1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

Roman Chernenko: Machine Learning Engineer in Agreena ApS company with 4 years of experience in ML for geospatial applications, satellite imaging and precision agriculture. Also, I have 10 years of experience with C++ cross-platform development, especially in the medical imaging domain, and embedded solutions.

Vitaly Bondar: ML Team lead in theMind (formerly Neuromation) company with 6 years of experience in ML/AI and almost 20 years of experience in the industry. Specialises in the CV and generative AI.

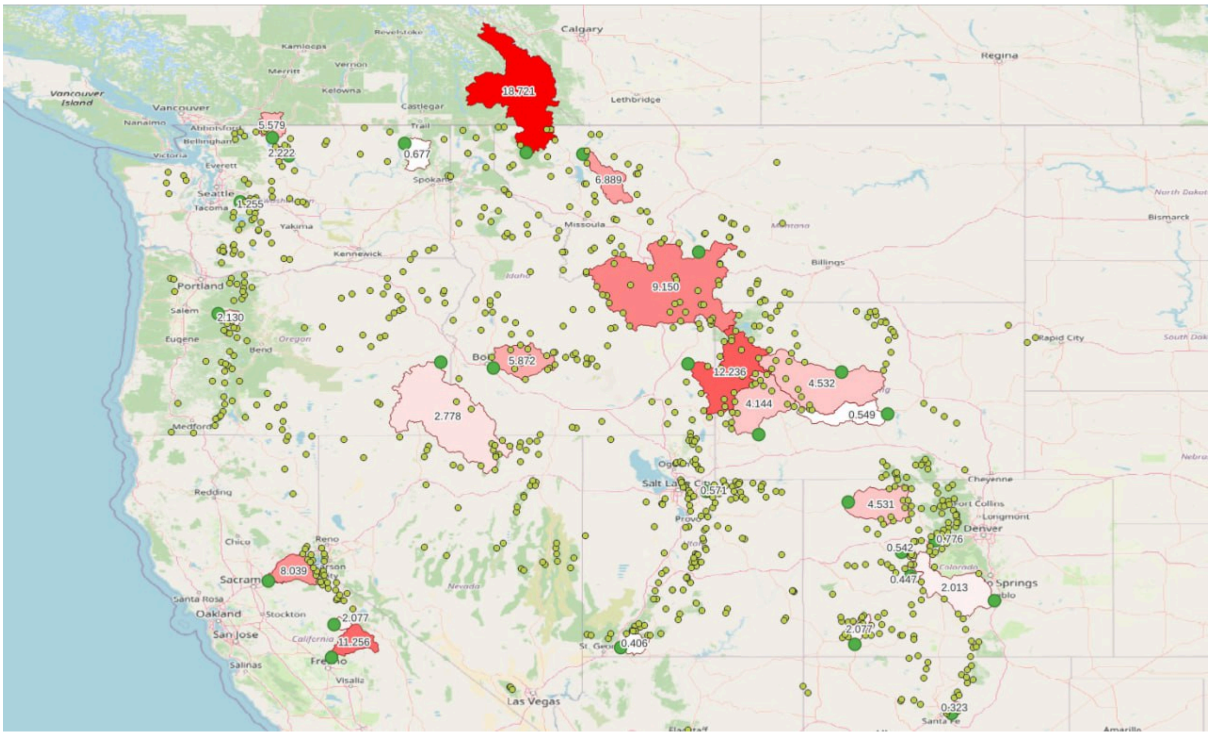
2. What motivated you to compete in this challenge?
The main motivation is a complicated interesting challenge with a significant prize amount. Also, we participated in a similar challenge “Snowcast Showdown” at DrivenData 2 years ago and finished 6th.

