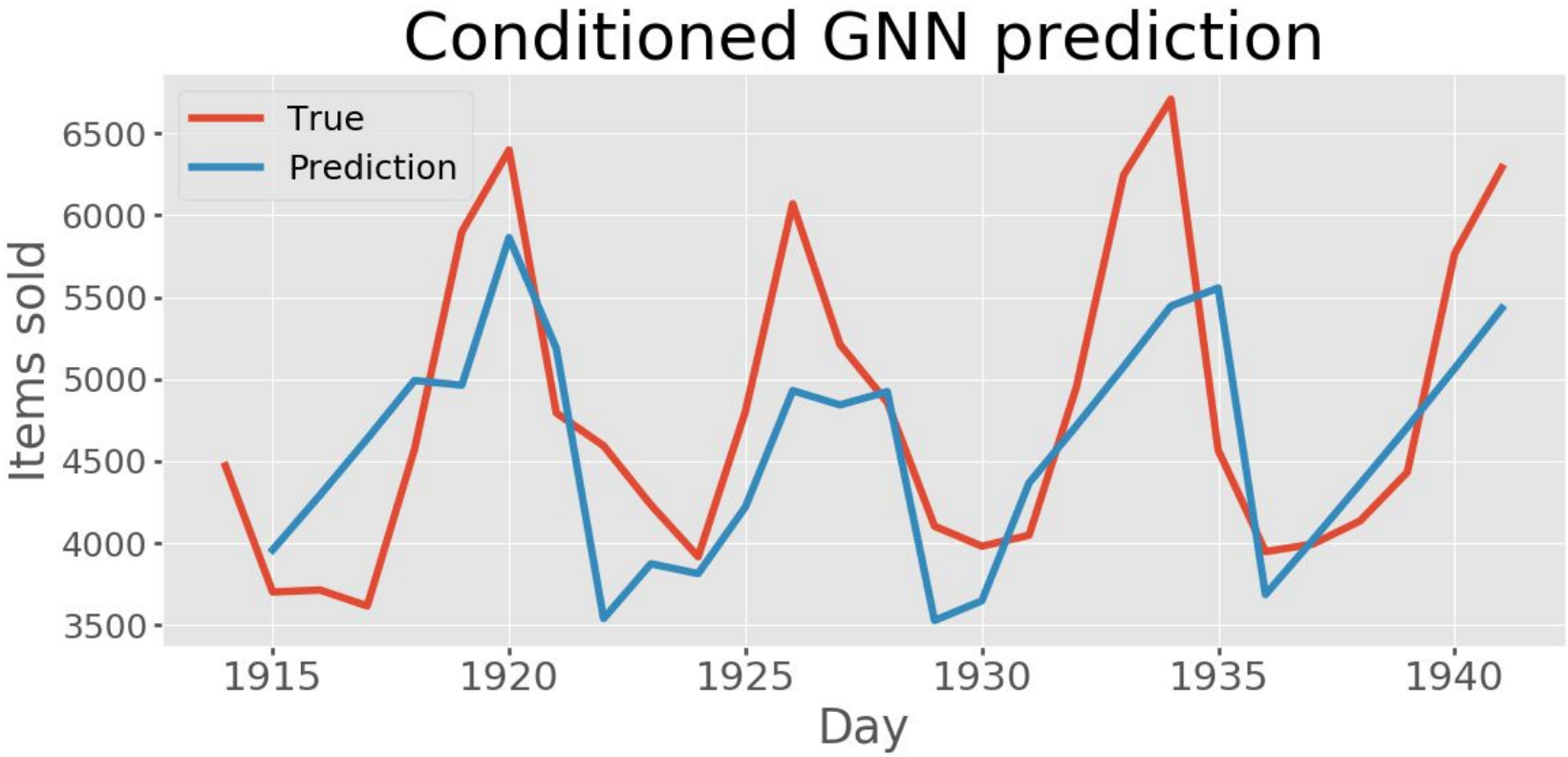
Prediction from 1 day

Last 28 day prediction

Conditioning added

Only using prices and sales

MAE = 556



Conditioned Time Series GNN

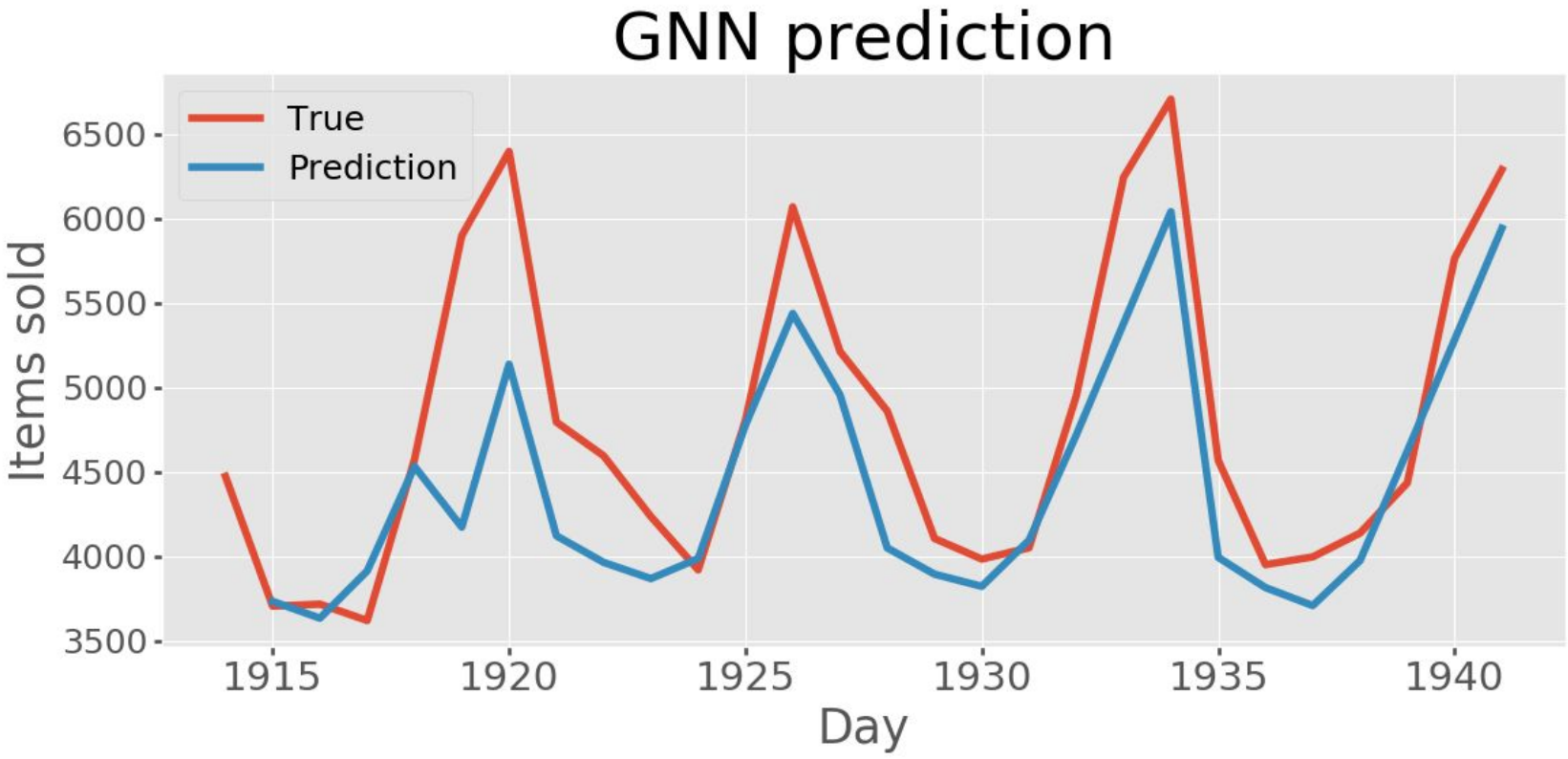
