DengAI: Predicting Disease Spread
Inductive Methods of Data Analysis

Kemal Erdem

June 2, 2020

Abstract

DengaAI [2] is a competition in which the goal is to predict a number of denga cases in a given time period. Each period contains information about weather from different sources. Each participant is provided with a training dataset and test dataset (without target values). The score is decided by calculating **MAE (Mean Absolute Error)** between model predictions and true results after submission. This report contains all models used in achieving score of **12.6779 MAE** and **14/9069 place** in the competition.

Contents

1 Problem description

Figure 1: *Aedes aegypti mosquito* - Copyright Sanofi Pasteur, *CC BY-NC-ND 2.0*

Dengue fever is a mosquito-borne disease that occurs in tropical and sub-tropical parts of the world. In mild cases, symptoms are similar to the flu: fever, rash, and muscle and joint pain. In severe cases, dengue fever can cause severe bleeding, low blood pressure, and even death.

Because it is carried by mosquitoes, the transmission dynamics of dengue are related to climate variables such as temperature and precipitation. Although the relationship to climate is complex, a growing number of scientists argue that climate change is likely to produce distributional shifts that will have significant public health implications worldwide.

Goal of the competition is to predict total number of dengue fever cases reported each week in San Juan, Puerto Rico and Iquitos, Peru. We're provided with environmental data collected by various U.S. Federal Government agencies—from the *Centers for Disease Control and Prevention* [1] to the *National Oceanic and Atmospheric Administration* [7] in the *U.S. Department of Commerce* [10]

2 Dataset

The goal is to predict the **total_cases** label for each **(city, year, weekofyear)** in the test set. There are two cities, San Juan and Iquitos, with test data for each city spanning 5 and 3 years respectively. Each submission contains predictions for both cities. The data for each city have been concatenated along with a **city** column indicating the source: **sj** for San Juan and **iq** for Iquitos. The test set is a pure future hold-out, meaning the test data are sequential and non-overlapping with any of the training data. Throughout, missing values have been filled as **NaN**s.

2.1 Features

Competition provided the following set of information on a **(year, weekofyear)** timescale:

City and date indicators:

- **city** - City abbreviations: **sj** for San Juan and **iq** for Iquitos
- **week_start_date** - Date given in yyyy-mm-dd format

NOAA's GHCN daily climate data weather station measurements:

- **station_max_temp_c** - Maximum temperature
- **station_min_temp_c** - Minimum temperature
- **station_avg_temp_c** - Average temperature
- **station_precip_mm** - Total precipitation
- **station_diur_temp_rng_c** - Diurnal temperature range

PERSIANN satellite precipitation measurements (0.25x0.25 degree scale):

- **precipitation_amt_mm** - Total precipitation

NOAA's NCEP Climate Forecast System Reanalysis measurements (0.5x0.5 degree scale):

- **reanalysis_sat_precip_amt_mm** - Total precipitation
- **reanalysis_dew_point_temp_k** - Mean dew point temperature
- **reanalysis_air_temp_k** - Mean air temperature
- **reanalysis_relative_humidity_percent** - Mean relative humidity
- **reanalysis_specific_humidity_g_per_kg** - Mean specific humidity
- **reanalysis_precip_amt_kg_per_m2** - Total precipitation
- **reanalysis_max_air_temp_k** - Maximum air temperature
- **reanalysis_min_air_temp_k** - Minimum air temperature

- **reanalysis_avg_temp_k** - Average air temperature
- **reanalysis_tdtr_k** - Diurnal temperature range

Satellite vegetation - Normalized difference vegetation index (NDVI) - NOAA's CDR Normalized Difference Vegetation Index (0.5x0.5 degree scale) measurements:

- **ndvi_se** - Pixel southeast of city centroid
- **ndvi_sw** - Pixel southwest of city centroid
- **ndvi_ne** - Pixel northeast of city centroid
- **ndvi_nw** - Pixel northwest of city centroid

2.2 Data input and submission format

For example, a single row in the dataset, indexed by (city, year, weekofyear): (sj, 1994, 18), has these values:

Input data sample:

```
week_start_date 1994-05-07
total_cases 22
station_max_temp_c 33.3
station_avg_temp_c 27.7571428571
station_precip_mm 10.5
station_min_temp_c 22.8
station_diur_temp_rng_c 7.7
precipitation_amt_mm 68.0
reanalysis_sat_precip_amt_mm 68.0
reanalysis_dew_point_temp_k 295.235714286
reanalysis_air_temp_k 298.927142857
reanalysis_relative_humidity_percent 80.3528571429
reanalysis_specific_humidity_g_per_kg 16.6214285714
reanalysis_precip_amt_kg_per_m2 14.1
reanalysis_max_air_temp_k 301.1
reanalysis_min_air_temp_k 297.0
reanalysis_avg_temp_k 299.092857143
reanalysis_tdtr_k 2.67142857143
ndvi_location_1 0.1644143
ndvi_location_2 0.0652
ndvi_location_3 0.1321429
ndvi_location_4 0.08175
```

Submission format:

```
city,year,weekofyear,total_cases
sj,1990,18,4
sj,1990,19,5
...
```

2.3 Score calculation

Performance is evaluated according to the **Mean Absolute Error (MAE)**.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

3 Data Analysis

3.1 Value description

Summary of the train dataset:

	year	weekofyear	ndvi_ne	ndvi_nw	ndvi_se	ndvi_sw	precipitation_amt_mm
count	1456.00	1456.00	1262.00	1404.00	1434.00	1434.00	1443.00
mean	2001.03	26.50	0.14	0.13	0.20	0.20	45.76
std	5.41	15.02	0.14	0.12	0.07	0.08	43.72
min	1990.00	1.00	-0.41	-0.46	-0.02	-0.06	0.00
max	2010.00	53.00	0.51	0.45	0.54	0.55	390.60

	reanalysis_air_temp_k	reanalysis_avg_temp_k	reanalysis_dew_point_temp_k
count	1446.00	1446.00	1446.00
mean	298.70	299.23	295.25
std	1.36	1.26	1.53
min	294.64	294.89	289.64
max	302.20	302.93	298.45

	reanalysis_max_air_temp_k	reanalysis_min_air_temp_k
count	1446.00	1446.00
mean	303.43	295.72
std	3.23	2.57
min	297.80	286.90
max	314.00	299.90

	reanalysis_precip_amt_kg_per_m2	reanalysis_relative_humidity_percent
count	1446.00	1446.00
mean	40.15	82.16
std	43.43	7.15
min	0.00	57.79
max	570.50	98.61

	reanalysis_sat_precip_amt_mm	reanalysis_specific_humidity_g_per_kg
count	1443.00	1446.00
mean	45.76	16.75
std	43.72	1.54
min	0.00	11.72
max	390.60	20.46

	reanalysis_tdtr_k	station_avg_temp_c	station_diur_temp_rng_c
count	1446.00	1413.00	1413.00
mean	4.90	27.19	8.06
std	3.55	1.29	2.13
min	1.36	21.40	4.53
max	16.03	30.80	15.80

	station_max_temp_c	station_min_temp_c	station_precip_mm
count	1436.00	1442.00	1434.00
mean	32.45	22.10	39.33
std	1.96	1.57	47.46
min	26.70	14.70	0.00
max	42.20	25.60	543.30

Table 1: Data Description

As we can see, some of the features are empty. E.g. **ndvi_ne** is missing 13.3% of all values. To fix that issue I'm using interpolation from neighboring values. Interpolation is applied to all features.

If we look on values we can easily distinguish two cities in dataset

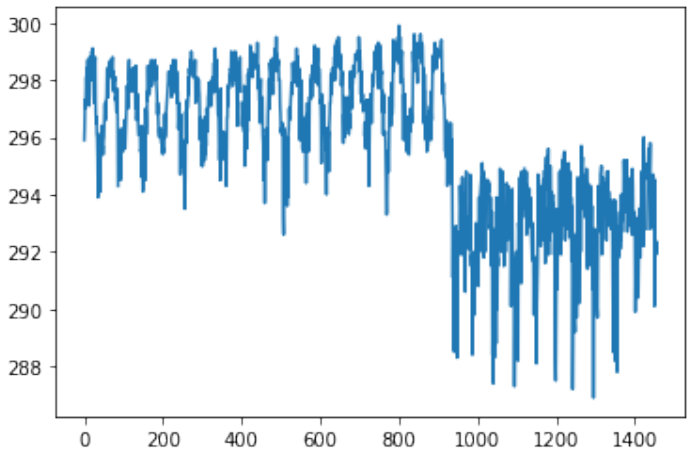

Figure 2: reanalysis_min_air_temp_k

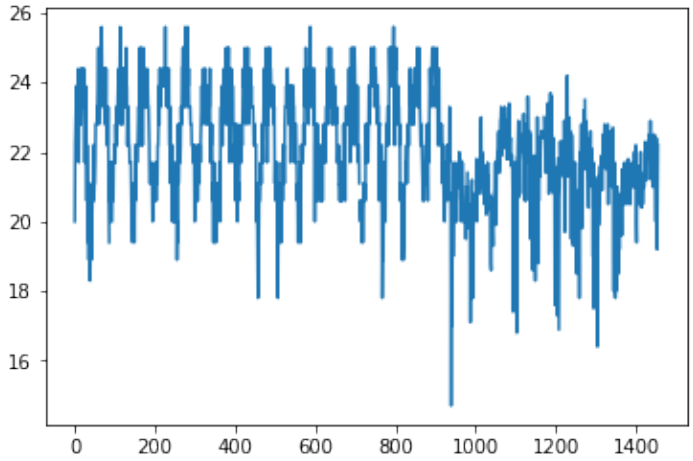

Figure 3: station_precip_mm

3.2 Pair correlation

To reduce the number of features we can use pairplots Fig. 4, Fig. 5.

