

Water Supply Forecast Rodeo: Final Cross Validation Report

By Matthew Aeschbacher

Abstract

Using historical data from 26 different hydrologic sites we created an ensemble of gradient boosting models that provide a probabilistic forecast for the 0.10, 0.50, and 0.90 quantiles of cumulative, seasonal streamflow volume. The models were designed to provide accurate predictions, robust to different types of stream sites. The models are also fast to train and predict and offer explainable predictions. There are two model architectures underlying the solution, each one implemented using two different gradient boosting on decision trees methods (Catboost¹ and LightGBM²) for a total of four models. The first model architecture predicts the quantiles of total cumulative streamflow volume in each season, for each site. The second model architecture predicts the quantiles of monthly cumulative streamflow volume within the season, for each site. The monthly predicted quantiles are then summed over the season to give the seasonal streamflow volume. The models use static features from each site including, the latitude, longitude, elevation, and categorical site id. The models also use dynamic features including, the day of the year, and features derived from localized precipitation, temperature, streamflow measurement from streamgages, and snow water equivalent measurements. The quantile predictions from each model are combined using a regularized regression to minimize the prediction error given by the mean quantile loss across the three quantiles. In this report we will further elaborate on the design decisions made while researching and developing the solution. We will also discuss the many steps taken to avoid leaking training data into test years during cross validation to provide a model that will perform well even outside of the cross-validation window.