Prediction from 3 day

Last 28 day prediction

Conditioning added

Only using prices and sales

MAE = 417

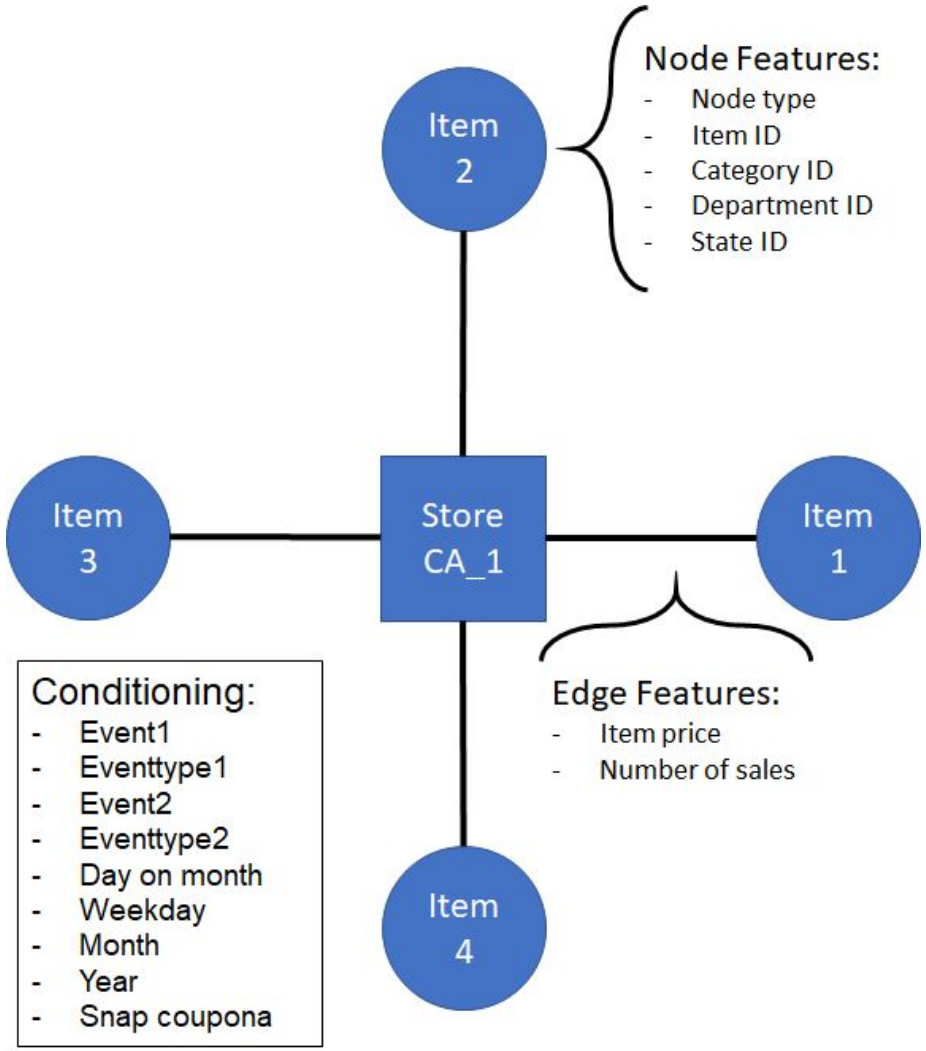


Evaluation of Graph RNN

1 day / 1 graph:

Node features matrix:
3050 x 6
Edge feature matrix:
12200 x 2
Adjacency matrix:
3050 x 3050
Conditioning:
1 x 9

Does not predict well



2 days / 2 graphs:

Node features matrix:
2*3050 x 6
Edge feature matrix:
2*12200 x 2
Adjacency matrix:
2*3050 x 2*3050
Conditioning:
1*2 x 9

28 days / 28 graphs:

Node features matrix:
85.400 x 6
Edge feature matrix:
341.600 x 2
Adjacency matrix:
85.400 x 85.400
Conditioning:
28 x 9

Does require a lot of RAM

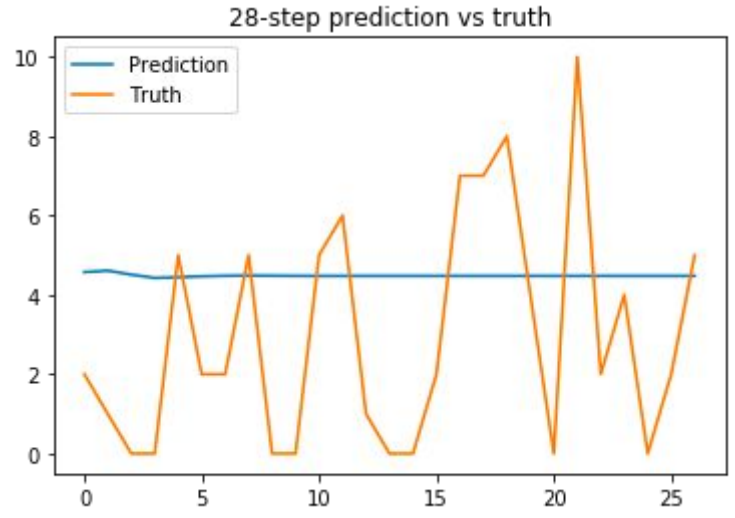
We have tried:
- Import and make data for every iteration --> Slow
- Do not use all data
- Bigger computer / GPU (Kaggle)
- Works for a sinus curve

We have not tried:
- Import data in batches
- Use pytorch's dataloader
- Dimensionality reduction
- Feature minimizing (SHAP)

Graph Based GRU

Only use small amount of data, and only use data from 1 day to predict the next.

Bad, constant prediction.



Use all the data and use data for 28 days at the time. Load data every iteration.

Very slow, especially because it has to use CPU when loading data for every iteration.

Use all the data and use data for 28 days at the time. Contain everything in the memory.

Cannot be contained in memory. Even on Kaggle (16Gb). Benefits had been large because of GPU.

RuntimeError: CUDA out of memory. Tried to allocate 570.00 MiB (GPU 0; 8.00 GiB total capacity; 6.12 GiB already allocated; 132.25 MiB free; 14.61 MiB cached)

Conclusion

We are not gonna win…

- But second place?

With great flexibility comes great preprocessing and optimization

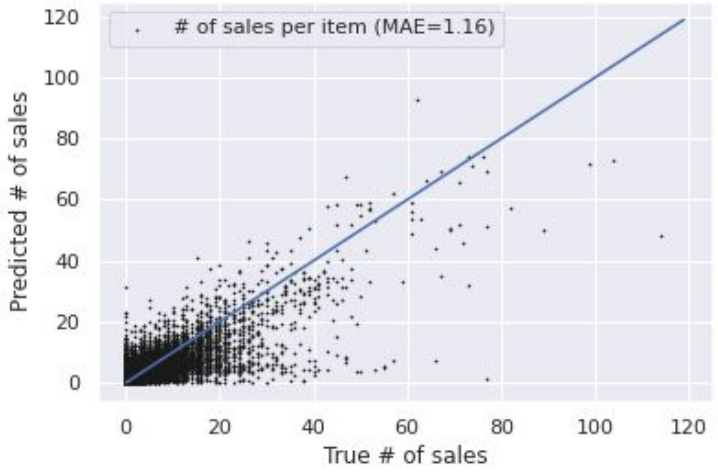
Method	Mean Absolute Error
LightGBM	255
LightGBM, Prior prediction as feature	194
LSTM	189
LSTM	381
GNN	753
CGNN	556
3-Day CGNN	417