First we look at the pairplot for *precipitation* values. We can spot that *reanalysis_sat_precip_amt_mm* and *precipitation_amt_mm* are perfectly correlated so we can just drop one feature (Fig. 4).

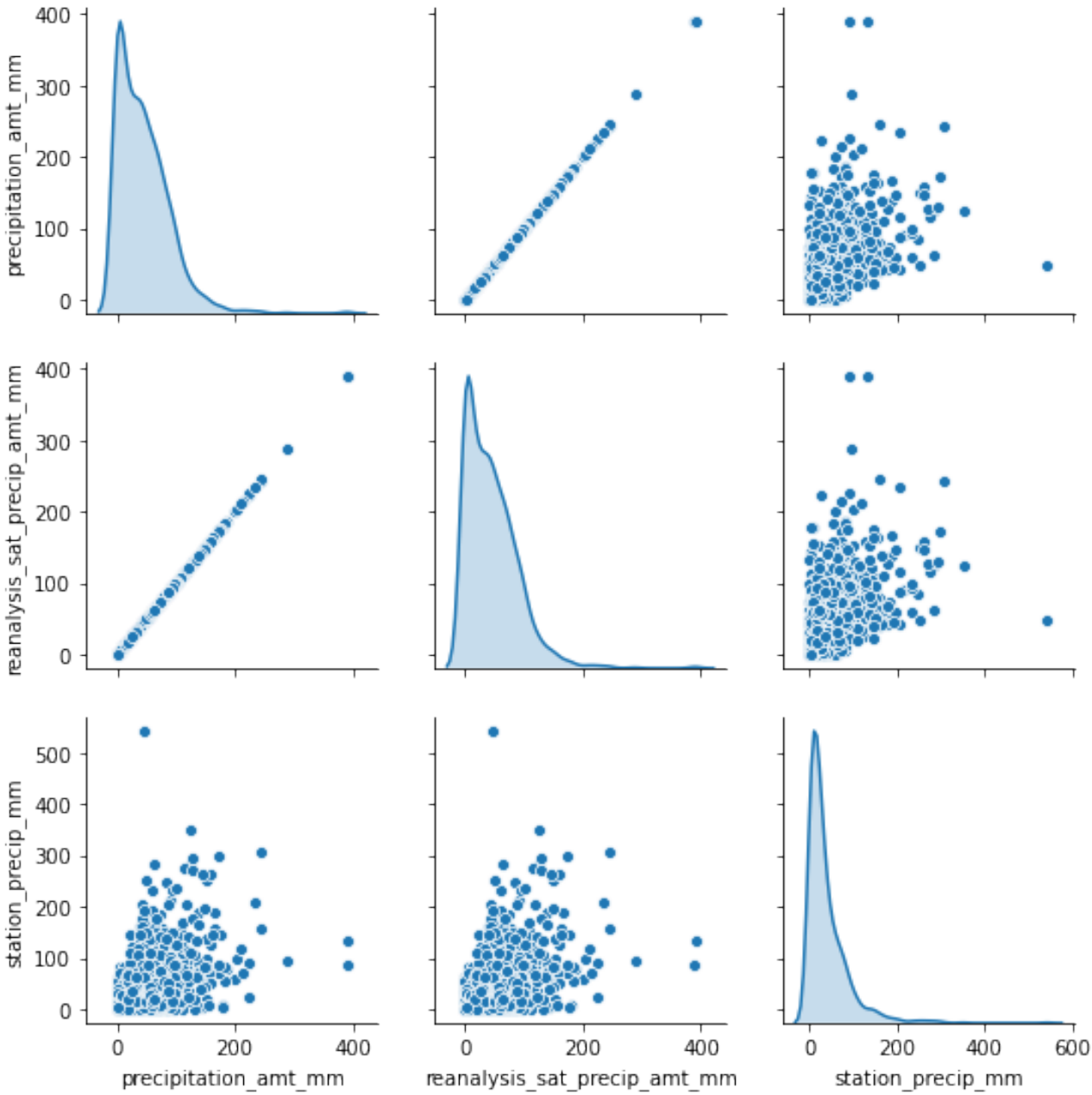

Figure 4: Percipation pairplot

On the other hand, even that temperature should be correlated with each other, in case of this dataset it isn't (Fig. 5).

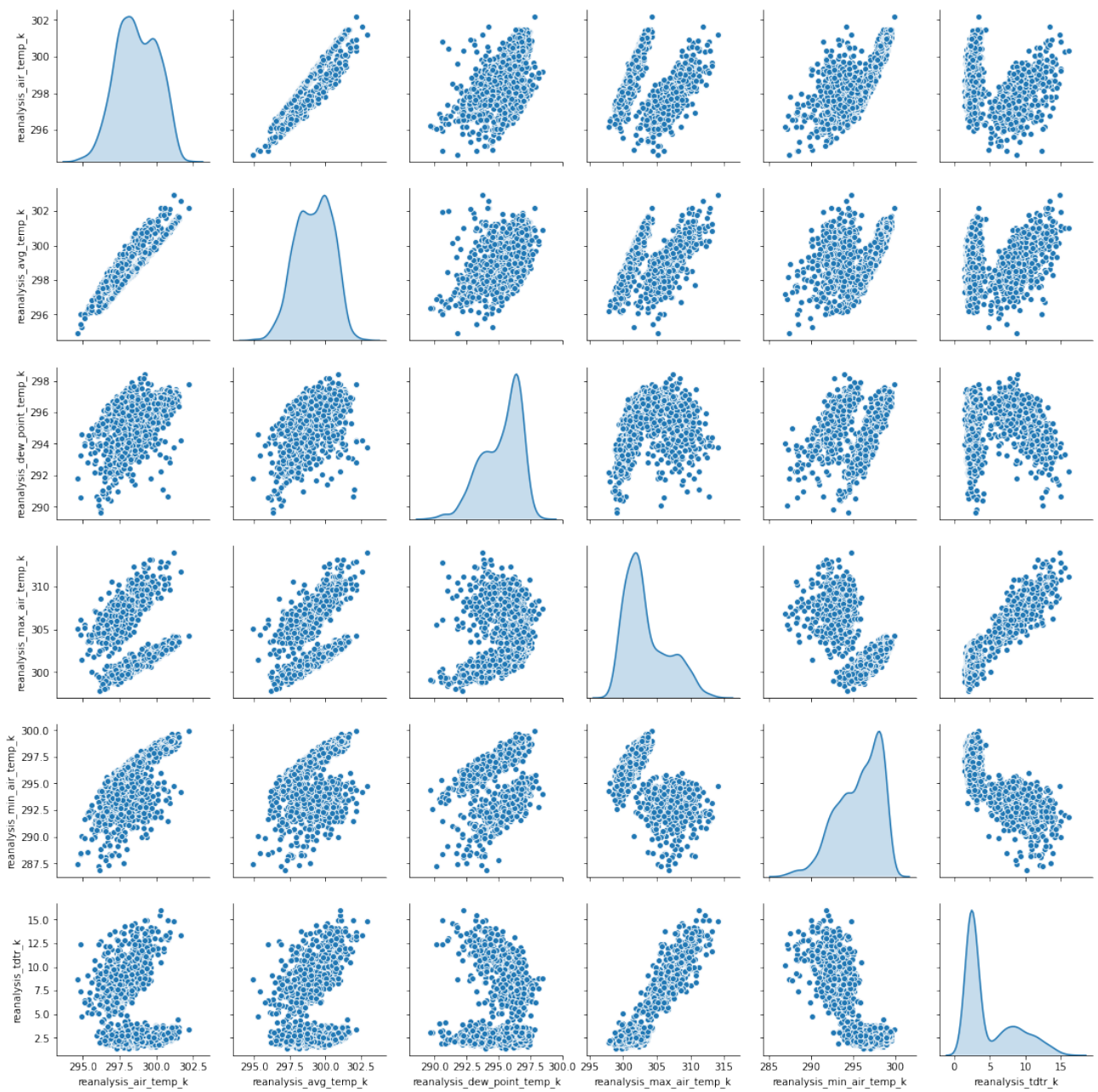

Figure 5: Temperature pairplot

3.3 Data Correltation

If we look at the standard data correlation plot we can see that some of the features are strongly positively correlated with each other. Correlation isn't perfect but it's still useful to simplify the model. E.g. *reanalysis_specific_humidity_g_per_kg* is strongly correlated with temperature and that makes sense (Fig. 6). Unfortunately none of the features is correlated strongly with *total_cases* which is our target. Best correlation we have is *weekofyear* and *year* (Fig. 7). Both correlations are between 0.2 and 0.3.

