Snowpack Prediction Challenge

1 Overview

Much of the high-value, productive irrigated agriculture in the Western U.S. relies on snowmelt-driven streamflow. Balancing irrigation needs with competing demands such as hydropower production, municipal supply, and environmental flows is critical in water-scarce regions. For effective water resource management, key decision variables used by water management agencies include forecasts of snowpack—measured as Snow Water Equivalent (SWE)—and the timing of snowmelt. SWE represents the amount of water that would be available if the snowpack were to melt completely. For instance, the U.S. Bureau of Reclamation relies on SWE forecasts to guide decisions on water storage and releases from reservoirs, ensuring diverse needs are met efficiently.

Recognizing the importance of this information, substantial investments have been made in monitoring systems over recent decades. The Snow Telemetry (SNOTEL) station network, which spans the Western U.S., provides daily SWE measurements along with other vital attributes such as snow depth, temperature, precipitation, and relative humidity. These data not only underpin critical water management decisions but also offer a foundation for developing models and forecasts that can better support the region's complex and competing water needs in the face of changing climatic conditions.

2 Problem Statement

We will consider submissions focused on the following problem statement:

Develop and implement a model to predict daily Snow Water Equivalent (SWE) for the winter season (December 1 to May 31) across the Western United States using spatio-temporal data.

The challenge involves predicting SWE across multiple locations in the Western United States based on the information provided. Predicting SWE for a given location and time is inherently complex due to its dependence on a range of environmental variables and spatial attributes such as elevation, latitude, longitude, and topography. Additionally, temporal dynamics play a crucial role, as the SWE season—the period from initial snow accumulation to complete snowmelt—adds another layer of complexity.

Participants will utilize the provided spatio-temporal dataset, comprising both static and dynamic features (detailed in Sections 4 and 5). The dynamic data spans the years 1991–2016 and is available for model training and testing. Your task is to predict daily SWE for the winter months of December through May in the following year (e.g., December 2012 to May 2013) using the provided feature datasets.

The ultimate goal is to develop innovative modeling approaches that enhance SWE prediction accuracy, thereby reducing risks to water systems and enabling water managers to make more informed and efficient decisions for hydrological resource management.

3 Dataset Access

The data for this challenge is stored on Kamiak

`/weka/scratch/project/hackathon/data/SnowpackPredictionChallenge/input_data/`

and also located in the following folder within AgAID Google Drive.

3.1 Folder Structure

- input_data
  - ❄ meteorological_data : this folder has csv files for individual meteorological variables for all the grids (The files are: `Modified_Output_windspeed.csv`, `Modified_Output_tmin.csv`, `Modified_Output_tmax.csv`, `Modified_Output_SRAD.csv`, `Modified_Output_SPH.csv`, `Modified_Output_Rmin.csv`, `Modified_Output_Rmax.csv`, `Modified_Output_precip.csv`)
  - ❄ swe_data : this folder has files in csv format. There are three files:
    - ■ SNOTEL location specific information. This file also contain static features like elevation and southness (File: `Station_Info.csv`).
    - ■ SWE data for model training and testing. This file will have daily data for 200 SNOTEL locations from 1991-2016 (File: `SWE_values_all.csv`).
  - ❄ addtional_test_locations : This folder contains the input data for a few test locations. Please note that no ground-truth data is provided for these locations. You are expected to apply your best-performing model or models to generate the predictions, as outlined in Section 7. The folder includes the following two files:
    - ■ All meteorological variables are combined into a single file, with each column representing an individual variable. (File: `Test_InputData_dynamicVars_2017_2019.csv`).
    - ■ Corresponding static features for the test locations. (File: `Test_InputData_staticVars_2017_2019.csv`).

