Snowcast Showdown: Model report

Oleksii Poltavets

Summary

Our goal is to build a machine learning model that can estimate snow water equivalent (SWE) at 1km resolution. The data consist of a variety of data sources. To do this, we decided to use a neural network model with different layer architectures: Fourier neural operator[1] (FNO), convolution layers, embeddings, and linear transformation. This architecture allows us to combine data of different nature: images, historical data and classification labels. Also FNO can decrease the influence of errors in the data.

1. Data sources

1. Static data:
    a. FAO-UNESCO Global Soil Regions Map[2]
    b. Copernicus DEM (90 meter resolution)[3]
2. The High-Resolution Rapid Refresh[4] with the forecast hour 00
    a. The model cycle runtime t12 features:
        i. Entire atmosphere, Composite reflectivity [dB]
        ii. 80 m above ground, U-Component of Wind [m/s]
        iii. 80 m above ground, V-Component of Wind [m/s]
        iv. Surface, Temperature [K]
        v. Surface, Water Equivalent of Accumulated Snow Depth [kg/m^2]
        vi. Entire atmosphere (considered as a single layer), Precipitable Water [kg/m^2]
    b. The model cycle runtime t00 features:
        i. Surface, Temperature [K]

[1] https://arxiv.org/abs/2010.08895

[2] https://www.nrcs.usda.gov/wps/portal/nrcs/detail/soils/use/?cid=nrcs142p2_054013

[3] https://object.cloud.sdsc.edu/v1/AUTH_opentopography/www/metadata/Copernicus_metadata.pdf

[4] https://rapidrefresh.noaa.gov/hrrr/

ii. Surface, 0-0 day accumulation, Water Equivalent of Accumulated Snow Depth [kg/m^2]

3. MODIS/Aqua Snow Cover Daily[5]
4. Information about sun daylight duration - Calculus based on date and latitude.[6]

Data are selected according to criterias: availability, regularity and static data. Most of the satellite images were rejected because it is difficult to maintain up-to-date images for a large number of small cells and too many images are high cloud cover.

2. Feature engineering

Static data are images but have different pixel sizes. So this data is resized with rasterio[7] to 2D array with shape 10x10. Soil Regions Map contain too detailed information about cells, so information reencoded from suborder to order classification. Before processing soil information with convolutionals each cell is vectorized with an embedding layer.

MODIS data are also images, but processing them with convolutions after several experiments did not have a large impact on performance, only slowing down the execution time of the model. As a result, the data will be reduced to one point with the average information in a particular place.

The High-Resolution Rapid Refresh (HRRR) is a NOAA real-time 3-km resolution, hourly updated, cloud-resolving, convection-allowing atmospheric model, initialized by 3km grids with 3km radar assimilation. Grids don’t match with cells 1km resolution so we chose the points closest to the middle of each cell. Maximum distance between the middle of each cell and selected points does not exceed 2.12 km. Features were selected by the effect on RMSE after a short experiment and the ability to logically relate them to the target value.

The forward-fill method is used to fill missing values for all historical data. If it's not possible, fill it as zeros. Processed data into features with some sort of normalization, where most of the data will be in range -1..1, average are around 0 and STD not exceed 2. Outliers don’t clip, so some values can be in the -10..10 range.

---

[5] https://nsidc.org/data/MYD10A1

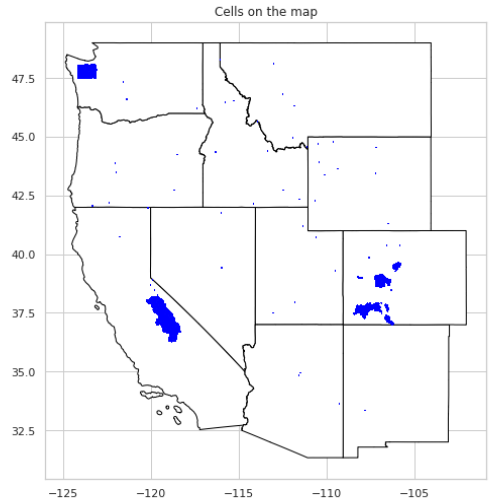
[6] https://gml.noaa.gov/grad/solcalc/calcdetails.html