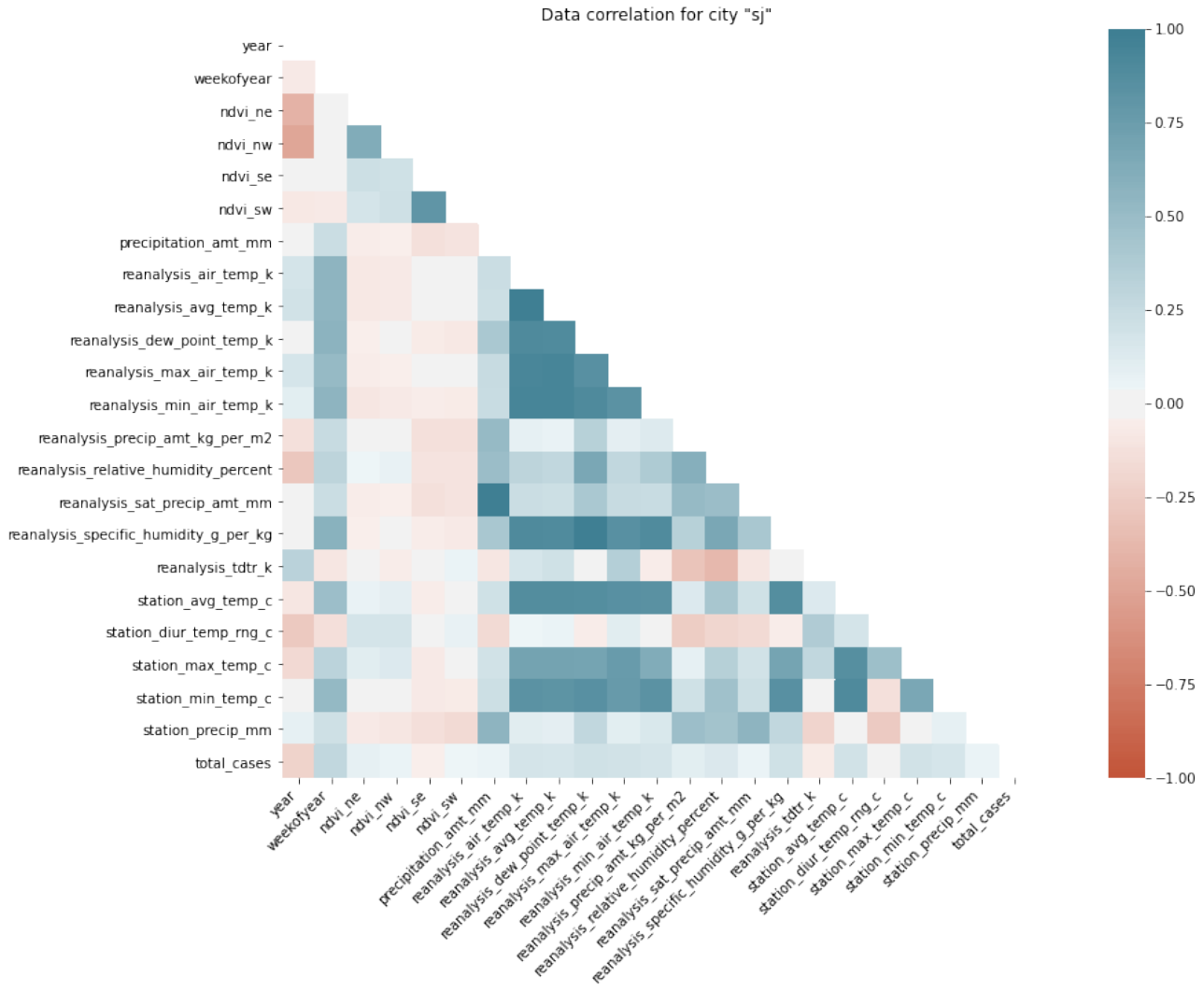

Figure 6: Feature correlation for **sj** city

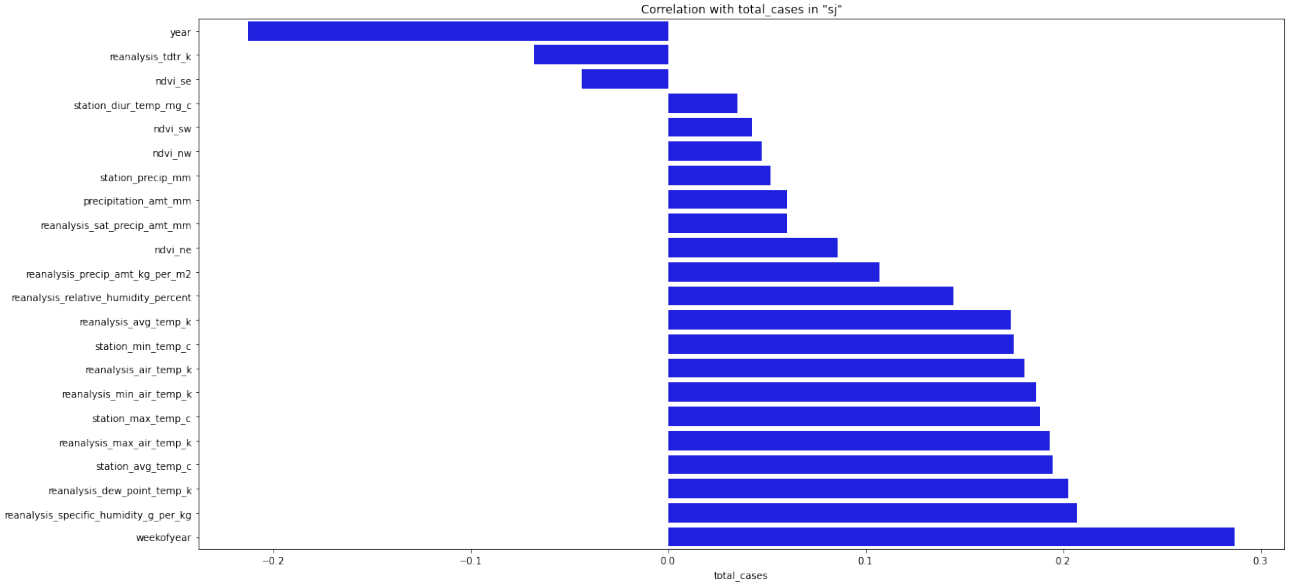

Figure 7: Feature correlation to **total_cases** for **sj** city

3.4 Dimentionality Reduction

Another try to extract information for data was to use t-SNE [5] algorithm. t-SNE manage to correctly make two distinct clusters from cities but that didn't help with assigning any meaningful information to the data (Fig. 8). The second used algorithm was PCA [4]. PCA wasn't able to distinguish data at all, the reason for that might be because data is not linearly separable (Fig. 9). Note that each dot shade is responsible for the total number of cases.