Technical Approach

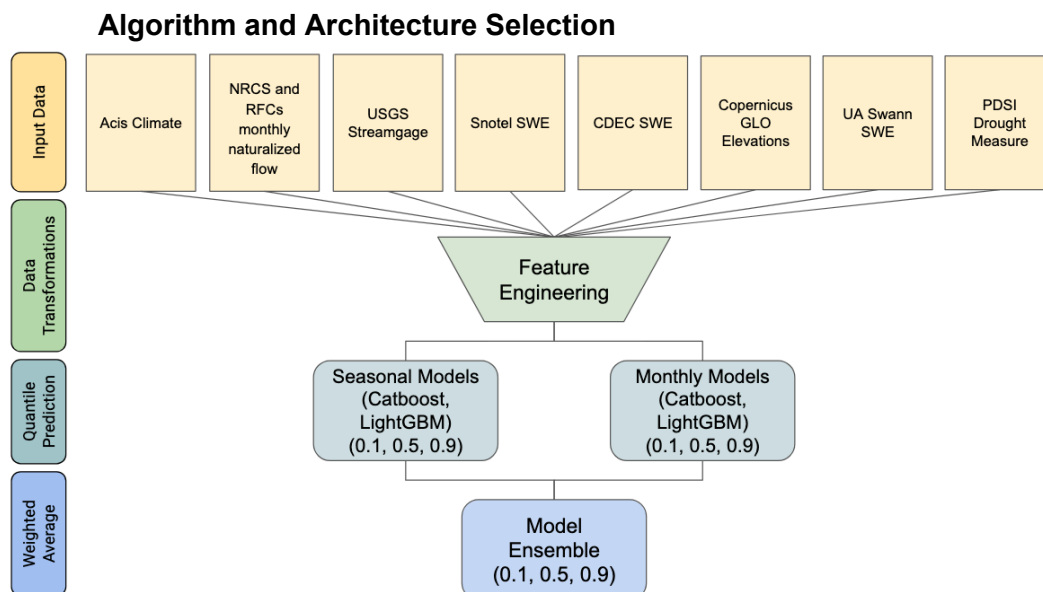


Figure 1: Model Architecture Diagram

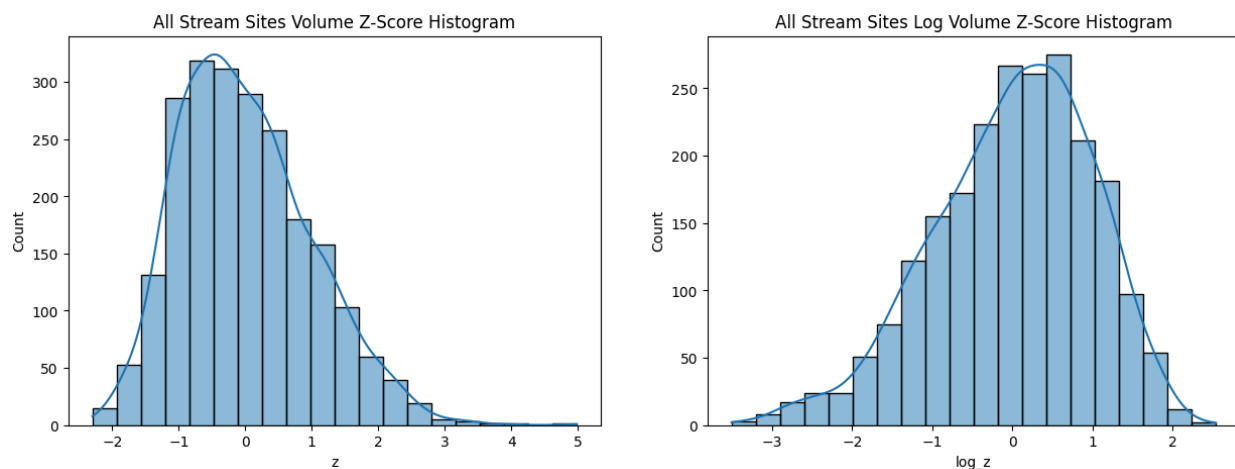
Overview

The solution uses two model architectures (seasonal and monthly) and two implementations of gradient boosting with decision trees (Catboost and LightGBM). Gradient boosting with decision trees is a supervised learning method.³ The models are trained on historical data consisting of the input features and historical labels (targets) that represent the ground truth. The target variable for the first model architecture is the seasonal, naturalized streamflow measurement for each site. The second model architecture uses the monthly, naturalized streamflow measurement for each applicable site as a target and the monthly values are then summed over the season. For each model architecture there is a separately trained model for each of the quantiles, 0.1, 0.5, 0.9. The predicted seasonal quantiles from the two model architectures are combined using a weighted average to get the final seasonal streamflow quantiles.

Log Volumes

Decision tree algorithms are insensitive to the scale of predictors (features) which means that performance is usually not improved by scaling features. However, in this case, scaling the target variable (label) with a log transform decreased the mean quantile loss of the models.

Depending on the model architecture, the target variable is the volume or monthly volume transformed using the natural logarithm. The models are trained to predict the log transformed target variable. During test execution the predictions are transformed back using the exponential function (the inverse transformation) to get the final predicted streamflow value for each quantile. Why did using the log transform on the target variable minimize the loss function? The cumulative seasonal streamflow for most sites has a skew-normal distribution with a longer tail and more outliers on the side of larger streamflow values. It may be that log transforming the streamflow values improves the predictions because the loss functions which minimize quantile loss are less strongly influenced by outlier values.



Why Two Model Architectures?

The model that is trained on the seasonal value of naturalized streamflow is necessary because some of the sites do not have monthly data labels but all of the sites have historical seasonal values. We also developed the monthly model because it offers an advantage when there are ground truth monthly observations within a season. The monthly model is trained to predict the

monthly quantile values for each of the relevant months within the streamflow measurement season. The quantiles for each of the months are then summed to obtain the total cumulative streamflow quantiles for the season. In the later months of the season, it is possible to observe ground truth labels for monthly streamflow that occurred in prior months within the season. Those ground truth values can be swapped with the monthly predictions, increasing the overall accuracy of the cumulative, seasonal streamflow prediction. Combining the predictions from the two models using a weighted average further lowers the mean quantile loss of the predictions. The seasonal model tends to work much better earlier in the season, while the monthly model dominates at the end of the streamflow season.

Gradient Boosting for Decision Trees

The gradient boosting trees algorithm was selected for initial experimentation because it trains quickly (allowing for rapid experimentation with feature engineering/selection), provides good baseline predictive accuracy, and offers many tools for interpreting the model. Gradient boosting with decision trees can also handle NaN value data inputs which greatly expands the possibilities for feature engineering and training data. We originally started by using the Xgboost implementation of gradient boosting with decision trees, which recently implemented quantile regression in the 2.0.0 release.⁴ We made progress minimizing the mean quantile loss using the Xgboost implementation but we found even better success experimenting with Catboost. During further model iterations we also implemented a model using LightGBM. LightGBM trains extremely fast and offers predictive accuracy comparable to Catboost.

Ensembling

An ensemble of the LightGBM and Catboost models further improves the score because ensembles tend to optimize for the bias-variance tradeoff. The Catboost and LightGBM implementations of gradient boosting for decision trees are not identical and they capture different and sometimes complementary variation from the training data. Ensembling the models is one of the many areas where it is very easy to leak holdout training data during cross-validation. The temptation is to fit the models while holding out the cross-validation subset and then ensemble the model using all information from all the labels. This potentially leads to overfitting. To avoid this situation, we fit a linear regression model of the LightGBM and Catboost models on only the training labels within the cross-validation run, while the test year for each iteration was still held out. This produced a much higher averaged mean quantile loss about 4 points higher than if we had fit the ensemble across all training years.