[7] https://rasterio.readthedocs.io/en/latest/

3. Model

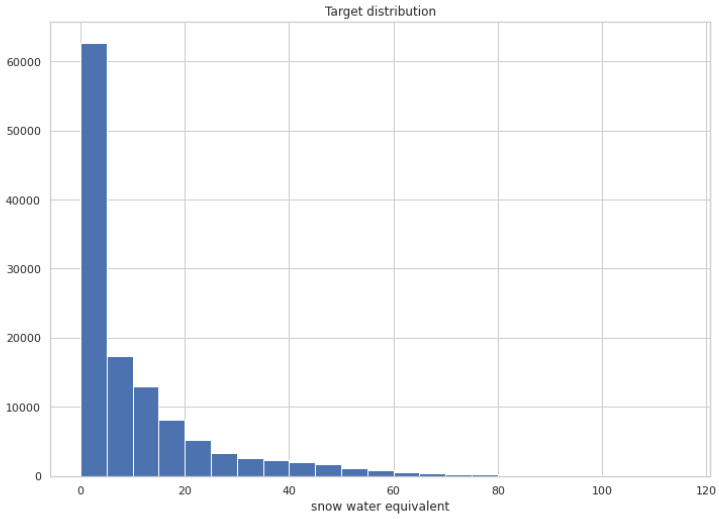
3.1. Training, validation.

The image on the right shows how cells are located on the map of the West U.S.



For training models used k-fold cross-validation with grouping targets by cell id. The library that provides this function is sklearn: StratifiedGroupKFold. Targets splitted into 5 folds with stratified distribution of targets (classes are whole tens: 7 - 0 class, 12 - 1 class, etc.). Final model uses the average of all 5 models trained by each fold. Loss functions for training models are MSE, but this loss function can produce huge gradients, so stability during the training process is obtained via applying gradient clipping. Optimizer method are Sharpness-Aware Minimization[8].

Numbers of cells with large values dramatically decrease with growing swe values. Right tail has a low number of targets, so error distribution in that range can not be interpreted as actual performance for large values, some folds have only few targets with large swe values in training cells of the fold split.



[8] https://github.com/davda54/sam

Model structure visualized with tensorboard in Appendix D.

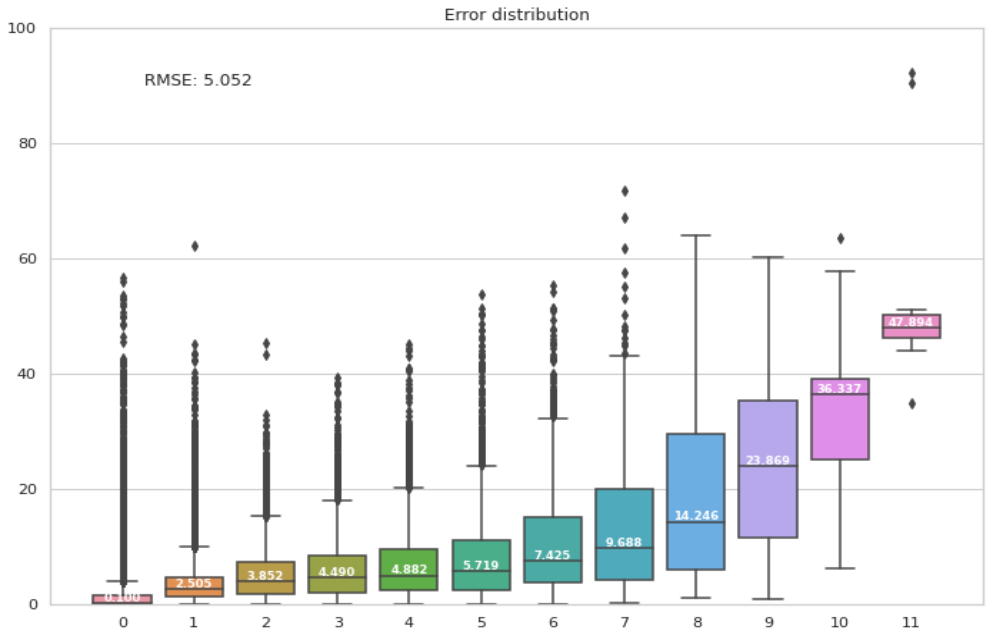
3.2.1. Model performance

General performance of model obtained via cross-fold prediction:

Model Summary

RMSE	R^2	Correlations		
		Pearson	Kendall	Spearman
5.05190	0.87796	0.93712	0.831049	0.937264

Error distribution (here and below x-axis classes means SWE range [0..10), [10..20), …, [110..inf)):



3.2.2. Performance across conditions

Cross-fold performance:

Fold №	**0**	**1**	**2**	**3**	**4**
RMSE	5.15467	4.76867	5.48453	5.33064	4.47224

*Detailed information about error distribution see in Appendix C.