3. High level summary of your approach: what did you do and why?

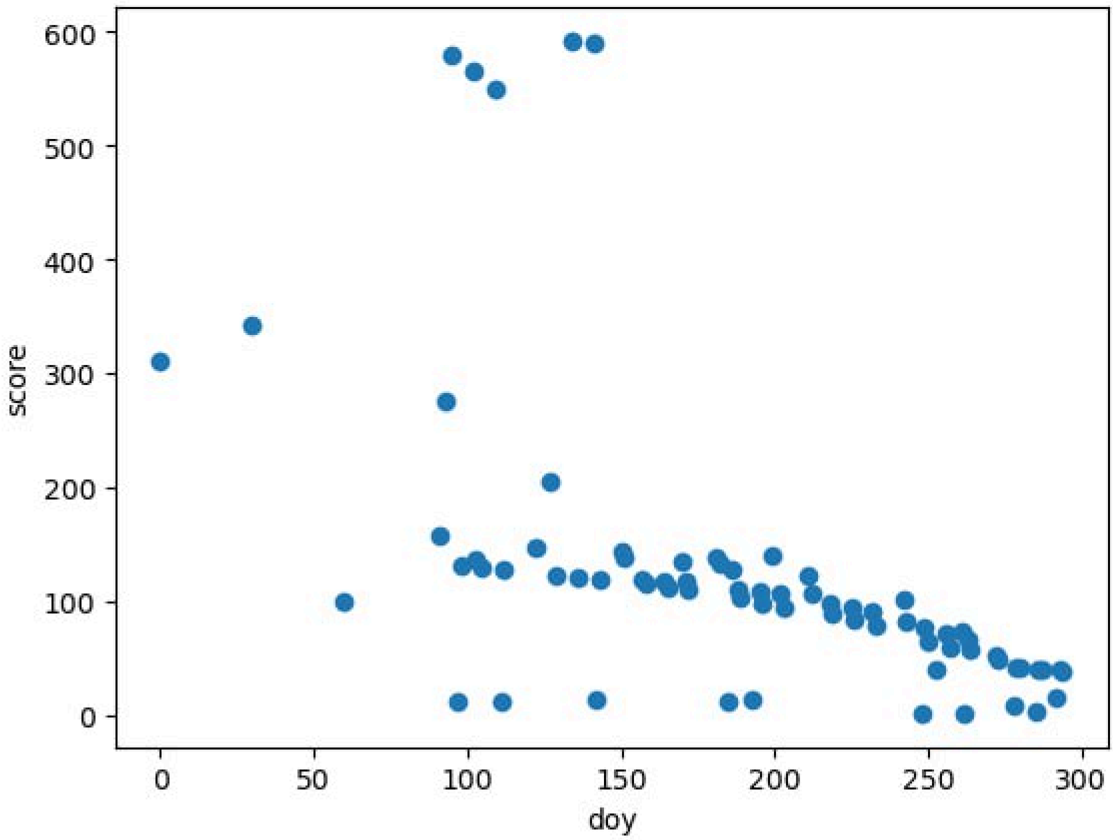
We recommend to refer our “Water Supply Forecast Rodeo” competition report by “ck-ua” team” document, where we made quite detailed explanations of data handling and ML decisions.