Catboost and Categorical Features

One downside to the Xgboost implementation of gradient boosting trees is that it doesn't easily handle categorical features. We tried using one-hot encoding of categorical variables but the performance was not significantly improved. We hypothesized that the categorical site_id for each stream site would be a useful feature for the model, allowing the model to build sub-trees unique to each stream site. The Catboost implementation of gradient boosting trees was designed with the intention of maximizing the utility of categorical features and also offers quantile regression.⁵ LightGBM also easily handles categorical features and quantile regression.

Future Improvements and Simplifications

The API for the Catboost and LightGBM algorithm is well-defined and designed for minimal hyperparameter tuning. One area that would simplify the implementation is to pick and use only one of the two GBT implementations. LightGBM trains the fastest and offers comparable accuracy to Catboost. An architecture based solely around LightGBM would not sacrifice more than a point of accuracy.

A limitation of gradient boosting for decision trees as a regressor is that it does not extrapolate. It is only capable of making predictions for a target variable within the distribution observed during the training process. For the 26 stream sites in this competition the inability to extrapolate did not appear to be an issue because there was enough training data to establish a sufficient distribution. However, the model might struggle predicting stream sites with less training data or could struggle in certain years that have extreme outlier streamflow particularly on the high side of the distribution. A possible mitigation is to train the model over a greater number of stream sites with a wide distribution of geographies and average streamflow volumes. Also GBT is a global model and some stream sites may perform better with only a subset of the features. Models localized to the streamsites when ensembled with the global GBT models could improve performance.

Other Models Considered

To explore a model capable of extrapolating outside the training distribution we experimented with a linear quantile regressor.⁶ The best performance for this model was achieved by identifying the three features that were most correlated with the target variable for each issue month, site_id pair. Then a single variable linear quantile regressor was trained for each feature, issue month, site_id. Finally the predictions for the three models using the most highly correlated features for each issue month, site_id pair were averaged. This model performed surprisingly well given its simplicity but it was not able to achieve the accuracy of the Catboost models. It also required NaN filtering and other complications in the data engineering pipeline. This model may perform better in cases where there is limited training data for a stream site and with further effort an ensemble of this model with the Catboost models could provide marginally improved predictive accuracy over the final solution proposed in this report.

Data Sources and Feature Engineering

One of the advantages of using gradient boosting for decision trees is that training models is straightforward and fast which allows for quick iteration while experimenting with different features. Most of the complexity in the solution is in the feature engineering that creates the inputs into the GBT models. While the source data for both models is the same, the actual features used by the seasonal and monthly model architectures are subject to different transformations. Recursive Feature Elimination with Cross-Validation (RFECV) was employed for feature selection. Features were chosen for each model using an iterative approach where the model was trained with and without the feature and the feature was kept if it improved the

model's loss score. The goal through research and development of the model was to explore as many of the approved data sources as possible.

Static Features

The solution uses both static (permanent characteristics of each site) and dynamic (time-sensitive) features. The static features from each site include, the latitude, longitude, elevation, and categorical site id.

- **site_id**: The categorical site identifier for each of the 26 streamflow sites.
- **pred_month**: The month to predict a naturalized streamflow value for (only used in the monthly model). Used as a categorical variable.
- **elevation**: The elevation where the streamflow measurement occurs for each site as provided in the metadata file.
- **elevation_stds**: The standard deviation of the elevation over the entire basin of each site. The source for this data is the Copernicus Digital Elevation Model GLO-90.⁷
- **latitude**: The latitude for each site as provided in the metadata file.
- **longitude**: The longitude for each site as provided in the metadata file.

Dynamic Features

The dynamic features include, the day of the year, and features derived from localized precipitation, temperature, streamflow measurement from streamgages, and snow water equivalent measurements.

Z-scores

Many of the dynamic features used by the model are averages of z-scores of physical measurements. The z-scores are calculated with respect to physical measurements from a specific station, in the case of SWE or climate data, a specific latitude/longitude, in the case of Palmer Drought measurements, or with respect to a specific stream gage. To obtain the z-scores individual measurements are normalized by subtracting the historical mean and dividing by the historical standard deviation.⁸ The historical mean and standard deviation are calculated with respect to a month_day combination and measurement location. These variables are suffixed with “_deviation” below. If the feature is an aggregation of multiple measurements, like the Snotel station measurements, then the historical mean and standard deviation are first calculated for each station and the z-scores for all the relevant stations on a given date are averaged with respect to the stream site. This is another area where it is possible to unintentionally leak data from training years into test years, via the aggregate statistics. To avoid leaking data we recompute the mean and standard deviation of the entire training dataset, holding out the test year, for each cross validation year. This added some additional overhead to generating the feature dataset for cross validation but ensured that no information from the training datasets were leaked into the test datasets.