Performance by years:

Year	2015	2016	2017	2018	2019	2020	2021
RMSE	3.619	5.234	8.191	4.394	6.511	3.698	3.319

*Detailed information about error distribution see in Appendix A.

Model error increases if the year was snowing and vice versa. Second reason for decreasing errors in the last 2 years can be improving the HRRR data gathering process. HRRRv3 from 12 July 2018.

Performance by region:

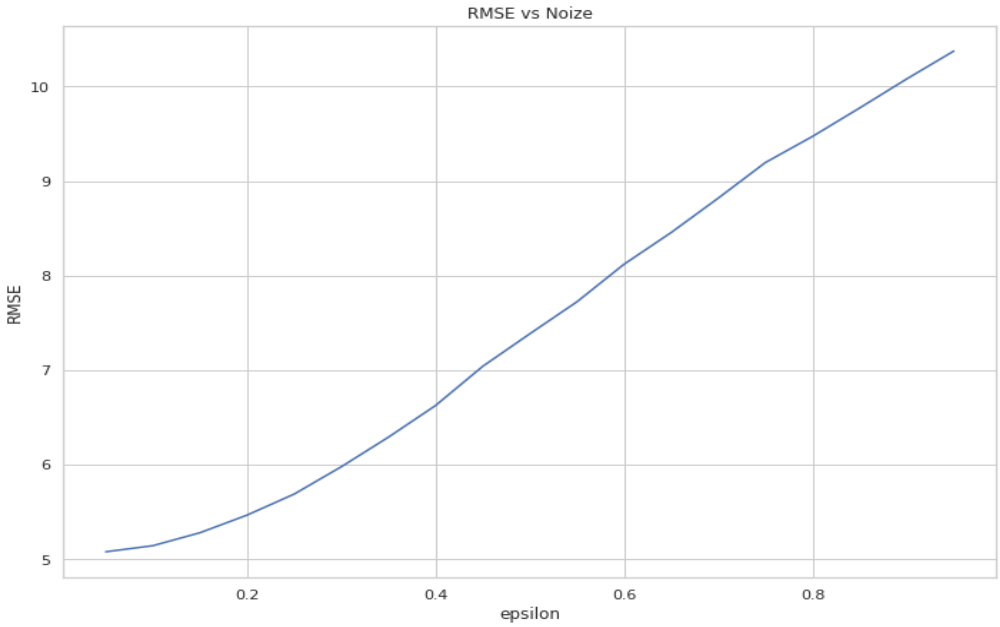
Region	Sierras	Central rockies	Other
RMSE	5.129	3.927	5.368

*Detailed information about error distribution see in Appendix B.

The increase in errors for a larger SWE has a statistical explanation: the underestimation ranges from 0 to the actual value, the overestimation ranges from the actual value to infinity. Larger values have large diapasons where the model can make errors. Also the number of cells with large swe values are low and errors don't represent generalized model results.

3.3. Robustness to error.

To assess the stability of the model forecasts, we launched a model with inputs with the addition of noise to it. The noise generated with torch.randn_like function, multiplied by STD of a particular feature, then multiplied by epsilon value. The result is in the image below.



Influence on errors for epsilon less than 0.1 are very small and become linear after epsilon more than 0.2. Small errors in the input features don't have influence and can be removed with the ability of FNO layers to removing small fluctuations in time series data.