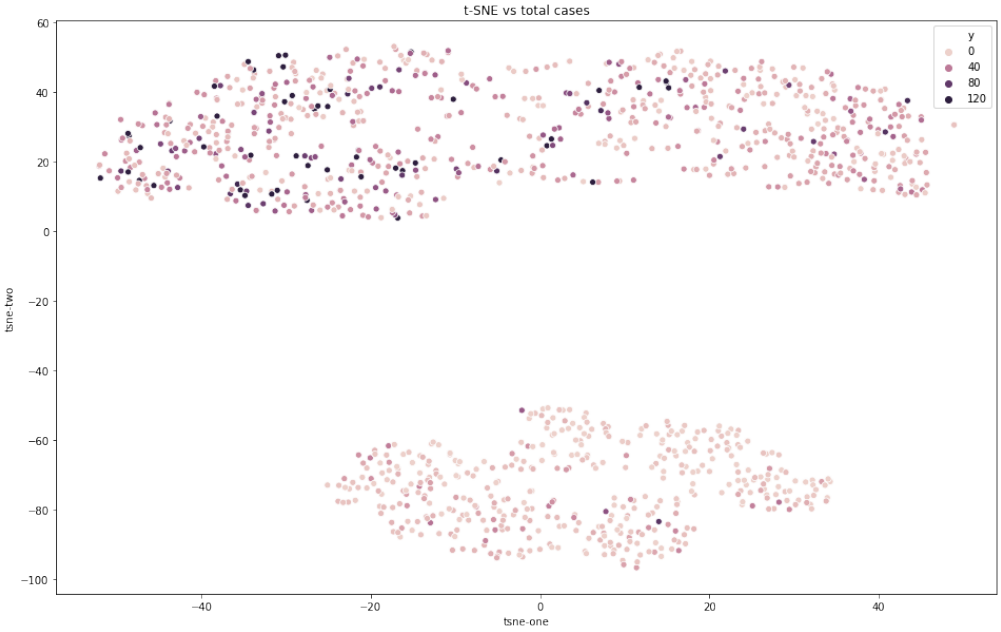

Figure 8: t-SNE visualization

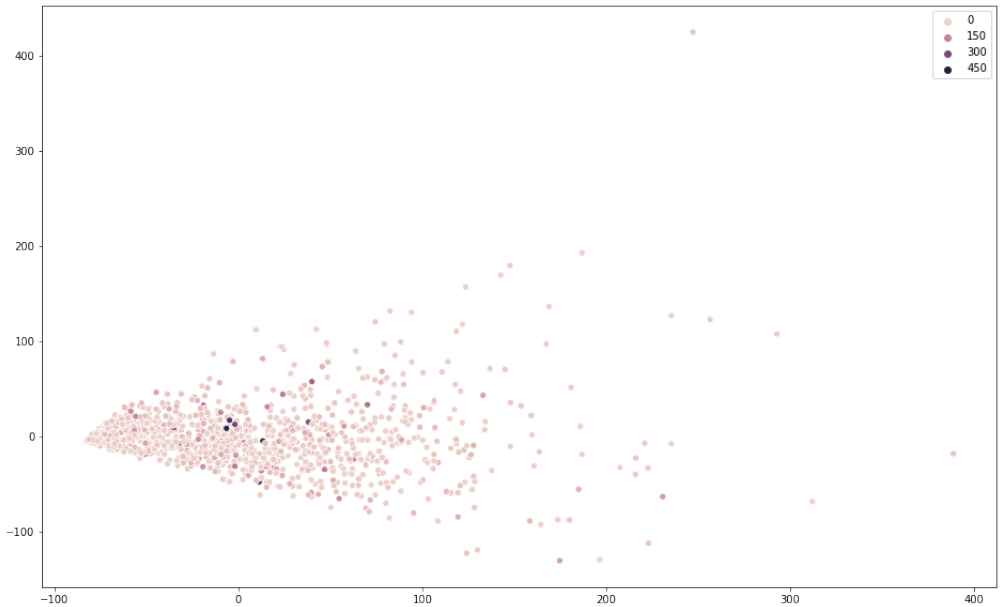

Figure 9: PCA visualization

3.5 Additional Data Analisis

Number of total cases in an average week is quite low. Only from time to time (usually once a year) total cases increases. Best way to see that is to check histogram of the target value (Fig. 10, Fig. 11). Most of the values (highest bar) are in the lower range, only a few are in the upper range. That means we have short "pikes" of total cases in our dataset.

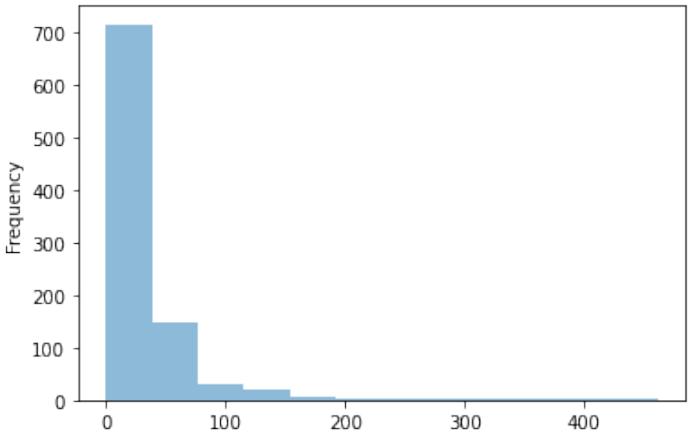

Figure 10: **sj** total cases histogram

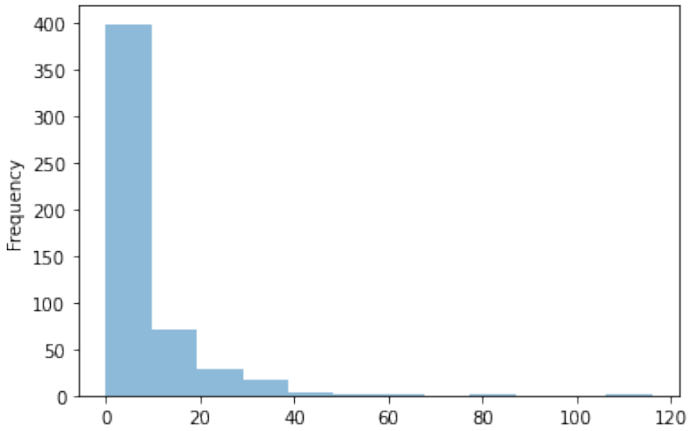

Figure 11: **iq** total cases histogram

3.6 NDVI values

Because NVID index values are one of the most missing values I've tried to find out how much additional error is added to dataset when interpolating data. To do that I've calculated **MAE** for every NDVI value and value created if NDVI was interpolated from its neighbors.

In **sj** city average **std** for NDVI values is **0.11**, interpolation of the same data creates MAE of 0.05 which is almost 1/2 of standard deviation for that feature. In **qi** city average **std** for NDVI values is **0.08**, interpolation of the same data creates MAE of 0.07 which is almost the same as standard deviation for that feature. This makes interpolation less useful for data fixing in the case of NDVI.

4 Preprocessing

4.1 Default data preprocessing

Before feeding data to the model we have to do some preprocessing. First thing to do is to separate features which have to be normalized, scaled and copied.

Normalize:

```
'reanalysis_air_temp_k',
'reanalysis_avg_temp_k',
'reanalysis_dew_point_temp_k',
'reanalysis_max_air_temp_k',
'reanalysis_min_air_temp_k',
'reanalysis_tdtr_k',
'station_avg_temp_c',
'station_diur_temp_rng_c',
'station_max_temp_c',
'station_min_temp_c'
```

Scale:

```
'precipitation_amt_mm',
'reanalysis_precip_amt_kg_per_m2',
'reanalysis_relative_humidity_percent',
'reanalysis_sat_precip_amt_mm',
'reanalysis_specific_humidity_g_per_kg',
'station_precip_mm',
'year',
'weekofyear'
```

Copy:

```
'ndvi_ne',
'ndvi_nw',
'ndvi_se',
'ndvi_sw',
```

We're not using normalization for *year* and *weakofyear* (and other features) because those values shouldn't have negative values (there is no negative percentage). After normalizing|scaling training data, all scales used for normalization|scaling are stored for potential test data preprocessing to achieve the same result. Copied data (NDVI index) doesn't require normalization or scaling because it's already within range of <**-1,1**>.

4.2 Time series preprocessing

Because at some point dataset is treated as timeseries instead of just simple regression problem we need to change preprocessing and stack multiple frames as one input. This is done by extracting data based on *HISTORY SIZE* attribute of the LSTM and data row index. Index of each data row is set to be **week_start_date**. That way we can feed data to LSTM models with different history size.

4.3 City data split

There was one column in dataset which described city name called *city*. Because this column contains only 2 different values (*sj* and *iq*) I've replaced it with two binary columns called **sj** and **iq** indicating to which city data belongs to.

5 Base Model

5.1 Full dataset training

First model used as a baseline was simple 3-layer NN with a structure:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================

_________________________________________________________________
input                        (None, 25)                0
_________________________________________________________________
dense (Dense)                (None, 64)                1600
_________________________________________________________________
dense_1 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 5,825
Trainable params: 5,825
Non-trainable params: 0
```

To train this model, the training dataset was split into training and validation dataset in 80/20 proportion. Default optimizer used for training was *Adam*:

- **learning rate** - Training learning rate
- β_1 - Lower β parameter
- β_2 - Upper β parameter

Model used preprocessed input with additional early stopping enabled when there were no improvements in 20 epochs. Total training epochs: **100**. (Fig. 12)

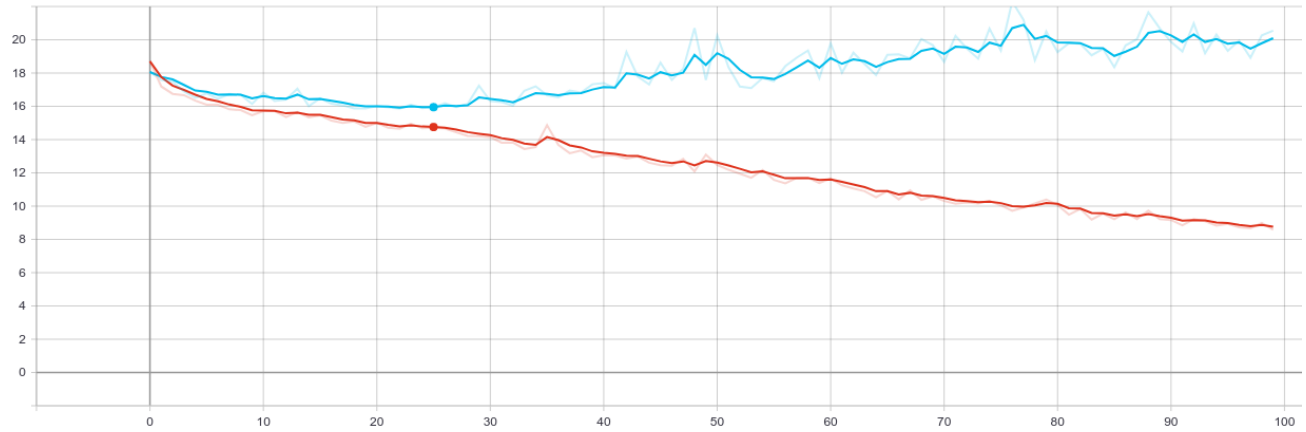

Figure 12: Point of early stoping (**red** - train MAE, **blue** - val MAE)

Result on the train dataset: **MAE: 14.64**

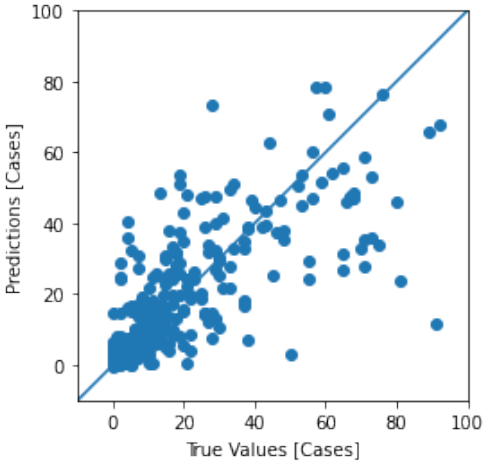

Figure 13: Predictions/True Values scatter plot

Predictions on the training dataset were't good (Fig. 14) and the result on test dataset after submission was:

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
30.1178	2713	8549	1 of 3

Figure 14: Result of the baseline model

5.2 Reduced dimentionality

The next baseline mode had a different set of input features used to train the model. This time model used only features with the highest correlation with **total_cases**.