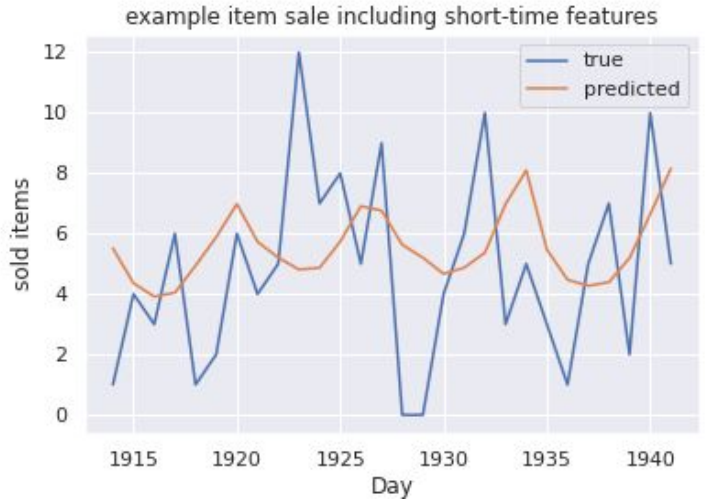
Appendix

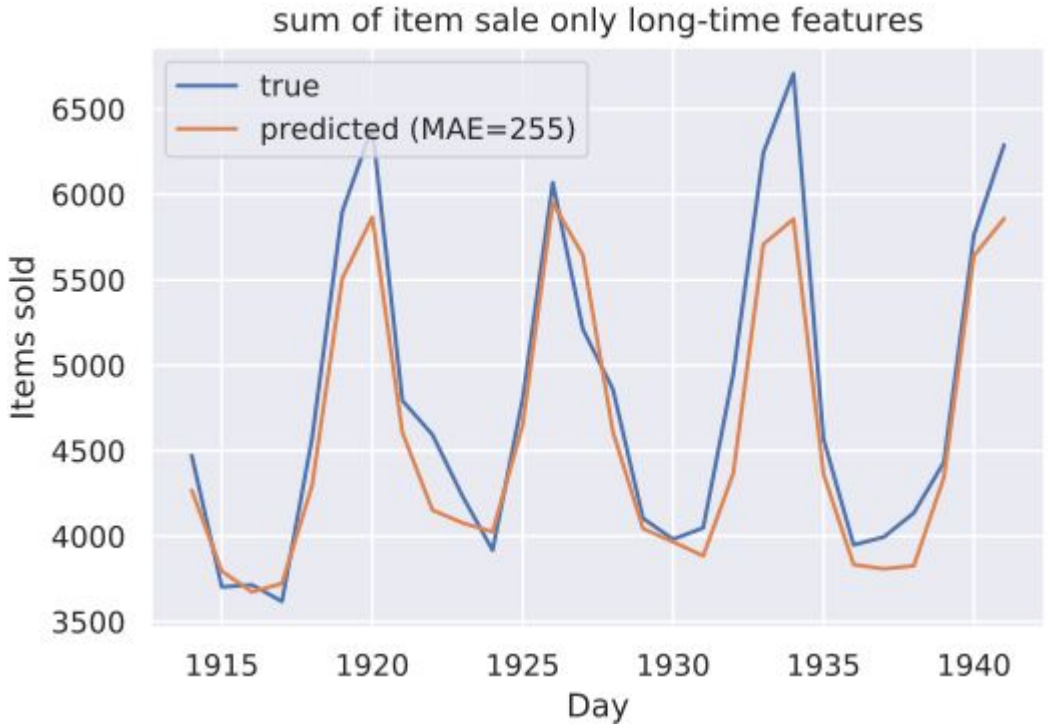
- Network architectures
  - Tree Based methods
  - Edge Graph Neural Network
  - Graph Based GRU
  - James Avery's ESN example from website adopted to Walmart Data
- ESN Results
- Table of expected results of the models

Tree Based - only known features

simple rolling mean of last 28 days give:
item level: MAE = 1.17
store level = MAE = 721







LightGBM models - technical details

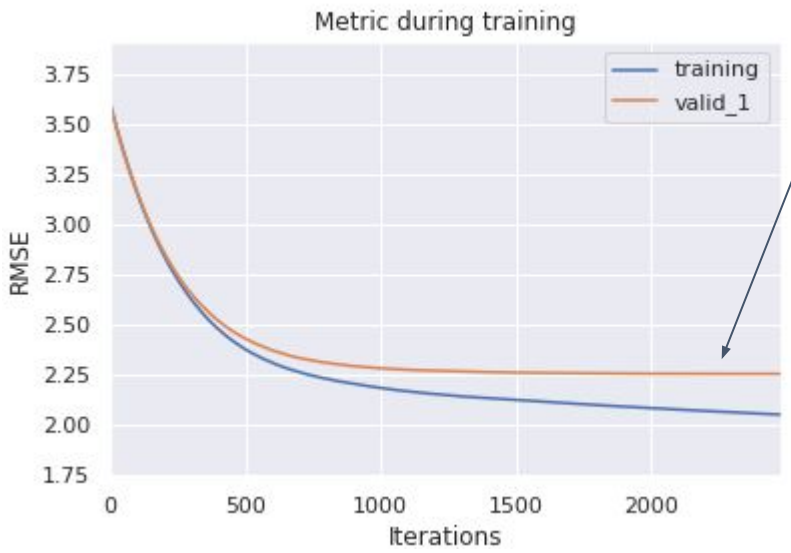
- Drop the first 4 years of data (~1.5 mio. data points left)

- Divide data set into
  - training (all days except last 56)
  - validation (next 28 days)
  - test (last 28 days)

- For HP optimization:
  - 20 combinations using random search
  - validation set used to evaluate