4. Do you have any useful charts, graphs, or visualizations from the process?

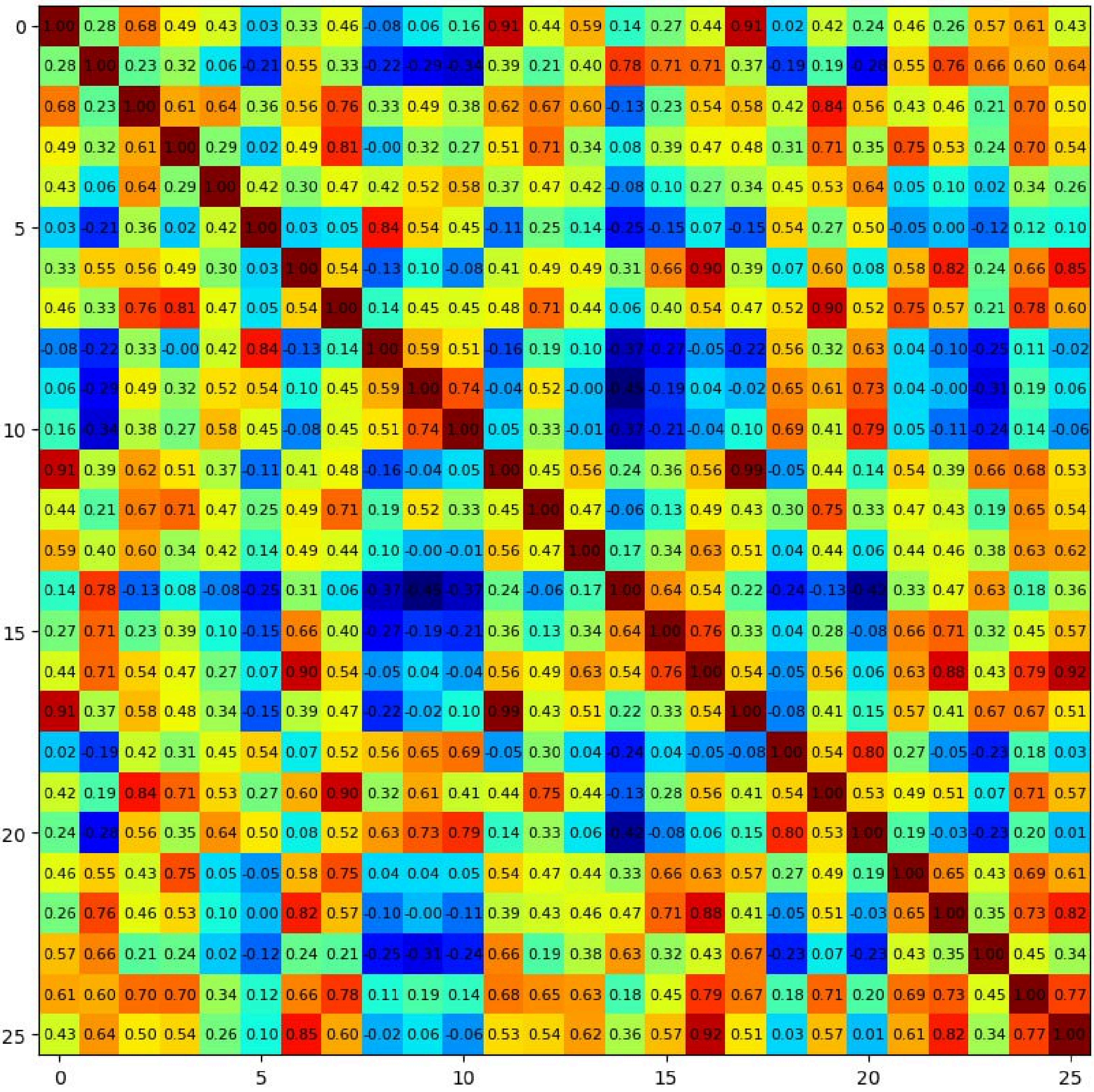
The individual addition to the score of each basin into the full score on the validation set. Basins are labeled with red polygons. Small circles - SNOTEL stations. Big green circles - sites in the training dataset.



The relationship between the number of days from starting of the water season and the score on the validation dataset.



The correlation matrix of target values between all 26 sites in the training dataset.