```
'year',
'weekofyear',
'reanalysis_tdtr_k',
'station_diur_temp_rng_c',
'precipitation_amt_mm',
'sj',
'iq'
```

Models structure changed a little. Now its first hidden layer has $input^2$ neurons and the second hidden layer has 128 neurons. An additional change was to add a callback to reduce *learning rate* after every 5 epochs without improvement.

If we compare training with only selected features and baseline we can see that training loss improved a little. Point of early stopping moved to 68 epoch and MAE on val dataset decreased to 15.6 from 16.0 earlier (Fig. 15).

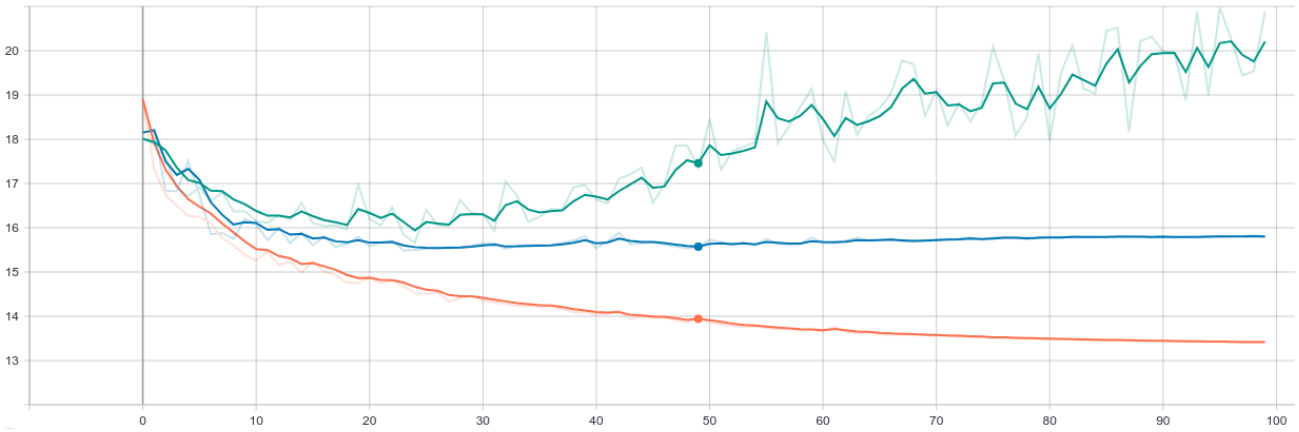

Figure 15: Point of early stoping (**orange** - train MAE, **blue** - val MAE, **green** - base MAE)

Applying this allowed model to reach a score of **27.67 MAE** on the test set and **2362** place (Fig. 16).

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
27.6731	2362	8610	2 of 3

Figure 16: Result of the model

6 LSTM

6.1 LSTM - introduction

Long Short Term Memory networks – usually just called "**LSTMs**" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by *Hochreiter & Schmidhuber (1997)* [3]. LSTMs are designed to avoid long-term dependency problems. They are using so-called *"memory cells"* to store information for a long period of time. Because this dataset is structured as a timeseries data, with test set from a different distribution, LSTM seems like the right idea to apply for the problem.

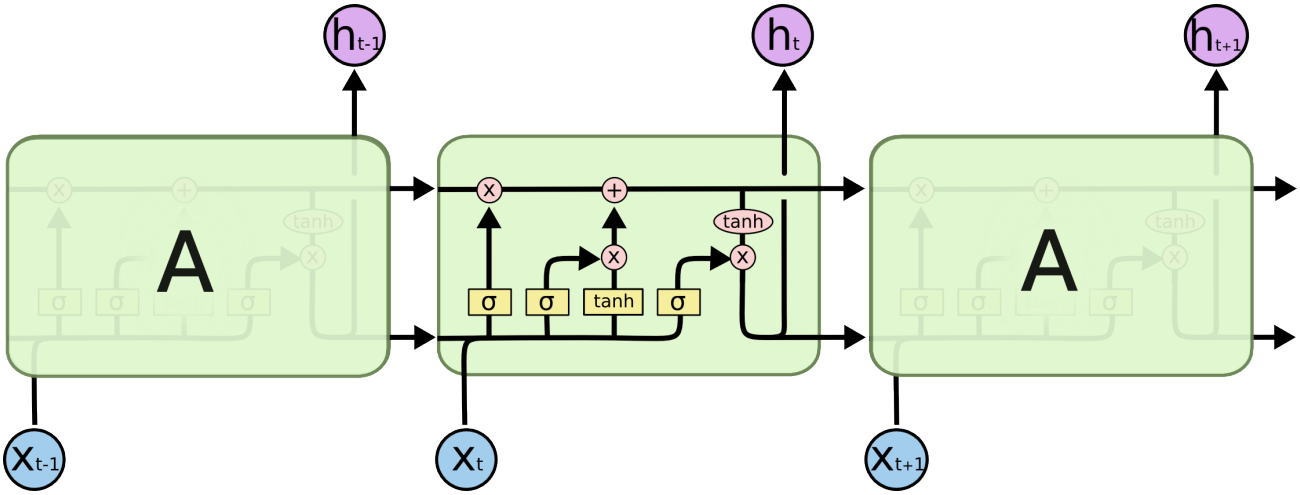

Figure 17: The repeating module in an LSTM contains four interacting layers. [11]

Because LSTM requires time series as an input I'm using an additional preprocessing method to generate multivariate lstm data (defined in *preprocessing_helpers.py*) (Section 4.2).

6.2 Base LSTM model

For a baseline LSTM model I'm using **32 nodes** version of LSTM. Because LSTM works better with RMSProp optimizer [8], this optimizer was chosen by default. **History size** is set to be **3**. This time two models are created to be trained for different cities. Because of that, we can achieve better results, but it's difficult to compare MAE between 2 models and 1 baseline model. After training base model for 10 epochs validation MAE stops decreasing at **11.2 MAE** (Fig. 18). Please notice that train loss is significantly higher than in the baseline case.

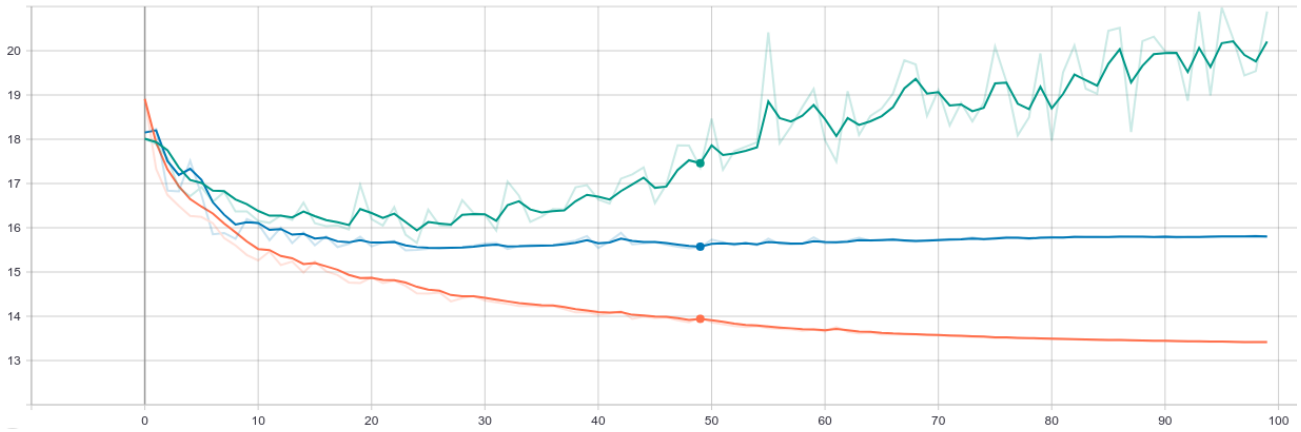

Figure 18: Point of early stoping (**grey** - train MAE, **oragne** - val MAE

Submission process looks different as well, this time there are 2 different outputs for the dataset. They have to be concatenated and submitter as one *.csv* file. Base LSTM model achieves **26.73 MAE** on testset which is already slightly better than baseline and moves position to **2005** (Fig. 19)

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
27.6731	2362	8610	2 of 3

Figure 19: Result of the model

.

6.3 LSTM with interpolated NDVI data

As described in section 4.3, another idea was to use interpolated NDVI data to improve model performance. Except that new values, all model parameters remain the same. Unfortunately adding NDVI data to LSTM didn't change the best result and even made it slightly worse. At this point, we know that LSTM is better than the default model but it's harder to find additional correlation in the data.

7 Time series on multi-layer NN

LSTM solution was a good idea but because of how LSTM works it's harder to overfit. Previous ideas were good enough in terms of data processing and treating data as a time series. Now, LSTM mode will be replaced with standard multi-multilayer NN. To avoid overtraining I'm introducing **Dropout** [9] **0.5**.