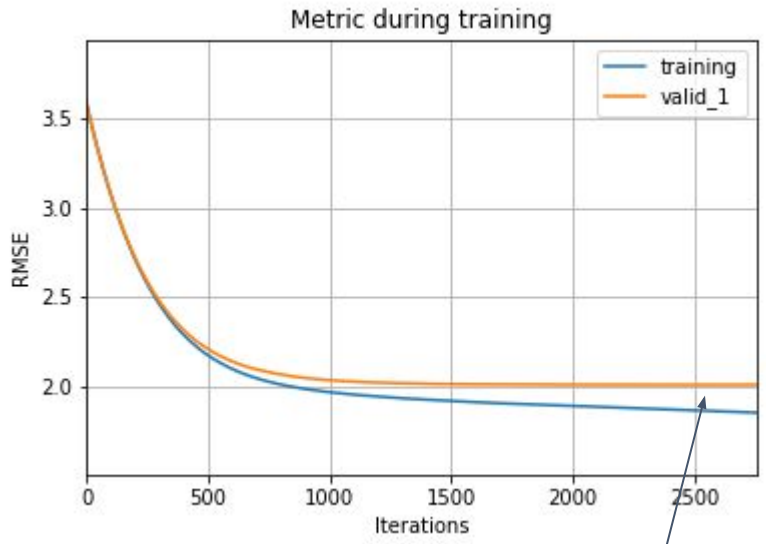
LightGBM learning curves

Only long-time features



validation RMSE starts to increase after 2280 iterations

Including short-time features



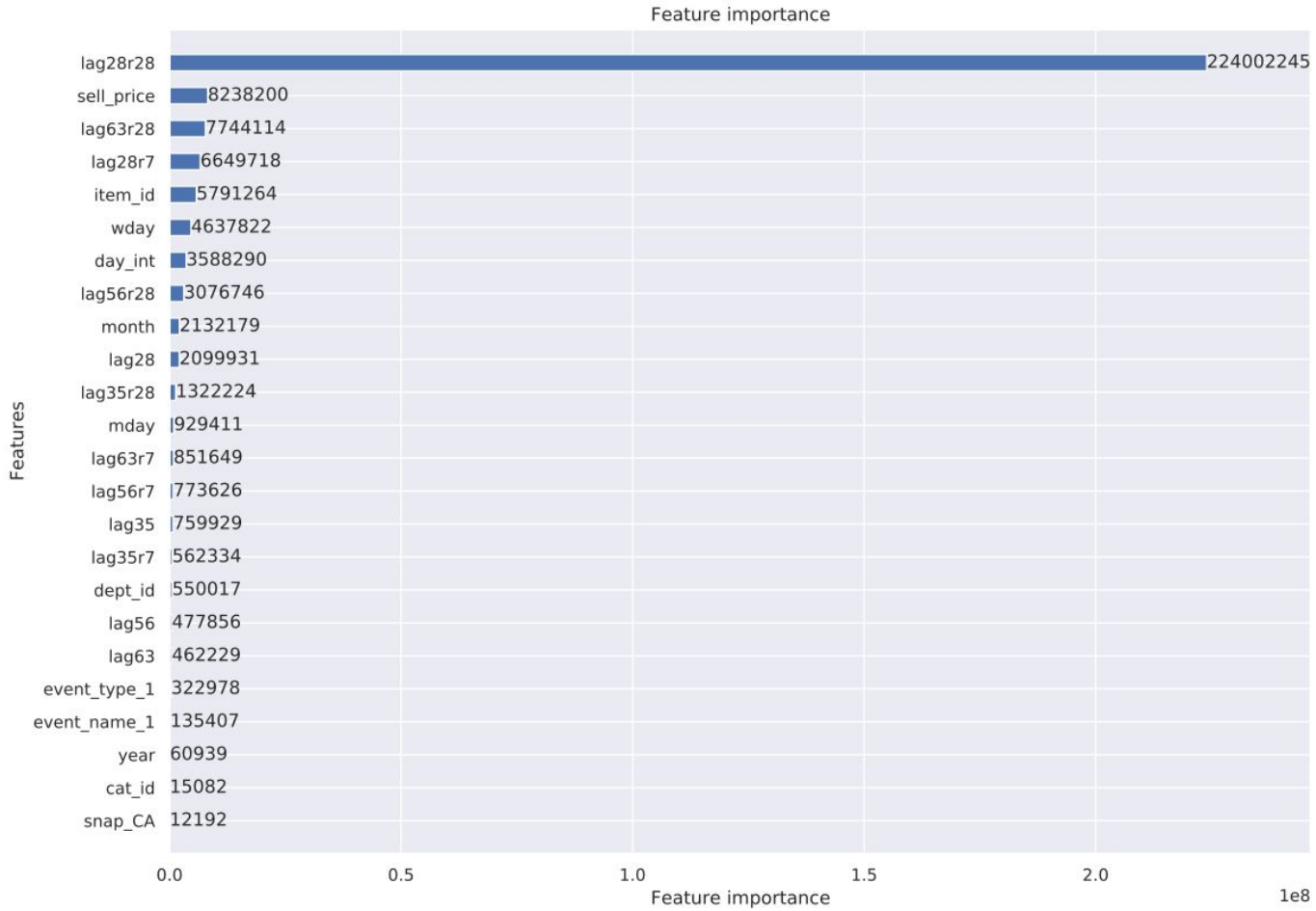
validation RMSE starts to increase after 2550 iterations

