Model documentation and write-up

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.

I work as a Senior Machine Learning Engineer for Uplight, Inc. where I build production machine learning systems that help orchestrate energy assets on the electric grid.

2. What motivated you to compete in this challenge?

Growing up and living in Denver, Colorado I have long been aware of the importance of seasonal streamflow to communities across the Western United States. I wanted to take this opportunity to learn more about the dynamics that shape streamflow from season to season and help give back to the community.

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

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. There are two model architectures underlying the solution, both based on the Catboost implementation of gradient boosting on decision trees. 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 weighted average to minimize the prediction error given by the mean quantile loss across the three quantiles.

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

See explainability report.

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

When running cross-validation (CV) with time series data it is important to isolate the feature data for both the training dataset and the prediction dataset for each CV time period to ensure that there isn't leakage of information that would artificially increase performance of the model. The first step I took

in my code to ensure feature set isolation was to build the test features and training features for each test year using an iterative process. The code below is from the `generate_train_features.py` file.

```
train_features = clean_and_merge_features(train_features, site_elevations,
elevation_cols, ['site_id'])

cv_test = pd.read_csv(DATA_DIR / 'cross_validation_labels.csv')
try:
    all_lat_lon = pd.read_csv(preprocessed_dir / 'train_lat_lon_pdsi.csv')
except:
    all_lat_lon = None
cv_dfs = []
cv_test_dfs = []
cv_splits = list(range(2004, 2024))
for cv_year in cv_splits:
    test = cv_test[cv_test['year'] == cv_year]
    test_features = generate_base_test_data(data_dir, preprocessed_dir, cv_year,
        metadata, test)
    cv_features, test_features = generate_dynamic_cv_features(data_dir,
        preprocessed_dir,
        metadata,
        train_features,
        test_features,
        cv_year,
        all_lat_lon)
    cv_features['cv_year'] = cv_year
    cv_dfs.append(cv_features)
    cv_test_dfs.append(test_features)

cv_features = pd.concat(cv_dfs)
cv_features.to_csv(preprocessed_dir / 'cv_features.csv', index=False)
logger.info(cv_features.columns)
logger.info(cv_features.shape)

cv_test_features = pd.concat(cv_test_dfs)
cv_test_features.to_csv(preprocessed_dir / 'cv_test_features.csv', index=False)
logger.info(cv_test_features.shape)
```

One of the feature strategies I implemented was to use measurements that were highly correlated with stream site streamflow volume in other years. I had to be careful to make sure that when computing the correlations for each CV iteration I removed the current year. The code below is from the `feature_utils.py` which is used to generate features using only the most correlated measurements from hold out years.

```
# Holdout feature data from the forecast year when computing mean and std for train
features to avoid data leakage
train_data = base_data[base_data['forecast_year'] != cv_year].copy(deep=True) #
Always hold out current year label
train = train[train['forecast_year'] != cv_year]

# test_data needs current year feature data
all_test_feature = base_data.sort_values([pivot_col, 'date']).copy(deep=True)
sub_test_feature = all_test_feature[all_test_feature['month'].isin(agg_months)]
sub_test_feature = sub_test_feature[sub_test_feature['forecast_year'] == cv_year]

all_test_dfs = []

train.set_index('forecast_year', inplace=True)

all_feature = train_data.sort_values([pivot_col, 'date'])
sub_feature = all_feature[all_feature['month'].isin(agg_months)]

feature_grouped = sub_feature.groupby(['month_day', pivot_col])[input_col].agg(
[np.mean, np.std]).reset_index()
feature_grouped.columns = ['month_day', pivot_col, f'{input_col}_mean',
f'{input_col}_std']
   logger.info(feature_grouped[[f'{input_col}_mean', f'{input_col}_std']].describe())
```