4 Data Description

The dataset consists of static and dynamic features used for analysis, along with their respective sources, as detailed below:

- **Static Features:**
  - ❄ Elevation (m)
  - ❄ Latitude (lat)
  - ❄ Longitude (long)
  - ❄ Southness (in terms of slope and aspect refers to the characteristic of a land surface facing south, meaning it receives the most direct sunlight exposure due to its orientation towards the southern cardinal direction)
- **Dynamic (Daily) SNOTEL Features:**
  - ❄ Snow Water Equivalent (SWE) (response variable)
  - ❄ Precipitation (mm)
  - ❄ Minimum (tmin in °C), and maximum (`tmax` in °C) temperatures.
  - ❄ Specific humidity (`SPH`)
  - ❄ Solar Radiation (`SRAD`)
  - ❄ Maximum (`Rmax`) and Minimum Relative Humidity (`Rmin`)
  - ❄ Windspeed (mp)

These are limited variables that are provided here. You can add additional independent or derived variables from your literature review based on the premise that it will enhance the model's accuracy. Please mention any additional steps that were used for data inclusion or improving modeling in your final submission with proper citation.

5 Data Preprocessing

Three different datasets are provided to you, which need to be cleaned and processed before being used for model training and testing. The three datasets are as follows:

- Daily SNOTEL observations across locations. This data frame will have columns: date (mm/dd/yyyy), latitude, longitude, and SWE (observed).
- Daily observations of historical meteorological data (all the dynamic variables provided as separate files) across different grids. This data frame will have columns: date (mm/dd/yyyy), latitude, longitude, precip, tmin, tmax, SPH, SRAD, Rmax, Rmin, and windspeed.
- Static variables for every grid. This data frame will have columns: latitude, longitude, elevation, and southness.

5.1 Preprocessing Steps

1. **Handle Missing Values:**
   - Check for missing values in all datasets.
   - Use appropriate imputation techniques (e.g., mean or median imputation, forward/backward filling for time series).
   - Document the imputation methods used in your final report.

2. **Spatial Association of SNOTEL Locations to Grids:**
   - SNOTEL stations provide SWE observations for approximately 200 locations, while the meteorological and static feature datasets span 850 grid points.
   - Use spatial distance metrics (e.g., Euclidean distance or spatial join) to associate each SNOTEL station with its nearest grid point.
   - Filter the meteorological and static datasets to include only the grid points corresponding to the SNOTEL locations.
3. **Combine Data:**
   - For each SNOTEL location:
     - ❄ Attach the corresponding static features from the nearest grid.
   - The final combined dataset should include the following columns: `date`, `latitude`, `longitude`, `SWE`, `precip`, `tmin`, `tmax`, `SPH`, `SRAD`, `Rmax`, `Rmin`, `windspeed`, `elevation`, and `southness`.
   - This final dataset will serve as the input for model training and testing.

6 Potential metrics for performance evaluation

Below is a list of potential metrics you can compute to evaluate model performance:

- **Nash Sutcliffe Efficiency (NSE)**:

$$\text{NSE} = 1 - \frac{\sum_{i=1}^{n}(P_i - O_i)^2}{\sum_{i=1}^{n}(O_i - \overline{O})^2}$$

  - ❄ P_i: Predicted values
  - ❄ O_i: Observed values
  - ❄ $\overline{O}$: long-term mean of observed values
  - ❄ n: Number of data points

  The value of NSE ranges from -∞ to 1, with a value closer to 1 implying the best performance, and $\leq$ 0 implying that prediction is worse than using the long-term mean as the prediction.

- **Relative Bias (%)**:

$$\text{Relative Bias} = \frac{\sum_{i=1}^{n}(P_i - O_i)}{\sum_{i=1}^{n} O_i} \times 100$$

  - ❄ P_i: Predicted values
  - ❄ O_i: Observed values
  - ❄ n: Number of data points

  This metric calculates the percentage difference between predictions and observations, helping identify systematic over- or under-prediction.