Choosing Relevant Measurements for a Stream Site

During the fetching of raw data from the sources we downloaded data from stations or measurements within the site basin or close geographic proximity to the stream site and added

all the data to an aggregate file. However, a thesis of our model approach is that not all measurements are equally useful in predicting streamflow for a given site. For example, some Snotel stations may only be operational during some of the test years, or they are located in an area where snow water equivalent doesn't impact streamflow very much. Therefore, we seek to only use the most useful measurements in the z-score features. To do this we compute the correlation between a measurement and the seasonal streamflow volume, then select only the most highly correlated measurements. Yet again, this is an opportunity for data leakage. To avoid leakage, we compute the correlation between the seasonal streamflow volume and the measurements while holding out the test year for each iteration of cross validation. We then use the same measurement stations or locations to compute the test feature.

Windowing Measurements

The majority of the dynamic features were also transformed by taking a rolling average, for example taking the mean of the deviation features over the last 30 days before the issue date. The first day of each streamflow season is October 1st of the year preceding the forecast year.

Types of Dynamic Features

- **day_of_year:** The day of the year corresponding to the issue date.⁹
- **prev_month_volume:** The volume of the naturalized flow at the forecast sites for the month prior to the issue date. The source of this data is the NRCS¹⁰ and the RFCs¹¹.
- **precip_deviation:** The z-score of precipitation as measured by weather stations within the site basin using either a 30-day or season long window for the rolling average. The source for this data is the Applied Climate Information System (ACIS), maintained by the NOAA Regional Climate Centers (RCCs).¹²
- **max_temperature_deviation:** The z-score of max temperature as measured by weather stations within the site basin using either a 30-day or season long window for the rolling average. The source for this data is the Applied Climate Information System (ACIS), maintained by the NOAA Regional Climate Centers (RCCs).
- **combined_swe_deviation:** The z-score of snow water equivalent as measured by stations in sites located in or near the site basin using either a 30-day or season long window for the rolling average. There are two sources for this data. All of the sites used data from applicable NRCS Snow Telemetry (Snotel) stations.¹³ The California sites used data from applicable California Data Exchange Center (CDEC) stations.¹⁴
- **streamflow_deviation:** The z-score of streamflow measurements for each site or an applicable site. These measurements represent actual observed flow of water at specific locations, and not the naturalized flow being forecasted. The source of this data are streamgages managed by the U.S. Geological Survey (USGS).¹⁵
- **ua_swe_deviation & acc_water_deviation:** The z-score of University of Arizona Gridded Snow Water Equivalent (SWE) data and accumulated water data. These are estimations of SWE and accumulated water from models developed at the University of Arizona. The estimations are aggregated over the respective watersheds as determined by the Hydrologic Unit Code (HUC) for each site.¹⁶
- **pdsi_deviation:** The z-score of the Palmer Drought Severity Index (PDSI) measure.¹⁷ The PDSI is a standardized index based on a simplified soil water balance and estimates

relative soil moisture conditions. The magnitude of PDSI indicates the severity of the departure from normal conditions. The PDSI measures are localized to a specific site if they occur within the site basin.

The seasonal streamflow model performed better when the window for the rolling average of the dynamic features was longer. The monthly streamflow model performed better when the window for the rolling average of the dynamic features was the 30 days or less prior to the issue date.

Physical Explanation or Intuition

One of the driving intuitions for converting the physical measurements of temperature, precipitation, snow water equivalent, and streamflow into z-scores was to create unitless features that allow for easy aggregation across stations and increase the generality of the model between stream sites. The z-scores calculated for a measurement are invariant with respect to time and space, meaning that you wouldn't a priori expect the z-score for a measurement to increase at any specific date in the streamflow season or in different geographic regions. However, the z-scores are correlated in time and space, so you would expect the z-score of measurements near each other to rise and fall together. For these reasons the correlation between the features based on physical measurements and the target variable, the log of naturalized streamflow, increased after they were transformed into a z-score for each streamflow site.

Other Data Sources considered

Transforming the predictions from the Snow Data Assimilation System (Snodas)¹⁸ into a z-score and using it in the combined SWE feature improved the model loss function marginally. However, the Snodas data added significant complexity to the engineering pipeline because the data uses a cumbersome file and compression format that adds significant execution run time to the solution. Additional processing was required to restrict the predictions to specific site basins. We experimented with the CPC Seasonal Outlooks¹⁹ for precipitation, trying various transformations looking for correlations with streamflow volume. Ultimately the CPC Seasonal Outlooks didn't reduce the loss of the final model. We also tried using the climate teleconnection indices, specifically the Oceanic Nino Index (ONI) in the models.²⁰ This feature exhibited potential for early season predictions on streamflow, especially with certain stream sites. The correlations between streamflow volume and the ONI in early January and February were equal to or better than some of the features used for a handful of streamflow sites. This feature was useful in the simplified linear regression model discussed earlier in this report. However, this feature did not improve the Catboost models and therefore was left out of the final feature list.