	0					5					10					15					20					25
0	1.00	0.28	0.68	0.49	0.43	0.03	0.33	0.46	-0.08	0.06	0.16	0.91	0.44	0.59	0.14	0.27	0.44	0.91	0.02	0.42	0.24	0.46	0.26	0.57	0.61	0.43
	0.28	1.00	0.23	0.32	0.06	-0.21	0.55	0.33	-0.22	-0.29	-0.34	0.39	0.21	0.40	0.78	0.71	0.71	0.37	-0.19	0.19	-0.28	0.55	0.76	0.66	0.60	0.64
	0.68	0.23	1.00	0.61	0.64	0.36	0.56	0.76	0.33	0.49	0.38	0.62	0.67	0.60	-0.13	0.23	0.54	0.58	0.42	0.84	0.56	0.43	0.46	0.21	0.70	0.50
	0.49	0.32	0.61	1.00	0.29	0.02	0.49	0.81	-0.00	0.32	0.27	0.51	0.71	0.34	0.08	0.39	0.47	0.48	0.31	0.71	0.35	0.75	0.53	0.24	0.70	0.54
	0.43	0.06	0.64	0.29	1.00	0.42	0.30	0.47	0.42	0.52	0.58	0.37	0.47	0.42	-0.08	0.10	0.27	0.34	0.45	0.53	0.64	0.05	0.10	0.02	0.34	0.26
5	0.03	-0.21	0.36	0.02	0.42	1.00	0.03	0.05	0.84	0.54	0.45	-0.11	0.25	0.14	-0.25	-0.15	0.07	-0.15	0.54	0.27	0.50	-0.05	0.00	-0.12	0.12	0.10
	0.33	0.55	0.56	0.49	0.30	0.03	1.00	0.54	-0.13	0.10	-0.08	0.41	0.49	0.49	0.31	0.66	0.90	0.39	0.07	0.60	0.08	0.58	0.82	0.24	0.66	0.85
	0.46	0.33	0.76	0.81	0.47	0.05	0.54	1.00	0.14	0.45	0.45	0.48	0.71	0.44	0.06	0.40	0.54	0.47	0.52	0.90	0.52	0.75	0.57	0.21	0.78	0.60
	-0.08	-0.22	0.33	-0.00	0.42	0.84	-0.13	0.14	1.00	0.59	0.51	-0.16	0.19	0.10	-0.37	-0.27	-0.05	-0.22	0.56	0.32	0.63	0.04	-0.10	-0.25	0.11	-0.02
	0.06	-0.29	0.49	0.32	0.52	0.54	0.10	0.45	0.59	1.00	0.74	-0.04	0.52	-0.00	-0.45	-0.19	0.04	-0.02	0.65	0.61	0.73	0.04	-0.00	-0.31	0.19	0.06
10	0.16	-0.34	0.38	0.27	0.58	0.45	-0.08	0.45	0.51	0.74	1.00	0.05	0.33	-0.01	-0.37	-0.21	-0.04	0.10	0.69	0.41	0.79	0.05	-0.11	-0.24	0.14	-0.06
	0.91	0.39	0.62	0.51	0.37	-0.11	0.41	0.48	-0.16	-0.04	0.05	1.00	0.45	0.56	0.24	0.36	0.56	0.99	-0.05	0.44	0.14	0.54	0.39	0.66	0.68	0.53
	0.44	0.21	0.67	0.71	0.47	0.25	0.49	0.71	0.19	0.52	0.33	0.45	1.00	0.47	-0.06	0.13	0.49	0.43	0.30	0.75	0.33	0.47	0.43	0.19	0.65	0.54
	0.59	0.40	0.60	0.34	0.42	0.14	0.49	0.44	0.10	-0.00	-0.01	0.56	0.47	1.00	0.17	0.34	0.63	0.51	0.04	0.44	0.06	0.44	0.46	0.38	0.63	0.62
	0.14	0.78	-0.13	0.08	-0.08	-0.25	0.31	0.06	-0.37	-0.45	-0.37	0.24	-0.06	0.17	1.00	0.64	0.54	0.22	-0.24	-0.13	-0.42	0.33	0.47	0.63	0.18	0.36
15	0.27	0.71	0.23	0.39	0.10	-0.15	0.66	0.40	-0.27	-0.19	-0.21	0.36	0.13	0.34	0.64	1.00	0.76	0.33	0.04	0.28	-0.08	0.66	0.71	0.32	0.45	0.57
	0.44	0.71	0.54	0.47	0.27	0.07	0.90	0.54	-0.05	0.04	-0.04	0.56	0.49	0.63	0.54	0.76	1.00	0.54	-0.05	0.56	0.06	0.63	0.88	0.43	0.79	0.92
	0.91	0.37	0.58	0.48	0.34	-0.15	0.39	0.47	-0.22	-0.02	0.10	0.99	0.43	0.51	0.22	0.33	0.54	1.00	-0.08	0.41	0.15	0.57	0.41	0.67	0.67	0.51
	0.02	-0.19	0.42	0.31	0.45	0.54	0.07	0.52	0.56	0.65	0.69	-0.05	0.30	0.04	-0.24	0.04	-0.05	-0.08	1.00	0.54	0.80	0.27	-0.05	-0.23	0.18	0.03
	0.42	0.19	0.84	0.71	0.53	0.27	0.60	0.90	0.32	0.61	0.41	0.44	0.75	0.44	-0.13	0.28	0.56	0.41	0.54	1.00	0.53	0.49	0.51	0.07	0.71	0.57
20	0.24	-0.28	0.56	0.35	0.64	0.50	0.08	0.52	0.63	0.73	0.79	0.14	0.33	0.06	-0.42	-0.08	0.06	0.15	0.80	0.53	1.00	0.19	-0.03	-0.23	0.20	0.01
	0.46	0.55	0.43	0.75	0.05	-0.05	0.58	0.75	0.04	0.04	0.05	0.54	0.47	0.44	0.33	0.66	0.63	0.57	0.27	0.49	0.19	1.00	0.65	0.43	0.69	0.61
	0.26	0.76	0.46	0.53	0.10	0.00	0.82	0.57	-0.10	-0.00	-0.11	0.39	0.43	0.46	0.47	0.71	0.88	0.41	-0.05	0.51	-0.03	0.65	1.00	0.35	0.73	0.82
	0.57	0.66	0.21	0.24	0.02	-0.12	0.24	0.21	-0.25	-0.31	-0.24	0.66	0.19	0.38	0.63	0.32	0.43	0.67	-0.23	0.07	-0.23	0.43	0.35	1.00	0.45	0.34
	0.61	0.60	0.70	0.70	0.34	0.12	0.66	0.78	0.11	0.19	0.14	0.68	0.65	0.63	0.18	0.45	0.79	0.67	0.18	0.71	0.20	0.69	0.73	0.45	1.00	0.77
25	0.43	0.64	0.50	0.54	0.26	0.10	0.85	0.60	-0.02	0.06	-0.06	0.53	0.54	0.62	0.36	0.57	0.92	0.51	0.03	0.57	0.01	0.61	0.82	0.34	0.77	1.00