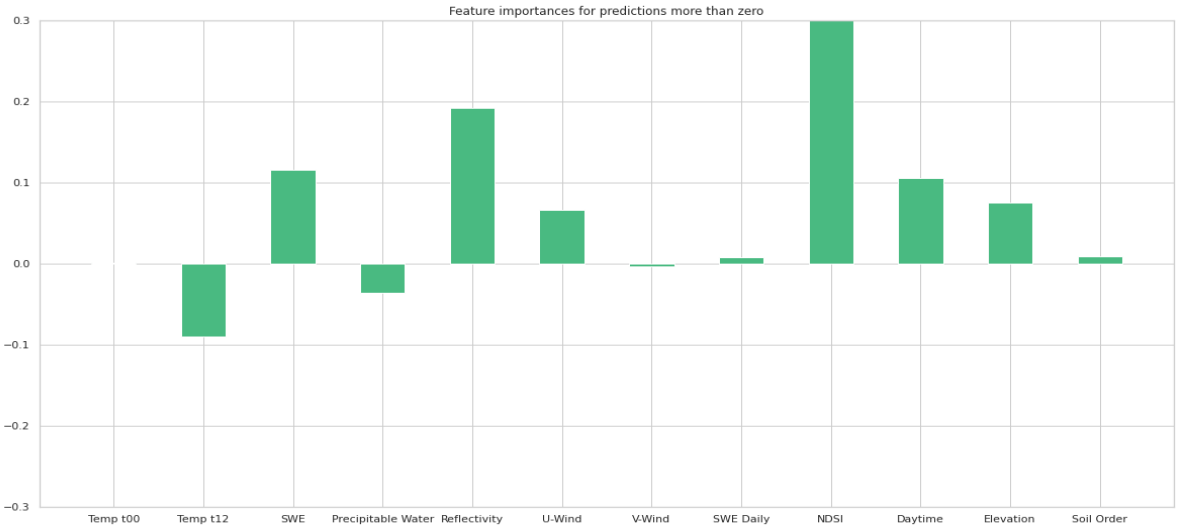
Second test was adversarial attack[9]. As expected, errors increase very fast. Reason for this is directly changing features with the biggest influence on forecast value. FGSM focused on gradient attack since gradients for mse loss function are big, prediction noticeably changed for small values of epsilon.

Epsilon	**0.01**	**0.03**	**0.05**	**0.10**
RMSE	9.188857	15.9411	20.064129	24.248375

3.4 Feature importances

[9] https://pytorch.org/tutorials/beginner/fgsm_tutorial.html

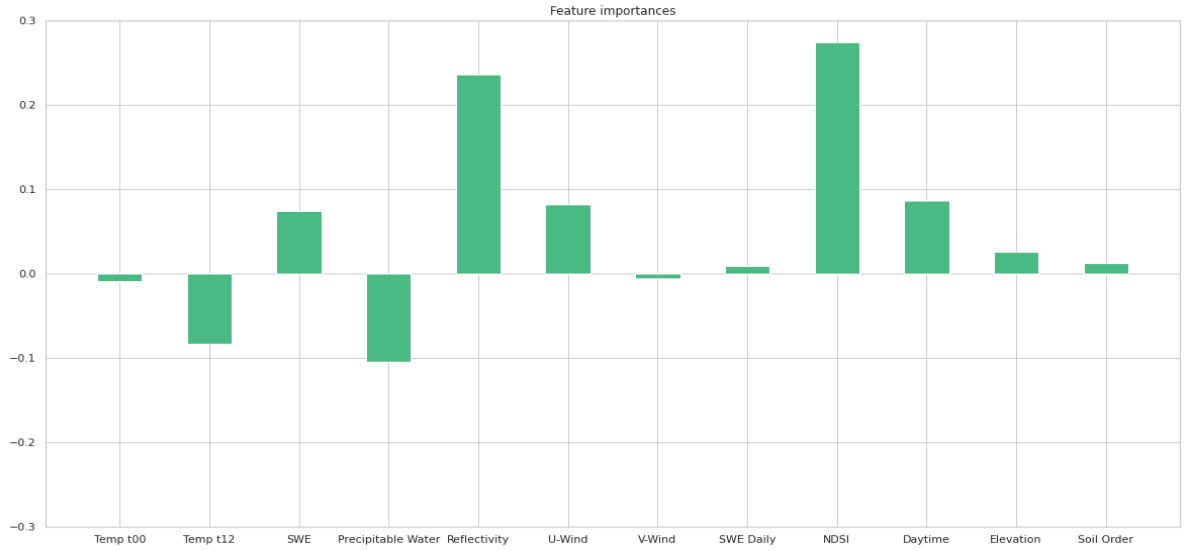
Target can be splitted in two different groups: with snow and without. So we calculate feature importances for these groups. Library for interpretation result are captum[10] and Integrated Gradients method[11].



Overall feature importance:

[10] https://captum.ai/tutorials/House_Prices_Regression_Interpret

[11] https://arxiv.org/pdf/1703.01365.pdf



Feature importance is a relative characteristic and shows how strongly a particular characteristic influence affects others. Signs indicate the direction of influences. “-” decrease predicted value, “+” increase.