```
    Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================

_________________________________________________________________
input                        (None, 338)                0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               86784
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 128)               32896
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 119,809
Trainable params: 119,809
Non-trainable params: 0
```

The structure of the model has changed a little. This time model is larger than baseline and input contains **338 features**. We are using only 13 features from the original dataset but also using 26 history records at the same time. Now model can find more complex relations between values in dataset.

As in the previous case, I'm creating two models (each for one city) and training them with data from that city.

It's the right place to notice that some of the records don't have "previous" records. To fix that problem I'm introducing prepending datasets. If we take the training dataset and prepend it with the same dataset we can get missing 25 history features from the top of that dateset. This might not make sense but it's more useful when thinking about test dataset which starts exactly when the training dataset ends. That allows the model to carry some useful information for predicting test dataset values.

```
--- Train dataset SJ city---
city,year,weekofyear
...
sj,2008,8
sj,2008,9
sj,2008,10
sj,2008,11
sj,2008,12
sj,2008,13
sj,2008,14
```

```
sj,2008,15
sj,2008,16
sj,2008,17

--- Test dataset SJ city---
city,year,weekofyear
sj,2008,18
sj,2008,19
sj,2008,20
sj,2008,21
sj,2008,22
sj,2008,23
sj,2008,24
...
```

From this point onwards I'm not using the validation dataset because it wasn't helpful with predicting performance on the test dataset (test dataset comes from a different distribution). There is no early stopping as well (we don't know where to stop) (Fig. 20).

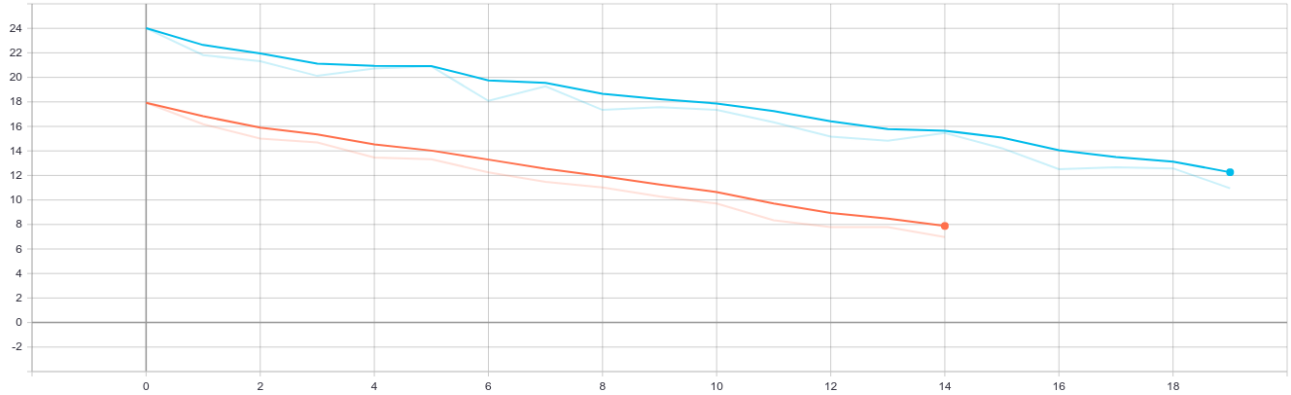

Figure 20: (**blue** - SJ train MAE, **oragne** - IQ train MAE

This model achieves **23.04 MAE** on testset which is only 3MAE better than LSTM but moves possition to **492** (Fig. 21). The reason for that is because most of the participants have scored around 25MAE which refers to proper model definition and dataset processing. From this point onward goal is to fit model into the competition rather than to create the best model in general.

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
23.0433	492	8975	1 of 3

Figure 21: Result of the model

.
During the process of training, I've tried different model sizes, optimizers, and sets of features. Set described above worked best on test dataset.

8 Grouping vectors idea

After going through some dengue papers and checking additional information about mosquitos I've decided to change the size of vectors for each feature. That size should refer to the right amount of data needed to affect the mosquitos population. Because two cities contained in the dataset are positioned in different environmental areas I'm referring to two studies from those areas [**dengue-car**] [12]. Each study tries to model dengue cases in a similar place so we can extract the right feature history length.

Because studies refer to its features as simple "temperature" or "humidity" I have to select only one of the list of features available. At the end here is a list of meaningful features for each city:

```
City = SJ
feature name - hisotry length
'reanalysis_sat_precip_amt_mm' - 7
'reanalysis_relative_humidity_percent' - 7
'station_avg_temp_c' - 4
'reanalysis_min_air_temp_k' - 4
'weekofyear' - 1

City = IQ
feature name - hisotry length
'reanalysis_sat_precip_amt_mm' - 12
'reanalysis_relative_humidity_percent' - 12
'station_avg_temp_c' - 4
'reanalysis_min_air_temp_k' - 4
'weekofyear' - 1
```

This time I'm using different model sizes for each city because of different input sizes (**50x25x1** nodes for SJ and **70x35x1** for IQ). The model parameters and training the process remain the same.

By using different vector sizes, new model achieves **20.23 MAE** on the test dataset (Fig. 22).

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
20.2380	243	9004	1 of 3

Figure 22: Result of the model

.

8.1 Moving Average

Another idea was to use a moving average [6] with target information from the previous predictions as a feature. Even if that idea would be a good choice for the standard model it flattens the prediction curve and makes MAE even higher. I've tried with different average length but it didn't manage to improve the end result.

9 Predictive Power Score

9.1 Definition

As described in the previous section, selecting the right vector length for each feature might be beneficial. The only problem is a limit of submissions per day per person. If we want to try each combination of feature-history size pair form 52 week length it's going to take 800 years to complete. A far better idea is to calculate **PPS** (**Predictive Power Score**).

PPS allows to figure out which vector has the highest predictive possibility from a list of available vectors. The idea is quite simple:

1. Extract all vectors with maximum history length (51 in this case)

2. From each vector create create **length - 1** new vectors e.g.:
If there is a vector like **[1,2,3,4,5]** it's going to create vectors **[1,2,3,4,5], [1,2,3,4], [1,2,3], [1,2]**.

3. For each vector create new a model and train it using ONLY that vectors with the same size for this particular feature. All models should be the same in terms of model size and parameters. The only difference is the input size. Target remains the same and it's *total_cases*.

4. Calculate MAE for every feature and vector size and plot it on heatmap.

9.2 PPS metrics

After calculating MAE for all features and vector sizes we can generate PPS matrix (Fig. 23, Fig. 24). Each cell is a different **MAE score** for **feature** (column) and **history_length** (row). Additionally, lower values are brighter to be easily distinguishable from higher values (we're looking for the lowest value in each column).

Because we need only one row from each column (best score) we're left with a list like that (I've changed *weekofyear* to be **1** because it doesn't have any effect since it's the same value):

```
-- SJ Features --
    'precipitation_amt_mm': 40,
    'reanalysis_air_temp_k': 16,
    'reanalysis_avg_temp_k': 15,
    'reanalysis_dew_point_temp_k': 39,
    'reanalysis_max_air_temp_k': 12,
    'reanalysis_min_air_temp_k': 21,
    'reanalysis_precip_amt_kg_per_m2': 30,
    'reanalysis_relative_humidity_percent': 34,
    'reanalysis_sat_precip_amt_mm': 40,
    'reanalysis_specific_humidity_g_per_kg': 14,
    'reanalysis_tdtr_k': 21,
    'station_avg_temp_c': 41,
    'station_diur_temp_rng_c': 40,
    'station_max_temp_c': 37,
    'station_min_temp_c': 26,
```