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

The weird units transformations between USGS streamflow values to daily volume:

```
CUBIC_FOOT_m3 = 0.028316846592
ACREFOOT_m3 = 1233.48183754752
CFs_to_KAFd = CUBIC_FOOT_m3 * 3600 * 24 / ACREFOOT_m3 / 1000
ds['volume'] = ds['00060_Mean']*CFs_to_KAFd
```

Simple adding of one skipped connection from the input feature (current season water level) to the last layer significantly improves the score and training stability:

```
class MLPSumRes(MLP):
   def forward(self, x: torch.Tensor) -> Dict[str, torch.Tensor]:
       y = super().forward(x)["out"]
       y = y + x[:, 28].unsqueeze(-1)
       return {"out": y}
```

PyTorch implementation of mean quantile loss. We can train the model to predict the 10th percentile, mean value, and 90th percentile of the forecasted value simultaneously using the same model with this custom loss-function.

```
class PercentileLoss(nn.Module):
   def _percentile(self, predicted: torch.Tensor, target: torch.Tensor,
percentile: float):
       return 2*(percentile*torch.clamp(target-predicted, min=0.0) +
(1-percentile)*torch.clamp(predicted-target, min=0.0)).mean()

   def forward(self, predicted: torch.Tensor, target: torch.Tensor):
       return (
           self._percentile(predicted[:, 0], target, 0.5) +
           self._percentile(predicted[:, 1], target, 0.1) +
           self._percentile(predicted[:, 2], target, 0.9)
       )/3.
```

6. Please provide the machine specs and time you used to run your model.

- CPU (model): `AMD Ryzen 7 5700G`
- GPU (model or N/A): N/A
- Memory (GB): `32 GB`
- OS: `Ubuntu 23.10`
- Train duration: `29m 23s`
- Inference duration: `1m 43s for all sites at 2024-01-01`

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

The point we found in the forecast stage only that we did not fix all seeds in the hindcast stage, so the models may give slightly different results after retraining.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

We used the QGIS software for geospatial data exploration.

9. How did you evaluate performance of the model other than the provided metric, if at all?

We implemented quantile loss as it was the main metric for the task. And then we use it as the loss during training, and as the metric during validation.
We did not use any additional metric.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

Incorporating dropout regularization into our solution always reduced the final score.

Additionally, our team explored some Bayesian-based approaches for the estimation of the full probability density function of the target value. But we faced many obstacles in the estimation of the posterior probability and we refused this direction.
Although we initially planned to explore the feasibility of small transformer models, but this direction remained unexplored due to time constraints.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

Possible improvements we see:

- add timeseries-like features and handle it with the convolutions or transformer based architecture
- usage of additional data from SNODAS model.
- including different kind of weather data
- including elevation data
- adding SNOTEL-like stations data from Canada

12. What simplifications could be made to run your solution faster without sacrificing significant accuracy?

An important aspect of our approach is fast inference. The model may handle all the Hindcast test set in 27 seconds on CPU, including features preparation.
But to address the exact question - it is possible to remove the ensemble and use a single model from our approach and it will still have reasonable quality.