Uncertainty Quantification

There are six total models, one model for each of the quantiles (0.1, 0.5, 0.9) of each architecture type (seasonal, monthly). Each of the models are Catboost implementations of gradient boosting with decision trees fit using a Quantile loss:

$$\frac{\sum_{i=1}^N (\alpha - I(t_i \leq a_i))(t_i - a_i)w_i}{\sum_{i=q}^N w_i}$$

Where α is a quantile in (0.1, 0.5, 0.9). The seasonal model architecture predicts the quantiles for the cumulative naturalized streamflow for the entire season. The monthly model architecture predicts the quantiles for the naturalized streamflow for each month in the season. The monthly predictions are then summed to calculate the final cumulative streamflow for the season. If there are observed values of naturalized streamflow in months prior to the month in which the issue date occurs, those are substituted for the quantile predictions. The quantiles for the cumulative seasonal streamflow predictions for each model type are combined using a weighted average that allocates a weight of 0.6 to the monthly model and 0.4 to the seasonal model for the first 4 months of issue dates, before allocating all of the prediction to the monthly model for the final 3 months of issue dates. For the models that do not have monthly labels (american_river_folsom_lake, merced_river_yosemite_at_pohono_bridge, san_joaquin_river_millerton_reservoir) the predicted quantiles from the seasonal streamflow models are used as the final predictions.

Training and Evaluation Process

Missing Values Processing and Historical Data

The first step in training the model was to generate training data. While some streamflow sites have seasonal training labels dating back to the early 1900's many of the data sources used to generate features do not have data going back that far. The Catboost implementation of gradient boosting on decision trees allows for null data in features, therefore, we can train on years where some of the features are null for all sites. We used Catboost's default implementation to process missing values, "missing values are processed as the minimum value (less than all other values) for the feature. It is guaranteed that a split that separates missing values from all other values is considered when selecting trees."²¹ This means that the model can recognize null data and train efficiently. However, the data points trained with null data may not apply to data points with non-null values for all features because they will take different paths in the decision trees. We are more likely to encounter data points with non-null features in the more recent years that make up the test population. So while training on historical data with null value features isn't likely to significantly hurt the models' performance, it isn't likely to help it either because the sub-trees that branch on null values won't apply to the data points that are not null. For that reason we chose a cut-off date of 1960 to generate training data, however we also experimented with 1980 (around the time that many Snotel stations started generating snow water equivalent measurements) and the performance difference was minimal.

Cross Validation

As described in other parts of this report there are 12 total models, one model for each quantile that predicts log-transformed seasonal streamflow, one model for each quantile that predicts

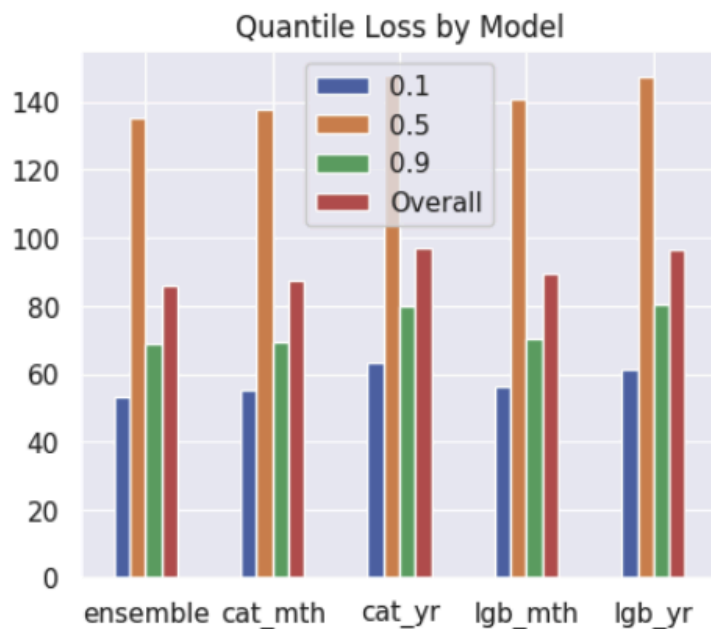
log-transformed monthly streamflow, and one model each implemented using Catboost and LightGBM. The predictions are then transformed using the exponential function to obtain values that apply to the actual labels. All model architecture decisions were made using leave one out cross validation over the years 2004-2023 for a total of 20 test folds. While the models were trained on all data after the year 1980 outside of the hold-out test year the models were only scored on the predictions obtained from the test year (hereinafter referred to as validation dataset). The cross validation iterated through all of the possible validation dataset permutations and then generated a score using mean quantile loss. The basis for all decisions on whether to include features, how to transform features or targets, how to ensemble the models, and hyperparameter tuning were made based on whether they lowered the cross-validation score for the model. If adding a feature significantly increased the complexity of the solution but only decreased the loss function by a small amount then that feature was not included in the final model.