LightGBM - feature importances

Historical sales features at least 28 days back

- Rolling mean over 28 days shifted 28 days is by far the most dominant feature

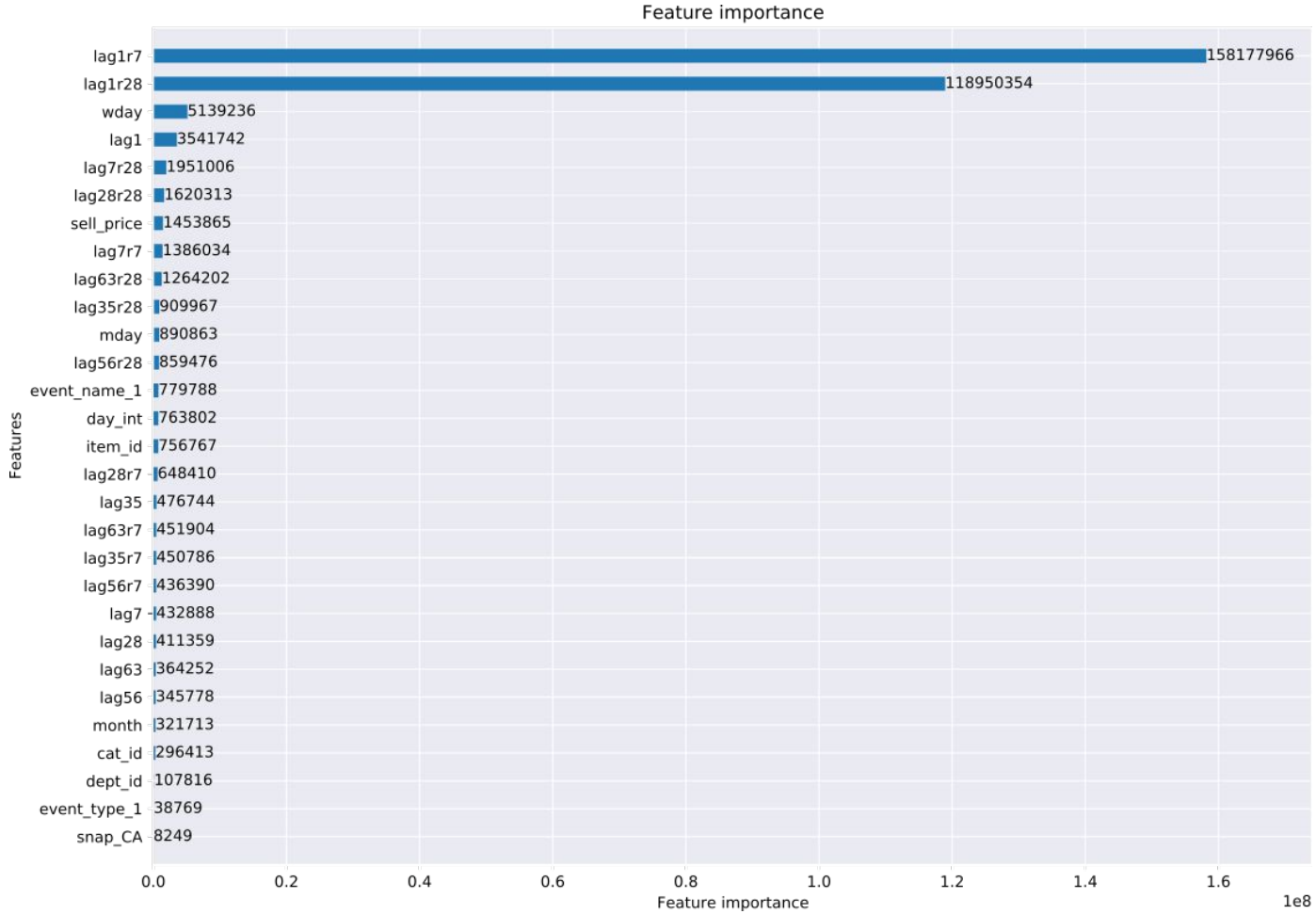
- features such as item price (sell_price), item id (item_id) and day in the week (wday) used to tune the average guess



LightGBM - feature importances’

Add historical sales data 1 and 7 days ago

- Rolling means are still by far the most dominant features though with the introduction of less shifted features means that these become more important. Here for rolling means shifted (lag) 1 day (lag1r7 and lag1r28).



LSTM - Technical Details

Dividing the data set:

- Training (all days except last 56)
- Validation (next 28 days)
- Test (last 28 days)

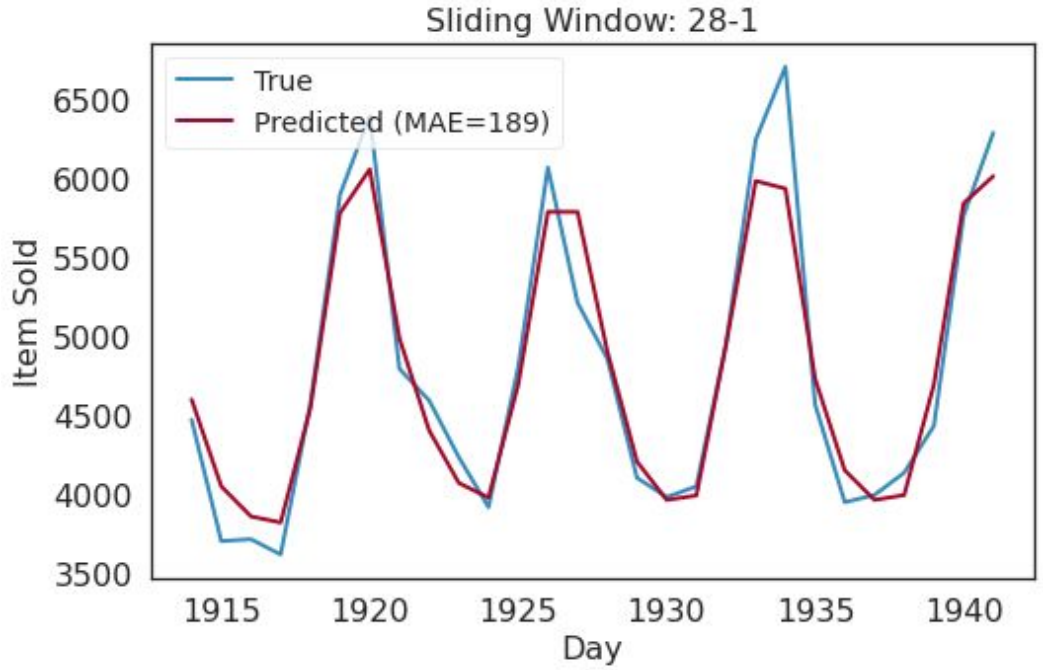
Hyperparameter opt:

- The hyperparameter space of LSTM neural networks is huge, and they are generally to train. Thus, only a few architectures were manually tested.

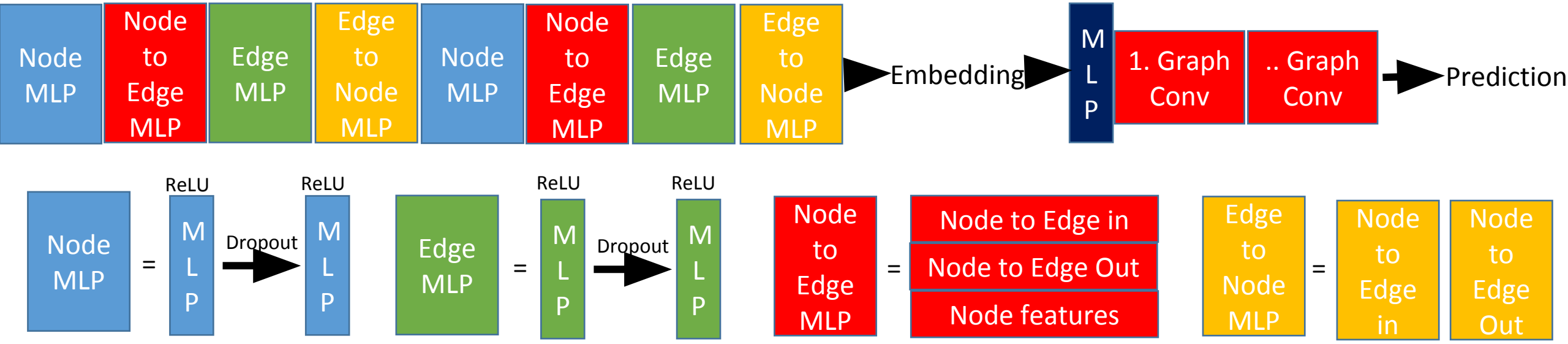
- We also tested different approaches to ingest time-series into the LSTM network.
  This includes training on the last 720, 360, 180, 56, 28 days and predicting on all 28 days at once.
  We also tested a sliding window protocol, using a sliding window of length 28 and 56 to predict only one day in advance.

LSTM - Learning Total Sales

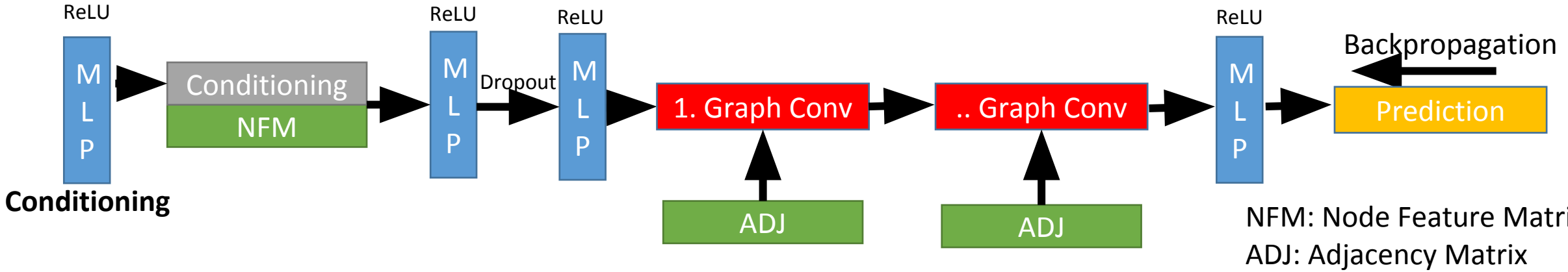
LSTM networks were quite good for directly predicting the total sales of the store with a MAE of 189 Items.