Most important is MODIS/Aqua Snow Cover NDSI raw index, then composite reflectivity, SWE HRRR value. Big changes in the features importance for melted snow: precipitable water (HRRR), elevation/terrain and SWE index (HRRR).

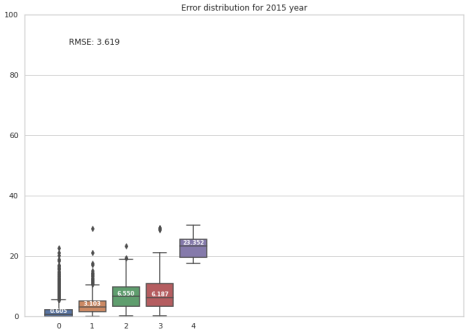
4. Machine specifications

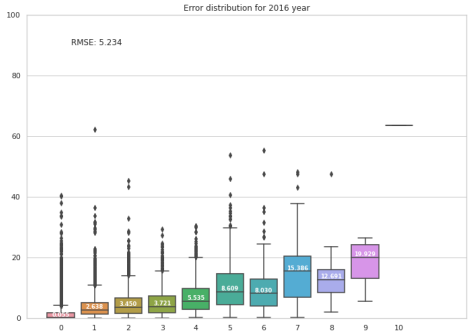
Model trained on a machine with 16GiB System memory and AMD Ryzen 5 5600X 6-Core. GPUs aren't used. Model is small and can be runned with only a few GiB memory, it depends on batch size (all evaluation cells can be estimated as one batch, more than 20,000 cells).

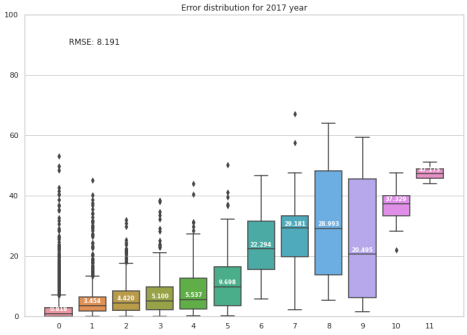
Also static data elevation/soil are fixed and don’t change in time, so can be precomputed and saved for reuse (its slightly decrease execution time). Using GPUs can considerably decrease execution time.

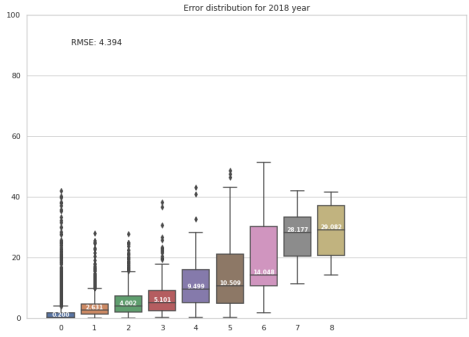
Appendix

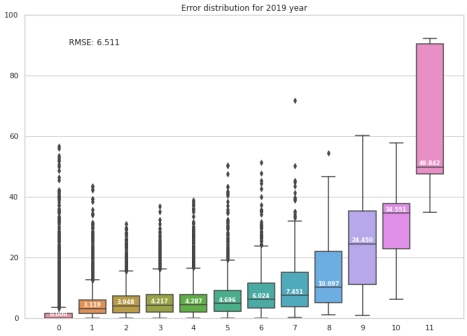
A. Distribution of errors by year.