- **Actual Error (Prediction - Observed)**:

$$\text{Actual Error} = P_i - O_i$$

  - ❄ P_i: Predicted value for a specific instance

❄ O_i: Observed value for the same instance

This represents the raw error for each data point, providing insights into individual prediction discrepancies.

- **Root Mean Square Error (RMSE)**:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}}$$

❄ P_i: Predicted values
❄ O_i: Observed values
❄ n: Number of data points

RMSE quantifies the average magnitude of prediction errors, with larger errors being penalized more heavily due to squaring.

Note: These are just a few example metrics commonly used to evaluate model performance. You are welcome to use other metrics or visualizations that better showcase your model's effectiveness and results.

7 Expected Outputs

Participants are required to train and test models on the provided dataset. You are free to choose any train-test split ratio. Report the performance of your model using evaluation metrics of your choice (some examples are provided in Section 6). Results can be presented as metrics or visualized through plots based on those metrics.

In addition to this, you will be given input features for 3 years for some locations (Input data is in folder `additional_test_locations`). Using these features, generate and submit a time-series plot of SWE predictions (example shown in Figure 1 below). Alongside the time-series plot, you must also provide a CSV file containing your predictions in the following format: [``Date'', ``Latitude'', ``Longitude'', ``SWE_prediction'']

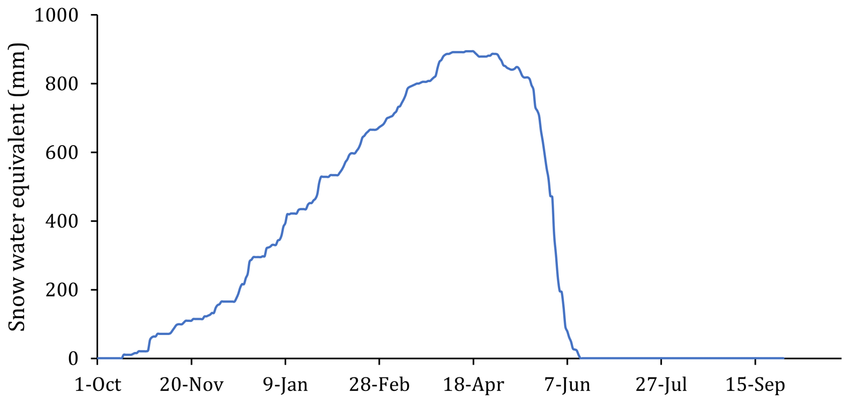

Figure 1: SWE curve for a station (Green Lake) for the water year 1984.

8 Useful packages

- Tutorials:
  - LSTM for Time Series Prediction in PyTorch
- Python tools:
  - Pytorch
  - Keras
  - Tensorflow
  - Pandas, Numpy
  - Scikit
  - Visualization: Matplotlib and Seaborn
- R:
  - Caret
  - Keras
  - Forecast
  - Tensorflow
  - Tidyverse, dplyr
  - Visualization: ggplot2

9 Running code on Kamiak

The commands to run the code should be included in the *.sh* file. An example *file.sh* file is shown below:

```
#!/bin/bash

#SBATCH --job-name=your_job_name
#SBATCH --nodes=1                      # request for nodes
#SBATCH --ntasks-per-node=1
#SBATCH --mem=64G                      # request for job memory
#SBATCH --time=12:00:00
#SBATCH --partition=partition_name
#SBATCH --gres=gpu:1                   # requesting the GPU
#SBATCH --output=log_output_%j.txt     # standard output log
#SBATCH --error=log_error_%j.txt       # standard error log

# Load necessary modules
module load anaconda3

source activate your_env_name  #specify your python environment name here
# Run your script
python /path_to/your_filename.py

# Deactivate the environment
source deactivate
```

Now run the *file.sh* with the command:

```
sbatch /path_to/file.sh
```

For more information on running code using Kamiak, please refer to: Kamiak Cheat Sheet or if you need to install libraries that are not available you can install them yourself using this guide.