Discussion of Performance

Overall Performance

The ensemble model was able to achieve a Mean Quantile Loss of 85.84. In many experiment runs it was possible to achieve far lower Mean Quantile Loss, but only by leaking training data due to imprudent cross-validation. The Quantile Loss for the monthly model is artificially lower because two stream sites with large volume, merced_river_yosemite at_pohono_bridge and american_river_folsom_lake, were not included. The Catboost model and LightGBM models performed very similarly but the combination of both performed the best. If we were to simplify the solution by choosing one model type we would choose LightGBM because it trains faster, even though it requires more attention to hyper parameter tuning.

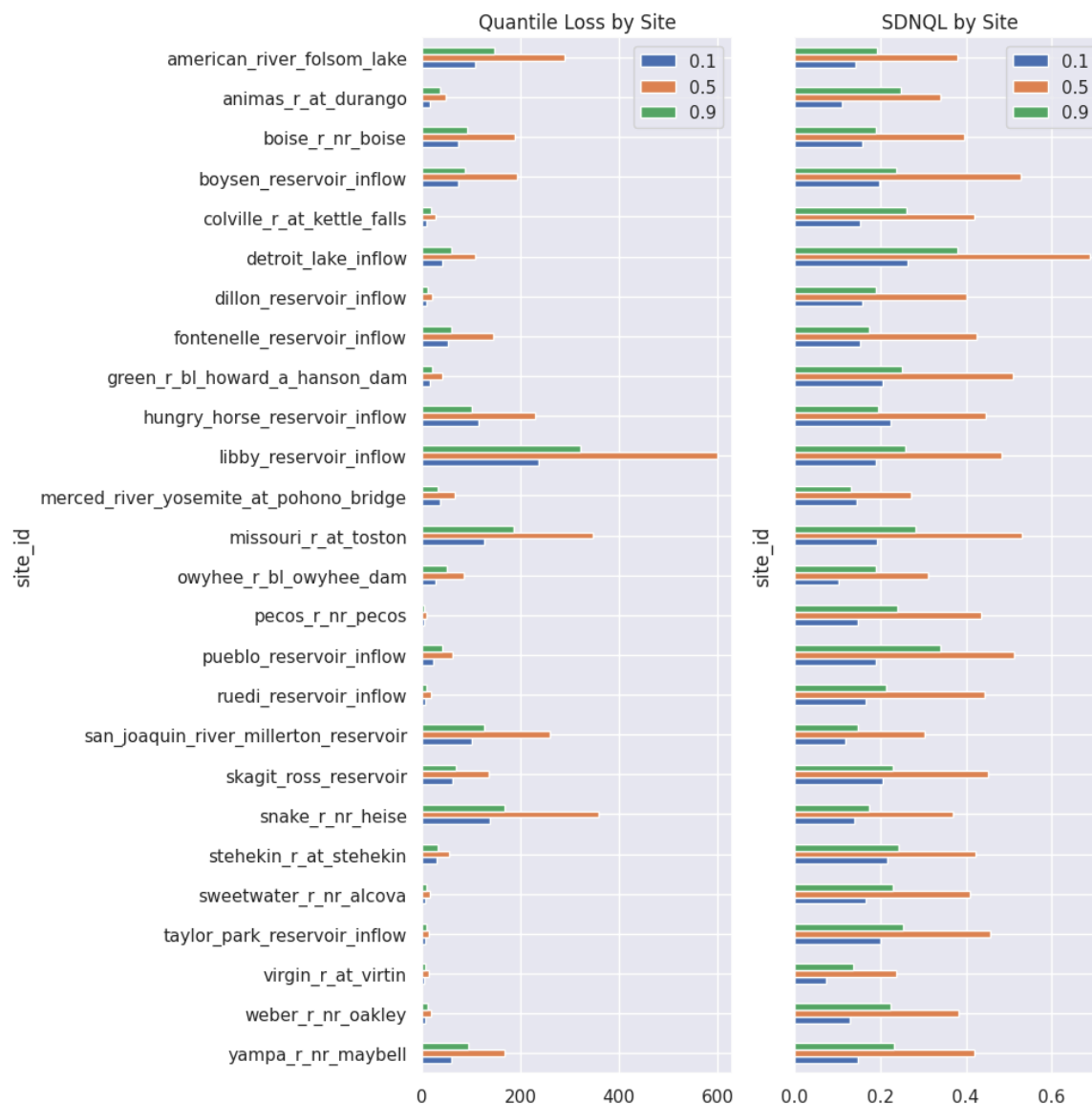
	ensemble	cat_mth	cat_yr	lgb_mth	lgb_yr
0.1	53.216	55.287	63.214	56.458	61.241
0.5	135.290	137.926	147.773	140.781	147.436
0.9	69.018	69.346	79.993	70.542	80.461
Overall	85.842	87.520	96.993	89.260	96.379



