PROJECT 3:
Forecasting short-term evolution of snow pack thickness with recurrent neural networks (RNNs): an application of Long-Short Term Memory (LSTM) algorithm

Applied Data Analysis and Machine Learning
FYS-STK4155 - University of Oslo

Pietro PERRONE
pietrope@uio.no

November 20, 2025

GitHub link to the project repository:
https://github.com/p-perrone/UiO_MachineLearning/tree/main/ml_project3
Link to DeepSeek chat:
https://chat.deepseek.com/share/i38u41hr9ln9ruwnxh

Abstract

Contents

1 Introduction

Snowpack plays a key role in mountain hydrology and climate systems, acting as a natural reservoir that stores precipitation in winter and releases it during melt periods, thereby controlling streamflow, water availability, and flood risk (Deng et al., 2019; Déry, 2010; DeWalle & Rango, 2008). As a seasonal component of the cryosphere, it contributes to the regulation of global surface energy balance, reflecting solar radiation thanks to its bright surface, low in absorptivity and transmissivity (Male & Granger, 1981; Roesch, 2006). Its evolution is governed by energy and mass balance processes, primarily snowfall, compaction, metamorphism, melt, and sublimation, which are strongly influenced by meteorological variables such as temperature, precipitation, solar radiation, cloud cover, humidity, and wind.

Accurate snowpack forecasting is therefore crucial for water resource management, hydropower production, avalanche forecasting, and climate impact studies. Methods range from physically based snow models that explicitly solve energy and mass balance equations, to statistical and data-driven approaches that exploit empirical relationships between snow variables and meteorological forcings (Gustafsson et al., 2001; Hou et al., 2025; Karevan & Suykens, 2020; Magnusson et al., 2020; Picard & Libois, 2024). In recent years, hybrid and machine-learning approaches, including regularized regression and sequence-based models, have gained traction for short-term snowpack prediction, particularly in contexts where dense observations are available and computational efficiency is required. Many of these studies implement Recurrent Neural Networks (RNNs), especially Long Short-Term Memory (LSTM) architectures, which are well suited for time-series modeling (Blandini et al., 2025; Wang et al., 2022).

RNNs are a class of neural architectures specifically designed to model sequential data by explicitly accounting for temporal dependencies through recurrent connections. However, standard RNNs suffer from vanishing and exploding gradient problems, which limit their ability to learn long-term dependencies. Long Short-Term Memory (LSTM) networks address these limitations by introducing a gated memory cell that regulate information flow, allowing the network to retain and update relevant informations over a given time horizon (Graves, 2012; Hewage et al., 2019; Hochreiter, S. & Schmidhuber, J., 1997). As a result, LSTMs have become a standard choice for time-series prediction in geosciences and hydrology, including snow and meteorological applications Compared to physically based models, the advantages of such methods depend on the specific objective and include: (i) robustness in capturing complex, non-linear relationships among meteorological variables without the need for explicit physical approximations, (ii) effective performance under limited observational availability, such as sparse weather stations or missing snow profile data, and (iii) relatively low computational cost, particularly once trained (Hewage et al., 2019; U. Kumar & Sharma, 2025).

Given the fairly recent diffusion of such techniques in geosciences contexts, it is crucial to properly understand their behavior and efficiency. In particular, for meteorological forecasting it is important to identify which variables play a dominant role in the model and to assess the required level of complexity, since over-parameterization may lead to unnecessary computational cost and reduced interpretability. In this study, we address these issues by performing an explicit selection of the input parameters (*i.e.*, the features) and by testing the forecasting capabilities of a sequential model based on Long Short-Term Memory (LSTM) networks, implemented using the `TensorFlow` Python module (TensorFlow Developers, 2025). Hourly meteorological data provided by the Functional Centre of the Aosta Valley Region (Western Alps, Italy) are used to predict the short-term evolution (24 hours) of snowpack at a weather station located at 1960 m a.s.l.. Such data are available on the Centre's website, and are very representative of typical mountain weather stations; for these reasons, we chose them for this analysis. The analysis is structured as follows:

(a) **Preprocessing**: loading 35 days of data (November–December 2025) from the Functional Centre, followed by scaling and gap filling (handling of missing values);

(b) **Feature selection**: splitting the dataset into training and testing subsets; fitting linear models with Ridge and Lasso (L2 and L1) regularization; tuning through bootstrapping, and evaluating coefficient stability and Mean Squared Error to assess the relative importance of each meteorological variable;

(c) **LSTM forecasting**: chronological splitting of the dataset into training and forecasting periods; construction of input sequences; evaluation of forecasting performance under different model complexities and feature sets (*i.e.* tuning of hyperparameters); forecasting and computational time estimation through bootstrapping.

2 Materials and methods

2.1 Data

The weather measurements we employed were collected at Pont Valsavarenche (https://presidi2.regione.vda.it/str_dataview_station/3070). The weather station is located at 1951 m a.s.l. on the bottom of glacial valley (45.5268 °N, 7.20093 °E in Coordinate Reference System WGS84 - EPSG:4326), it is equipped a mechanical pluviometre, an anemometer, and regular weather stations instrumentation. They are hourly based, and are summarized in Table 1; a 10 days time series is also shown in Figure 1, to give an overarching view of typical patterns followed by such variables. Each parameters corresponds to one of the features that our models will employ to

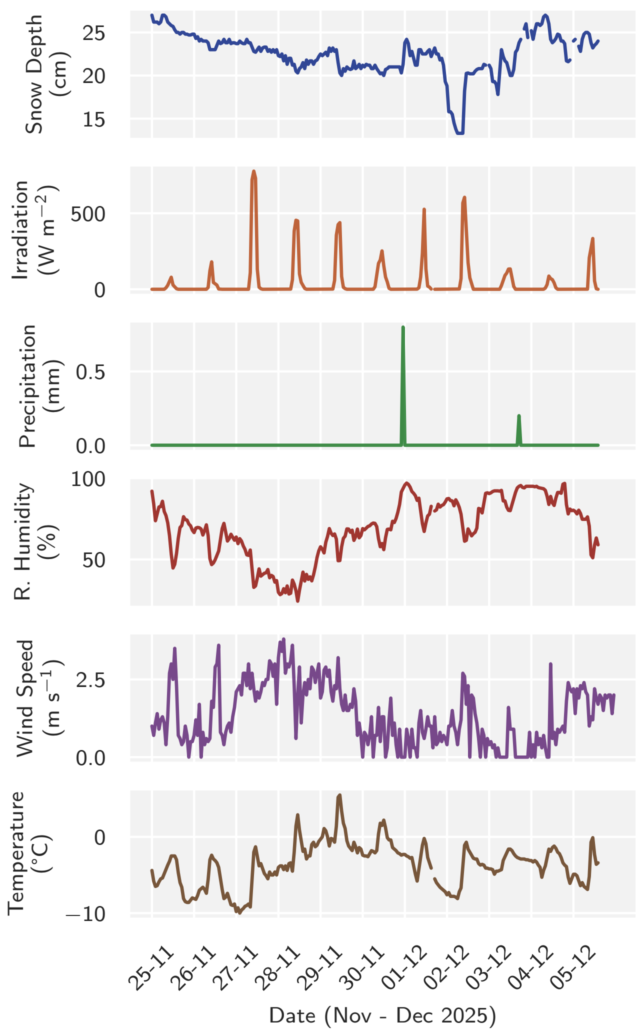

Figure 1: Time series of meteorological variables | The diurnal-nocturnal cyclic patterns can be observed in solar radiation and temperature

forecast either snow depth and temperature. They range from November 2 to December 5 2025.

2.2 Theory of central algorithms

In the following sections, we address all the different variables with specific symbology. The training design matrix $\boldsymbol{X}$ is composed of $n =\sim 720$ samples (30 days $\times$ 24 hours) and a variable amount of features p (2 to 6, depending on the model), that represent the different meteorological variables. $\boldsymbol{y}$ is the true (observed) set of values, typically 24 in total, that the models aim to reproduce, by minimizing the loss between this last and the forecasted values $\tilde{\boldsymbol{y}}$. The set of optimal parameters obtained through such minimization is denoted $\boldsymbol{\theta}$, hence $\tilde{\boldsymbol{y}} = \boldsymbol{X}\boldsymbol{\theta}$.