Edge Graph Neural Network



Graph Recurrent Neural Network

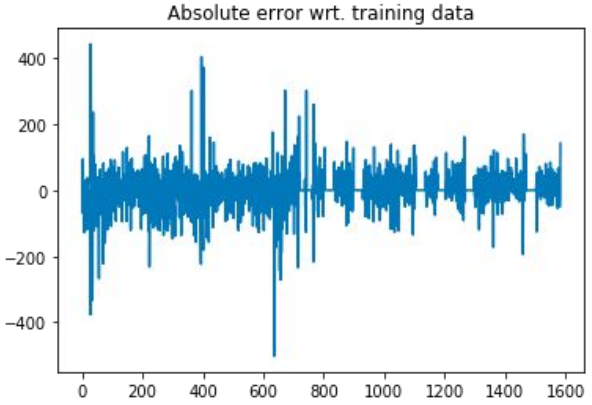


Hidden Variables: 1500
Connections in. Sparse Matrix: 100
Spectral Radius : 1.5

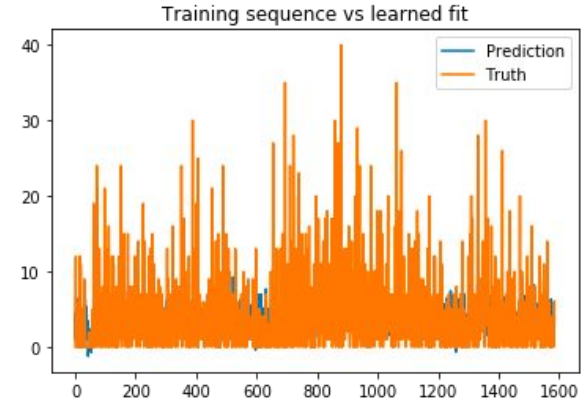
Echo State Networks

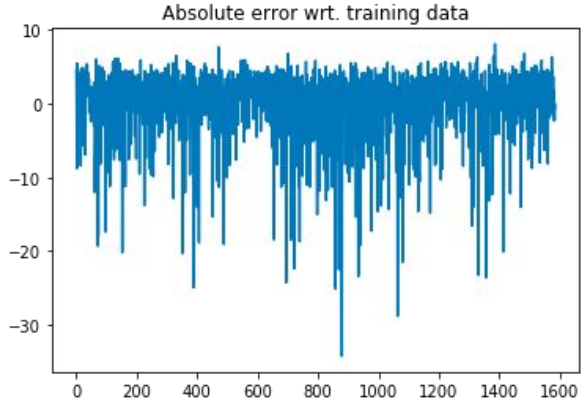
Predicts well on training data

Example1



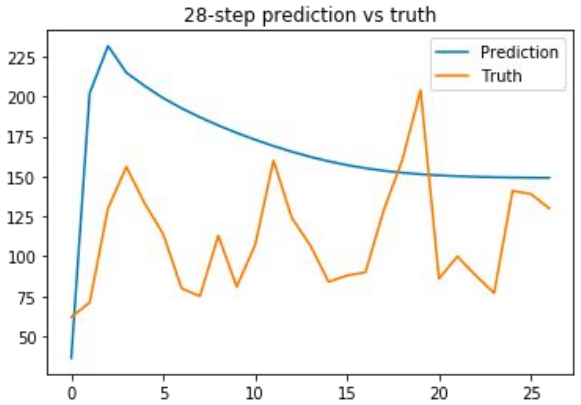
Example2

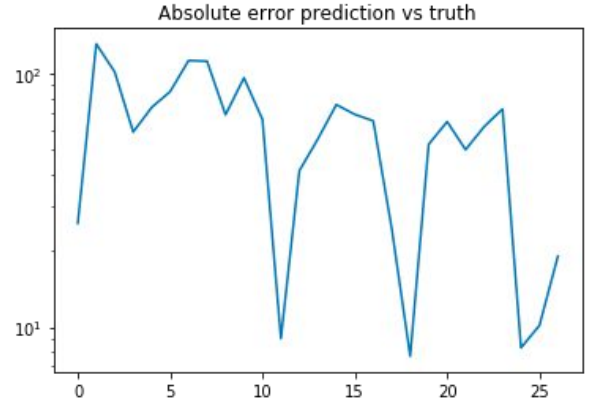




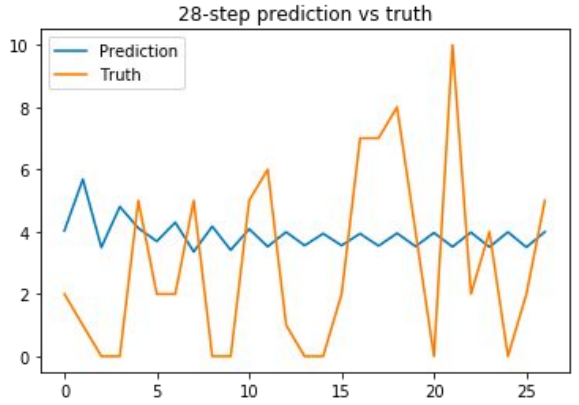
Does not generalize to validation data

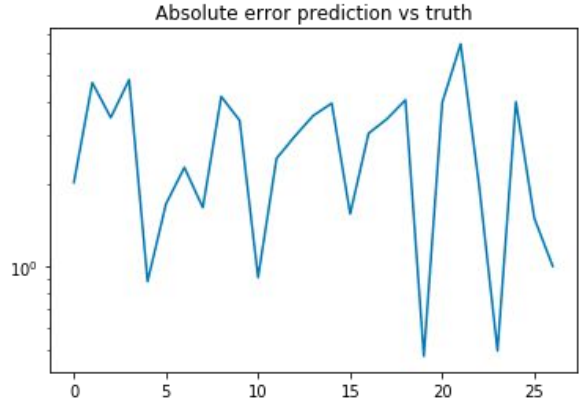
Example1





Example2





Tested ML-Algorithms

	Tree-based	LSTM	ESN	Graph-based edgeGNN	Graph-based GRU
Implementation	Easy	Difficult	Easy	Very difficult	Very difficult
Performance for few days	Good	Very good	Good	Medium	Very good
Performance for many days	Medium	Good	Good	Medium	Very good
Memory requirements for few days	Medium	Large	Medium	Small	Small
Memory requirements for many days	Large (grows linearly)	Very Large (grows linearly)	Large	Incredible Large (grows > power2)	Incredible Large (grows > power2)
Training Time	Fast	Slow	Fast	Medium	Slow