Location Specific Performance

We've introduced a few analysis techniques to enhance the analysis of model performance. To help analyze performance across different stream sites we needed a way to account for differing stream site volume. The stream sites vary greatly with respect to the total volume of seasonal streamflow. Stream sites with larger volumes will produce greater mean Quantile loss than

stream sites with lower volumes, even if the year to year variance of the larger volume stream site is better captured in the model. To correct for stream site volume we divide the Quantile loss by the historical standard deviation of streamflow volume for each site. This correction factor, hereinafter referred to as standard deviation normalized quantile loss (SDNQL), is further supported by the fact that the stream sites that were toughest to predict had lower correlations with many of our input features. In the figures below we can see that the largest volume stream site, libby_reservoir_inflow, is responsible for the largest loss. However, after normalizing the streamflow loss by the standard deviation of the stream site we see that the most difficult stream site to predict is detroit_lake_inflow. Briefly speculating, detroit_lake_inflow has a small drainage basin and is less correlated with snow water equivalent measurements. The variable that detroit_lake_inflow is most correlated (inversely) with is the maximum temperature throughout the streamflow season, but even here the correlation isn't as strong as with some other sites.

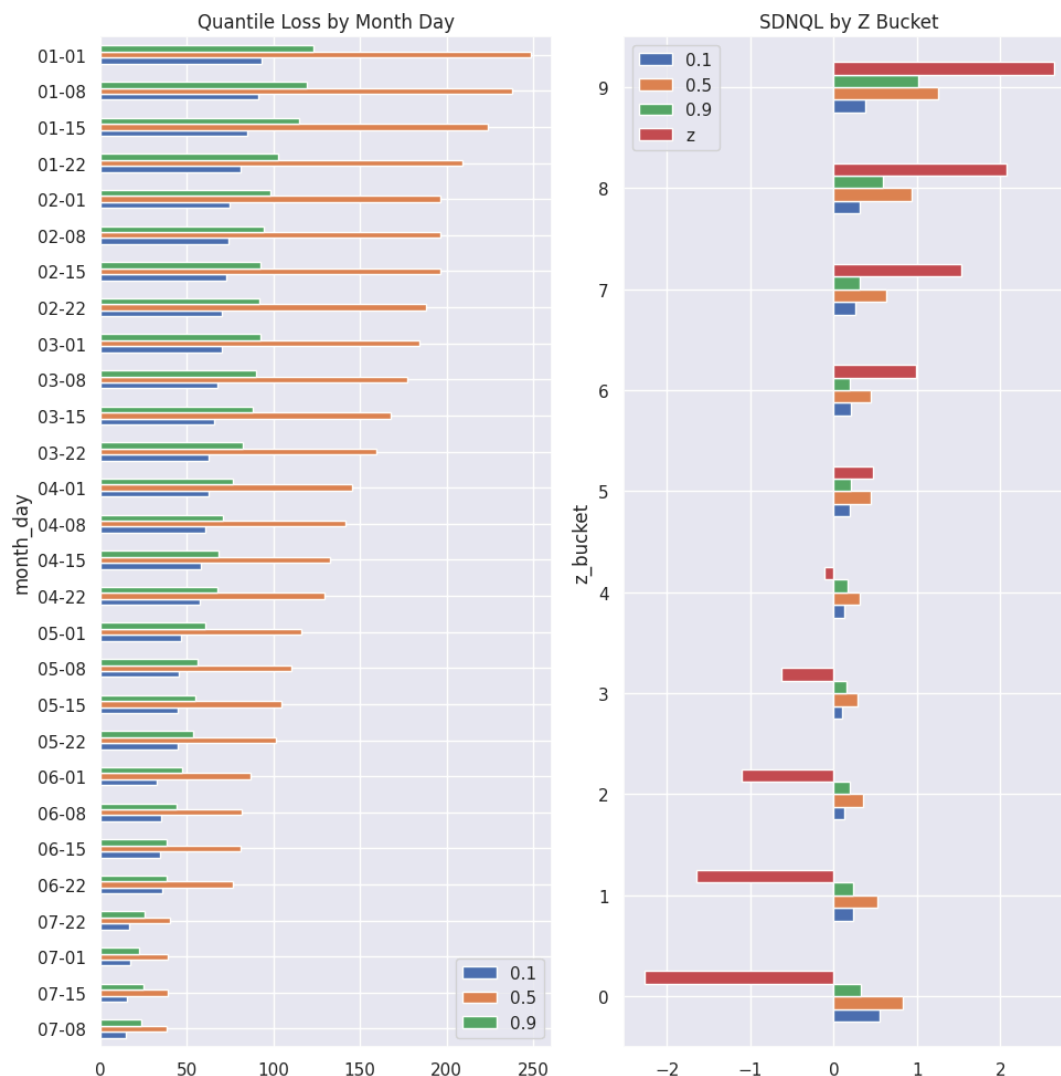


Performance by Lead Time

As expected the model performed better towards the end of the streamflow season, as more data about the streamflow season became available. The monthly model, which incorporates the previous monthly naturalized flow as it becomes available, takes on greater importance towards the end of the stream season, which is why there are jumps in performance on the first of the month when the data from the previous month becomes available.

Performance by Amount of Cumulative Streamflow Volume

To help analyze whether a particular year was low or high streamflow for a season we calculated the z-score using the historical mean and standard deviation of every stream site. We then placed streamflow values into buckets by z-score to make it easier to plot. What the plot shows is that the model performs best across all quantiles when the streamflow volume is near its median. The model struggled the most with years with high volume, resulting in a much higher loss in the median and 90th quantiles. The model also struggled a bit with extremely low streamflow volume, resulting in a much higher loss in the median and 10th quantiles.



Performance by Year