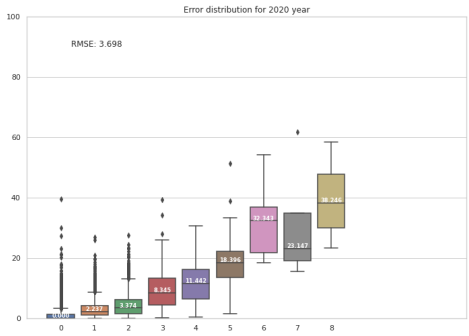


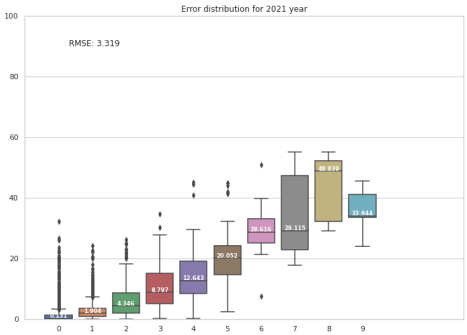




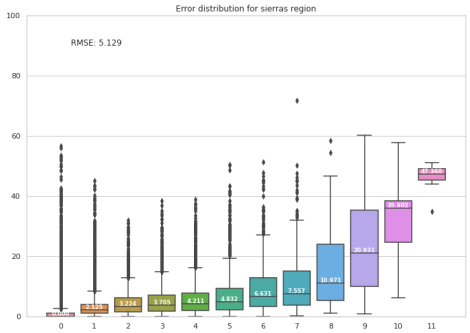


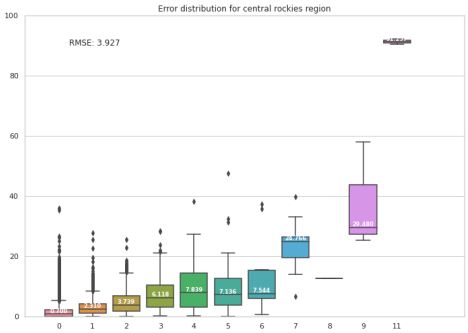


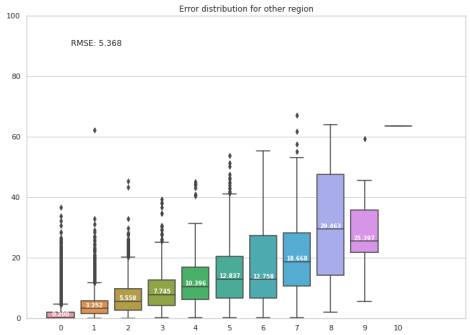




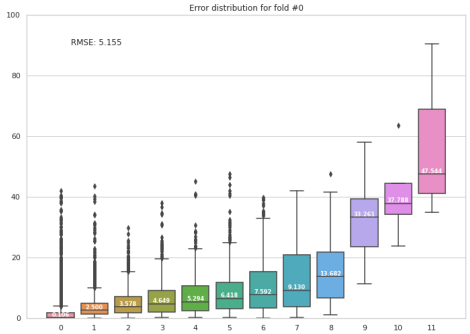
B. Distribution of errors by location.

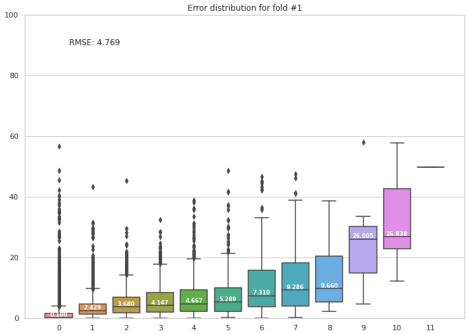


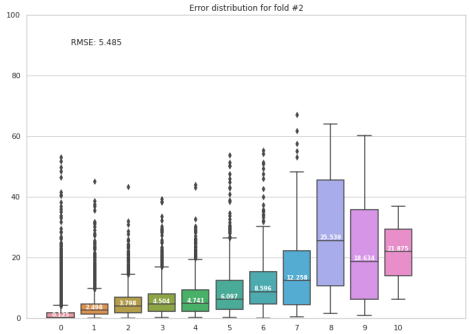


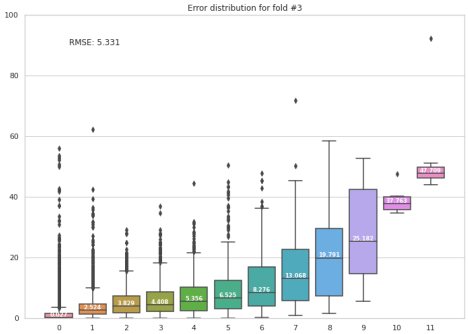


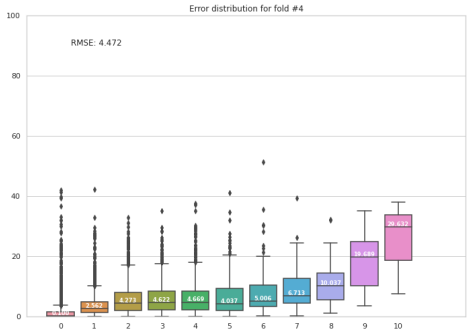
C. Distribution of errors by folds.











D. Model structure