Finally, when running CV I subset the generated training and test feature sets to ensure no data leakage. The code below is from the `streamflow_cross_validation.py` file.

```
streamflow_features = pd.read_csv(preprocessed_dir / 'cv_features.csv')
streamflow_features['cv_year'] = streamflow_features['cv_year'].astype(str)
test_features = pd.read_csv(preprocessed_dir / 'cv_test_features.csv')
```

```
min_year = 1960
logger.info(f'Using training data as of {min_year}')
streamflow_features = streamflow_features[streamflow_features['year'] >= min_year]

# Take log of volume
streamflow_features['volume_log'] = np.log(streamflow_features['volume'])


cv_splits = list(range(2004, 2024))


all_predictions = []

for year in cv_splits:
    logger.info(f'Running cross validation for year {year}.')
    train_features = streamflow_features[(streamflow_features['cv_year'] ==
        str(year)) &
        (streamflow_features['year'] != year)]
    val_features = test_features[(test_features['year'] == year)]
    print(f'{train_features.shape}, {val_features.shape}')

    no_monthly_data = ['american_river_folsom_lake',
        'merced_river_yosemite_at_pohono_bridge']
    monthly_train_features, monthly_train_labels =
        generate_monthly_features(train_features, no_monthly_data)

    # Train Catboost models
    cat_mdl_yr_10, cat_mdl_yr_50, cat_mdl_yr_90 = train_yearly_catboost_model(
        train_features=train_features,
        feature_cols=cat_yr_feature_cols,
        preprocessed_dir=preprocessed_dir,
        cv_year=year)

    cat_mdl_mth_10, cat_mdl_mth_50, cat_mdl_mth_90 = train_monthly_catboost_model(
        train_features=monthly_train_features,
        train_labels=monthly_train_labels,
        feature_cols=cat_mth_feature_cols,
        preprocessed_dir=preprocessed_dir,
    cv_year=year)
```

```
# Train LightGBM models
lgb_mdl_yr_10, lgb_mdl_yr_50, lgb_mdl_yr_90 = train_yearly_lgb_model(
    train_features=train_features,
    feature_cols=lgb_yr_feature_cols,
    preprocessed_dir=preprocessed_dir,
    cv_year=year)

lgb_mdl_mth_10, lgb_mdl_mth_50, lgb_mdl_mth_90 = train_monthly_lgb_model(
    train_features=monthly_train_features,
    train_labels=monthly_train_labels,
    feature_cols=lgb_mth_feature_cols,
    preprocessed_dir=preprocessed_dir,
    cv_year=year)

# Predict models on held out test dataset
cat_pred_yr = predict_yearly_model(val_features,
    cat_yr_feature_cols,
    cat_mdl_yr_10,
    cat_mdl_yr_50,
    cat_mdl_yr_90,
    'cat')

score_streamflow(cat_pred_yr, 'cat_', '_yr')

cat_pred_mth = predict_monthly_model(val_features,
    cat_mth_feature_cols,
    cat_mdl_mth_10,
    cat_mdl_mth_50,
    cat_mdl_mth_90,
    no_monthly_data,
    'cat')

cat_mth_result = pd.merge(cat_pred_mth, val_features, on=['site_id',
        'issue_date'], how='left')
score_streamflow(cat_mth_result, 'cat_', '_mth')

lgb_pred_yr = predict_yearly_model(val_features,
    lgb_yr_feature_cols,
    lgb_mdl_yr_10,
    lgb_mdl_yr_50,
```

```
        lgb_mdl_yr_90,
        'lgb')
    score_streamflow(lgb_pred_yr, 'lgb_', '_yr')


    lgb_pred_mth = predict_monthly_model(val_features,
        lgb_mth_feature_cols,
        lgb_mdl_mth_10,
        lgb_mdl_mth_50,
        lgb_mdl_mth_90,
        no_monthly_data,
        'lgb')
    mth_result = pd.merge(lgb_pred_mth, val_features, on=['site_id', 'issue_date'],
        how='left')
    score_streamflow(mth_result, 'lgb_', '_mth')
```

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

- CPU (model): 2.6 GHz 6-Core Intel Core i7
- GPU (model or N/A): N/A
- Memory (GB): 16 GB 2667 MHz DDR4
- OS: Mac OS Ventura 13.6.4
- Train duration: 2 hours to generate train features, less than 10 min. to train models for each CV year.
- Inference duration: Less than 10 min. (including feature generation) for each CV year

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.)?

N/A

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

N/A

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

N/A

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

I tried an ensemble of linear models. I also tried incorporating different data sources, like SNODAS and the ONI teleconnections. I tried training a neural network using PyTorch that was getting similar performance but ran out of GPU credits in Google Colab.

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?

I would continue to explore different model architectures including deep learning while also trying to scale the problem to include more stream sites.

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

Some of the added features, like PDSI drought measurements and UASWANN SWE and Water accumulation, only reduce the quantile loss a little while adding additional complexity and runtime. These features could be removed and the solution predictions would only suffer a little bit.