```
    'station_precip_mm': 32,
    'weekofyear': 1

-- IQ Features --
    'precipitation_amt_mm': 33,
    'reanalysis_air_temp_k': 10,
    'reanalysis_avg_temp_k': 4,
    'reanalysis_dew_point_temp_k': 6,
    'reanalysis_max_air_temp_k': 41,
    'reanalysis_min_air_temp_k': 40,
    'reanalysis_precip_amt_kg_per_m2': 3,
    'reanalysis_relative_humidity_percent': 7,
    'reanalysis_sat_precip_amt_mm': 33,
    'reanalysis_specific_humidity_g_per_kg': 26,
    'reanalysis_tdtr_k': 34,
    'station_avg_temp_c': 40,
    'station_diur_temp_rng_c': 26,
    'station_max_temp_c': 39,
    'station_min_temp_c': 25,
    'station_precip_mm':10,
    'weekofyear': 1
```

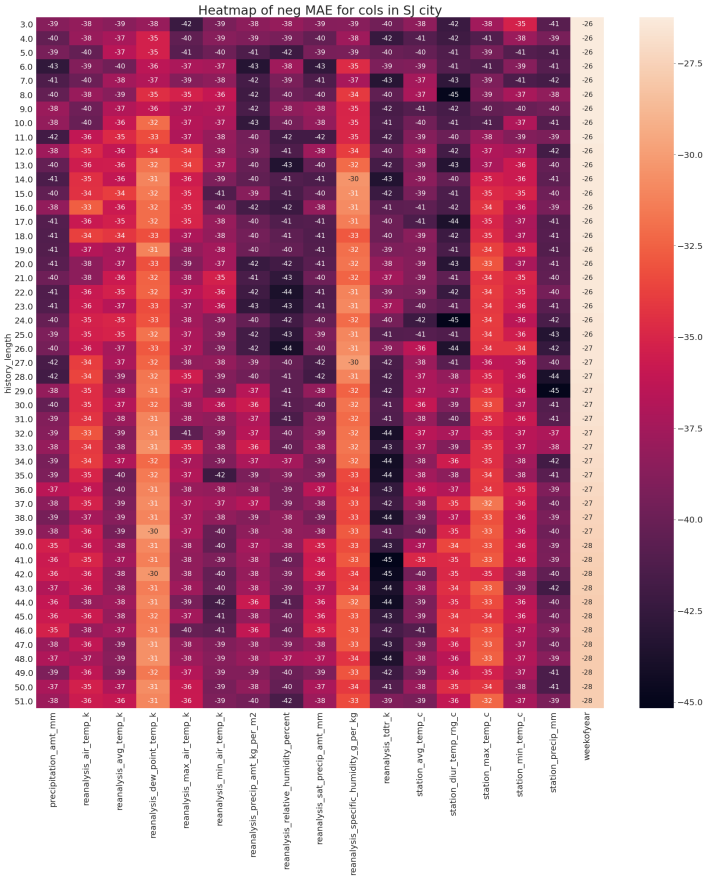

Figure 23: **sj** PPS

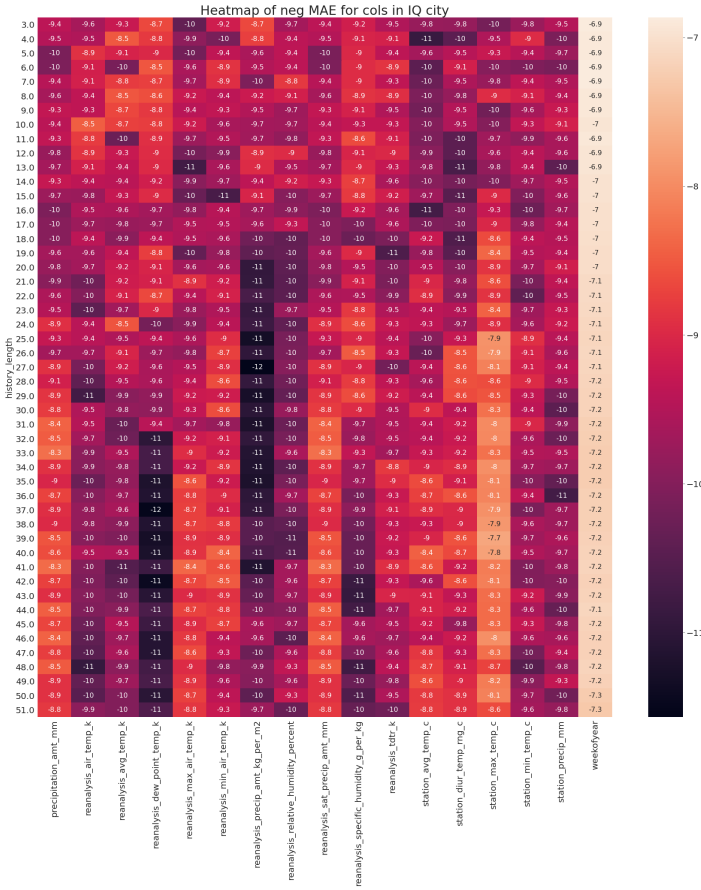

Figure 24: **iq** PPS

9.3 Applying PPS to model

After adjusting each feature vector length there is a need to change model size accordingly. At this point **sj** city has **459** input features and **iq** has **378** input features. Because of that I'm using **(300x150x1)** model for **sj** and **(280x140x1)** model for **iq**. The number of training epochs increases as well and it's 40 at this point.

Model performance on the test dataset increases to **17.57 MAE** and that changes current rank to **119** (Fig. 25).

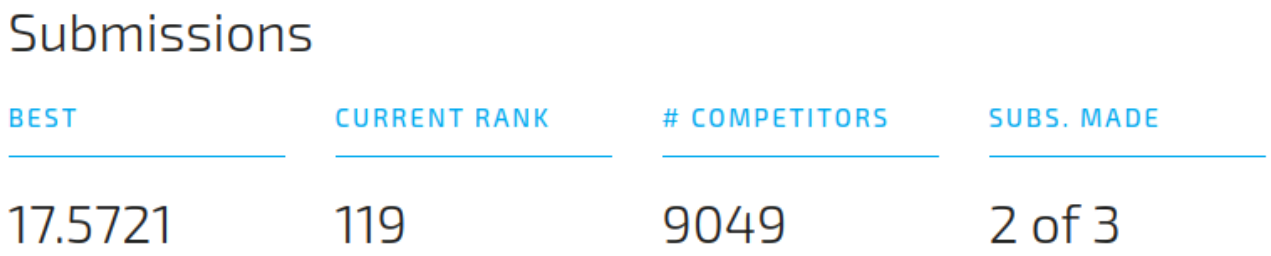

Figure 25: Result of the model

9.4 Pruning list of features

At this point, model achieves very low MAE on the training set (**3MAE**) so there is no point even looking on loss because it clearly overfits train dataset. Another idea to improve performance on test dataset is to prune the list of features base on PPS. To do this we're removing features with highest MAE. There is no one point we should stop doing this, a number of target features is an arbitrary number and has to be defined base on the experience. After removing only features with clearly higher MAE we're left with **11 features for sj** and **12 features for iq**. Because we're no longer trying to prevent overfitting, the new model is larger (**400x200x1 for sj**) and trained on even more epochs (**50**).

This achieves an even better score than the previous version and ends up with exactly **15MAE** adn **119** position in the ranking (Fig. 26. After some tweaks to the model parameters we can improve score a little bit and get **14.94 MAE** and **69** rank (Fig. 27).

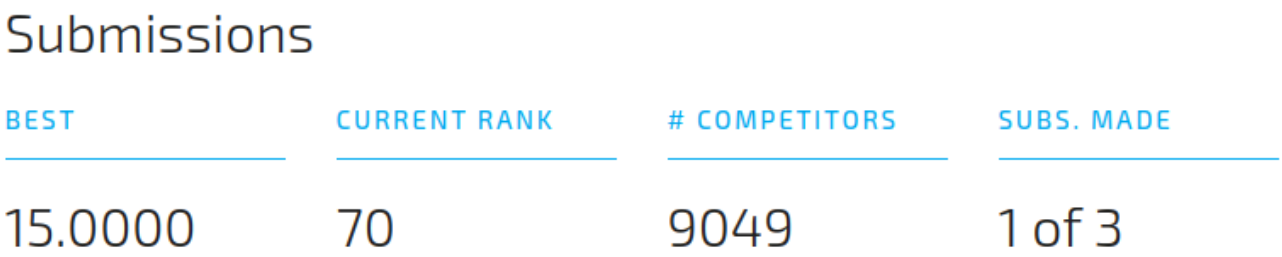

Figure 26: Result of the model

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
14.9423	69	9049	3 of 3

Figure 27: Result of the model

9.5 Alternating PPS models

Adjusting model size and parameters couldn't improve the score at this point anymore. Another idea was to use a different model for alternating years. This idea is to train two sets of models for each city, one for even and one for odd year numbers. That idea comes just from looking on the training dataset and it's not directly based on any experience (Fig. 28).

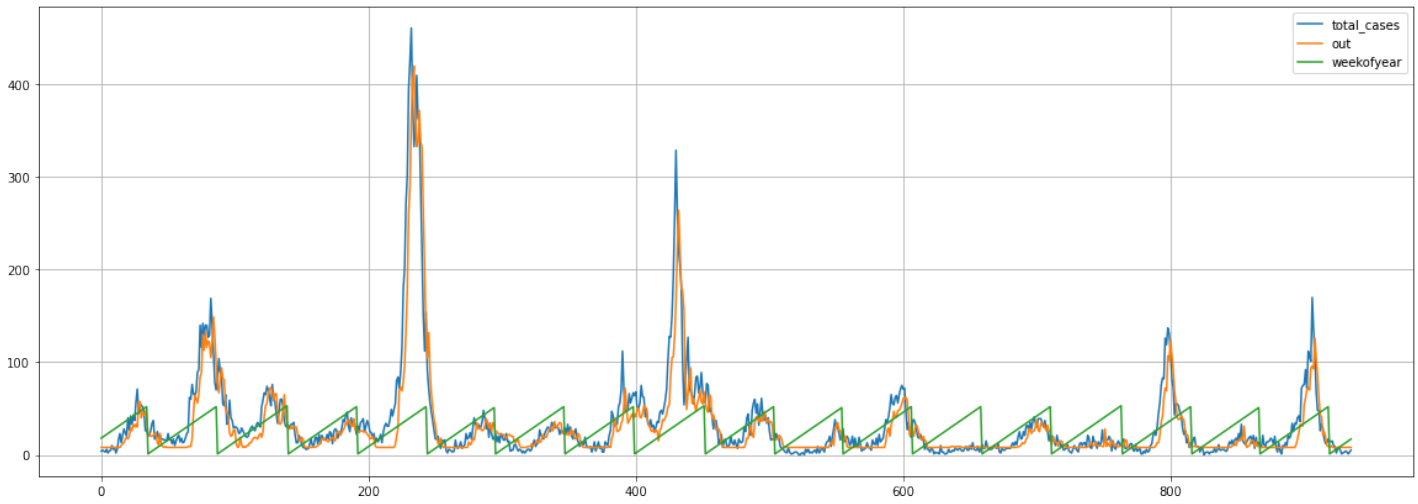

Figure 28: Train dataset *total_ cases* (**blue** - actual, **orange** - predicted)

Each alternating model is trained on alternating years from train dataset and then predictions are concatenated into one file. It's good to mention that year doesn't start at 1st of Jan. Each year starts from 18th week e.g. 2008 starts at 18th week of 2008, 2009 starts at 18th week of 2009 etc.

By doing that and testing a lot of different model sizes with different training parameters I've managed to improve score even more (Fig. 29).

Submissions

BEST	CURRENT RANK	# COMPETITORS	SUBS. MADE
13.5841	36	9051	2 of 3

Figure 29: Result of the model

A huge disadvantage is that in this case there is no indicator of how a model will perform on test dataset.

We can create a model with different sets of vectors based on PPS, with different model size and parameters and train MAE tells us nothing. Even worse, all models are overfitted to training dataset so MAEs are similar to each other. Because there is a limit of 3 submissions per day we cannot send all of them and test which one is the best for test dataset but I've tried around 30 different versions of alternating models with some good results (Fig. 30, Fig. 31, Fig. 32).

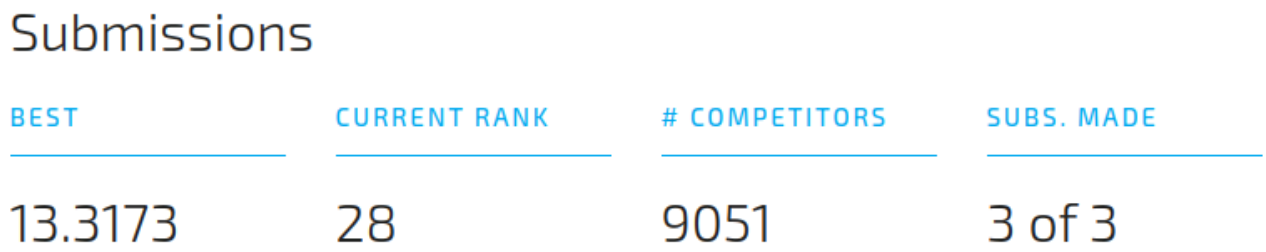

Figure 30: Result of the model

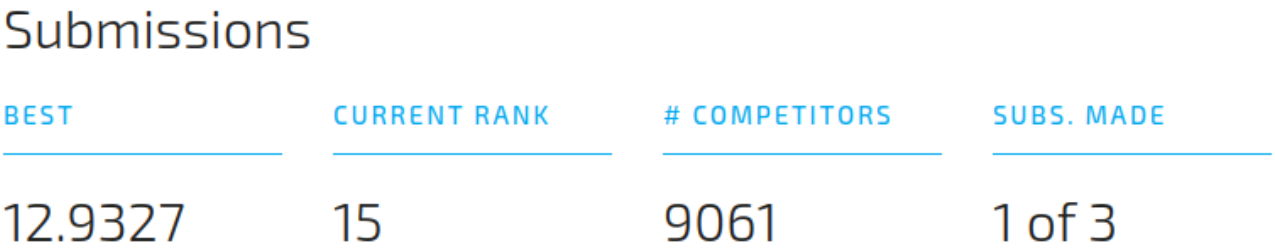

Figure 31: Result of the model

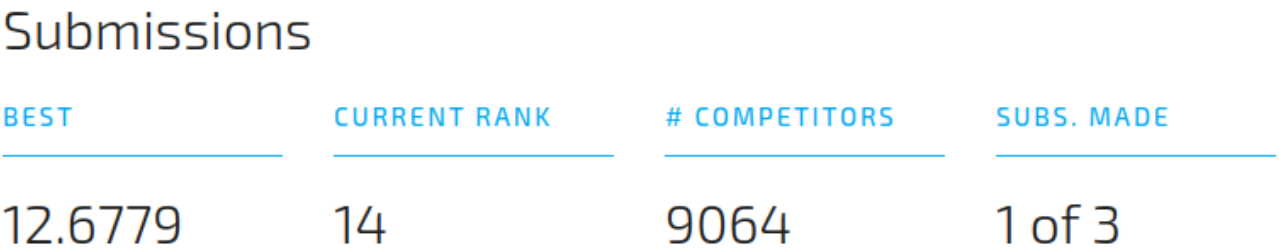

Figure 32: Result of the model

From the point of **12.68 MAE** i didn't manage to make any improvements by just changing model or input features. I didn't try all possible combinations, only some that seems to make sense.

10 Conclusion

After **54 submissions** I've managed to get **14th place** out of **9069 competitors** (02.06.2020) with a score of **12.6779 MAE**. The score didn't change much since 30th submission (13.32 MAE) because from this point onward it was mostly trying with different model settings rather than the ideas.