2.2.a Scaling the data

We applied standard scaling to all the input data. Given an a vector $\boldsymbol{x}$, all its elements are subtracted by its mean and divided by its standard deviation (Hastie et al., 2009):

$$x_i \rightarrow \frac{x_i - \overline{x}}{\sigma(\boldsymbol{x})} \tag{1}$$

Standard scaling has been chosen over other scaling methods, since our former dataset does not show any strong outliers, and it has been previously found that this technique yields highest level of accuracy in geoscientific contexts (Katipoglu et al., 2023; Nishat et al., 2025; Poushi & Chaki, n.d.).

2.2.b Feature selection: bootstrapping with Ridge and Lasso regularizations

Feature selection was performed by computing the optimal values of each parameter with Ridge and Lasso linear models. These analysis methods modify the simplest form of ordinary least squares (OLS), by adding a penalty to the cost function $\mathcal{C}(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta})$ that results in a shrinkage of optimal parameters.

Ridge applies an L2-regularization, *i.e.* adds an L2-norm $\|\boldsymbol{\theta}\|_2$, minimizing as follows:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \left[\mathcal{C}(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_2^2\right] \tag{2}$$

Lasso adds an L1-norm $\|\boldsymbol{\theta}\|_1$:

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p} \left[\mathcal{C}(\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||_1\right] \tag{3}$$

The L2-norm is given by the sum of squares of all the optimal parameters, whereas the L1-norm by the sum of the absolute values (Hastie et al., 2009). More informations are also available in report from Project 1. For the aim of this work, it is relevant to underline the different behaviors of such methods in feature selection: Ridge, by adding a squared norm, never shrinks the parameters to 0, whereas Lasso does. Ridge is therefore useful for assessing relative feature importance, but Lasso has to be used to actually select only the features to retain (Tibshirani, 1996).

All the regressors that we implemented in this work minimize the same cost function, the *Mean Squared Error* MSE:

$$\mathcal{C}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) : \mathrm{MSE}(\boldsymbol{y}, \tilde{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=0}^{n-1}(y_i - \tilde{y}_i)^2 \tag{4}$$

Report from Project 1 contains further informations on this function. It represents a standard in minimization problems, notably in geosciences contexts (Gupta et al., 2009; Lines & Treitel, 1984). Its quadratic form ensures smooth and twice-differentiable gradients, enabling efficient optimization with gradient-based methods, and leads to a convex objective function in linear models with a unique global minimum (Bishop, 2006; Hastie et al., 2009). Moreover, minimizing MSE corresponds to maximum likelihood estimation under the assumption of Gaussian-distributed errors, providing a clear probabilistic

Table 1: Meteorological variables used in the analysis.

Variable	Unit	Description
Snow depth	cm	Total snowpack depth measured at the station
Irradiation	$\mathrm{W\,m^{-2}}$	Incoming shortwave solar radiation at the surface
Precipitation	mm	Liquid-equivalent precipitation amount
Relative humidity	%	Ratio of actual to saturation water vapour content
Wind speed	$\mathrm{m\,s^{-1}}$	Horizontal wind velocity at measurement height
Air temperature	°C	Near-surface air temperature

interpretation (Wasserman, 2004).

Bootstrapping (more insights in report from Project 1) provides an empirical distribution of parameters estimates by repeatedly resampling the training set with replacement and refitting the model for a number of bootstraps B. We employed this resampling method for evaluating the loss and the stability of each feature coefficient resulting from Ridge and Lasso refitting. We chose MSE as loss function and sign–stability as metrics. Sign stability is a feature-selection criterion that assesses whether the direction (sign) of a regression coefficient is consistent under resampling. Let $\boldsymbol{\theta}$ the optimal parameters vector fitted on the full dataset, and $\boldsymbol{\theta}^{*(b)}$ the estimate from bootstrap replicate b. For each feature j, the sign–stability score is defined as