There was a great amount of variability in the performance by year. 2011 and 2012 were particularly difficult years due to high streamflow volumes for several sites. One thing we observed while experimenting with feature selection is that adding or removing a feature might lead to huge improvements for some years while simultaneously decreasing performance for other years. We think this reflects the difficulty in capturing true signal from dispersed physical measurements that make up the features.



Changes Between Stages

There were few changes made to the model between the Hindcast and Forecast stage. The main changes were to facilitate predicting in real-time. We added the UA Swann features which generated a modest improvement in the Quantile loss. We also cleaned up some code in the pre-processing, including a bug in the CDEC data pre-processing. There were far more significant changes made between the Forecast stage and the Final stage.

- Brought in the PDSI data
- Experimented with different values for the number of days to window aggregations of features before a given issue date
- Brought in the LightGBM model
- For features that consist of aggregated measurements from a number of locations/stations we implemented the solution that looks for the most correlated locations/stations in the training sets and only builds the aggregations around those locations/stations in the train/test sets
- Experimented with different ensembling approaches

Machine Specifications

The models were trained using the CPU on a MacBook Pro with a 2.6 GHz 6-Core Intel Core i7 processor and 16 GB 2667 MHz DDR4 memory. The python environment was configured to resemble the runtime available via the water-supply-forecast-rodeo-runtime Github repository.^{[22](#)}

References

1. catboost.ai
2. <https://lightgbm.readthedocs.io/en/stable/>
3. https://en.wikipedia.org/wiki/Supervised_learning
4. https://xgboost.readthedocs.io/en/latest/python/examples/quantile_regression.html
5. <https://catboost.ai/en/docs/concepts/loss-functions-regression#Quantile>
6. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.QuantileRegressor.html
7. <https://planetarycomputer.microsoft.com/dataset/cop-dem-glo-90>
8. https://en.wikipedia.org/wiki/Standard_score
9. <https://pandas.pydata.org/docs/reference/api/pandas.Period.dayofyear.html>
10. <https://www.nrcs.usda.gov/>
11. <https://water.weather.gov/ahps/rfc/rfc.php>
12. https://www.rcc-acis.org/docs_webservices.html
13. <https://www.nrcs.usda.gov/wps/portal/wcc/home/aboutUs/monitoringPrograms/automatedSnowMonitoring/>
14. <https://cdec.water.ca.gov/>
15. <https://waterservices.usgs.gov/docs/dv-service/daily-values-service-details/>
16. <https://climate.arizona.edu/snowview/csv/Download/Watersheds/>
17. <https://www.drought.gov/data-maps-tools/us-gridded-palmer-drought-severity-index-pdsi-gridmet>
18. <https://nsidc.org/data/g02158/versions/1>
19. https://www.cpc.ncep.noaa.gov/pacdir/NFORdir/HUGEdir2/explanation_fdf.html
20. <https://www.cpc.ncep.noaa.gov/data/indices/oni.ascii.txt>
21. <https://catboost.ai/en/docs/concepts/algorithm-missing-values-processing>
22. <https://github.com/drivendataorg/water-supply-forecast-rodeo-runtime>