```
13.5841 burnpiro  2020-05
13.3173 burnpiro  2020-05
13.5361 burnpiro  2020-05
14.2788 burnpiro  2020-05
13.3750 burnpiro  2020-05
13.5697 burnpiro  2020-05
13.3654 burnpiro  2020-05
13.7356 burnpiro  2020-05
14.2596 burnpiro  2020-05
13.4303 burnpiro  2020-05
13.1731 burnpiro  2020-05
12.9327 burnpiro  2020-05
13.4303 burnpiro  2020-05
12.9279 burnpiro  2020-05
12.6779 burnpiro  2020-06    - BEST
13.8029 burnpiro  2020-06
13.1346 burnpiro  2020-06
13.1178 burnpiro  2020-06
13.4183 burnpiro  2020-06
12.7788 burnpiro  2020-06
```

There is no one idea that is responsible for the best score, it's more like an iterative process of getting to this point. But if I would have to select one method it would be **PPS** because it's useful to understand dataset even more and allows to score within **0.2% of competitors**.

References

[1] *Centers for Disease Control and Prevention*. URL: http://www.cdc.gov/.

[2] DrivenData. *DengAI: Predicting Disease Spread*. [Online; accessed 01-June-2020]. URL: https://www.drivendata.org/competitions/44/dengai-predicting-disease-spread/.

[3] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. URL: http://www.bioinf.jku.at/publications/older/2604.pdf.

[4] I.T. Jolliffe. *Principal Component Analysis*. 1986.

[5] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* (2008). URL: http://www.jmlr.org/papers/v9/vandermaaten08a.html.

[6] *Moving average - Wikipedia*. URL: https://en.wikipedia.org/wiki/Moving_average. (accessed: 02.06.2020).

[7] *National Oceanic and Atmospheric Administration*. URL: http://www.noaa.gov/.

[8] *RMSProp*. URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. (accessed: 02.06.2020).

[9] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[10] *U.S. Department of Commerce*. URL: https://www.commerce.gov/.

[11] *Understanding LSTM Networks - colah's blog*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. (accessed: 02.06.2020).

[12] Maria F. Vincenti-Gonzalez et al. "Spatial Analysis of Dengue Seroprevalence and Modeling of Transmission Risk Factors in a Dengue Hyperendemic City of Venezuela". In: *PLOS Neglected Tropical Diseases* 11.1 (Jan. 2017), pp. 1–21. DOI: 10.1371/journal.pntd.0005317. URL: https://doi.org/10.1371/journal.pntd.0005317.