$$S_j = \frac{1}{B}\sum_{b=1}^{B}\left[\operatorname{sign}\left(\theta_j^{*(b)}\right) = \operatorname{sign}(\theta_j)\right], \tag{5}$$

which measures how consistently a feature keeps the same directional effect. Features with high stability (e.g., $S_j \approx 1$) are considered reliable, while unstable features are likely noise (Meinshausen & Bühlmann, 2010; Zou & Hastie, 2005). Once computed MSE and sign–stability for each parameter, the final selection is carried out by setting a minimal threshold based on the maximum value of the optimal parameters vector, base on U. Kumar and Sharma, 2025; each j-th parameter is selected only if its absolute value is greater than this threshold:

$$|\theta_j| \geq k|\max(\boldsymbol{\theta})| \tag{6}$$

with $k = 0.01$, denoting a significance level factor.

2.2.c RNNs and Long Short-Term Memory

Generalities

Insights about the bases of Neural Networks (NNs) are provided in report from Project 2. Particular types of NNs are Recurrent Neural Networks (RNNs). These are a class of NNs specifically designed to model sequential data, by introducing recurrent connections that allow information to persist across time steps. This architecture makes RNNs well suited for time series analysis, where temporal dependencies play a fundamental role (Elman, 1990; Goodfellow et al., 2016).

However, standard RNNs suffer from vanishing and exploding gradient problems, which limit their ability to learn long-term dependencies. Long Short-Term Memory (LSTM) networks address these issues through gated memory cells that regulate the flow of information, enabling stable learning over longer sequences and improving performance in many geoscientific and meteorological forecasting applications (Hewage et al., 2019; Hochreiter, S. & Schmidhuber, J., 1997).

LSTM functioning

At each time step t the LSTM maintains two states:

— a hidden state $\mathbf{h}^{\langle t\rangle}$;

— a cell state $\mathbf{C}^{\langle t\rangle}$

Given the input vector $\mathbf{x}^{\langle t\rangle} \in \mathbb{R}^n$, the previous hidden state $\mathbf{h}^{\langle t-1\rangle}$, and the previous cell state $\mathbf{C}^{\langle t-1\rangle}$, the LSTM update equations are defined as follows (Goodfellow et al., 2016; Raschka & Mirjalili, 2017).

Forget gate

The forget gate controls how much information from the previous cell state is retained:

$$\mathbf{f}^{\langle t\rangle} = \sigma(\boldsymbol{W}_{\mathrm{fx}}\mathbf{x}^{\langle t\rangle} + \boldsymbol{W}_{\mathrm{fh}}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_{\mathrm{f}}) \tag{7}$$

Here, $\sigma(\cdot)$ denotes the sigmoid activation function. An overview of different activation functions is provided in report from Project 2. $\boldsymbol{W}_{\mathrm{fx}}$, $\boldsymbol{W}_{\mathrm{fh}}$ and $\mathbf{b}_{\mathrm{f}}$ are the different sets of weights and biases for this gate (the same will be for the following ones).

Input gate

The input gate determines how much new information is written to the cell state. It consists of both a sigmoid function, which defines what percentage of the input will be stored in the long-term memory cell, and the tanh function, which determines what is the full memory that can be stored in the long term memory. These results can be calculated and multiplied together, their sum it is added to the cell state or stored in the long-term memory, denoted as $\mathbf{g}^{\langle t\rangle} \oplus \mathbf{i}^{\langle t\rangle}$:

$$\mathbf{i}^{\langle t\rangle} = \sigma(\boldsymbol{W}_{\mathrm{ix}}\mathbf{x}^{\langle t\rangle} + \boldsymbol{W}_{\mathrm{ih}}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_{\mathrm{i}}) \tag{8}$$

$$\mathbf{g}^{\langle t\rangle} = \tanh\left(\boldsymbol{W}_{\mathrm{gx}}\mathbf{x}^{\langle t\rangle} + \boldsymbol{W}_{\mathrm{gh}}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_{\mathrm{g}}\right) \tag{9}$$

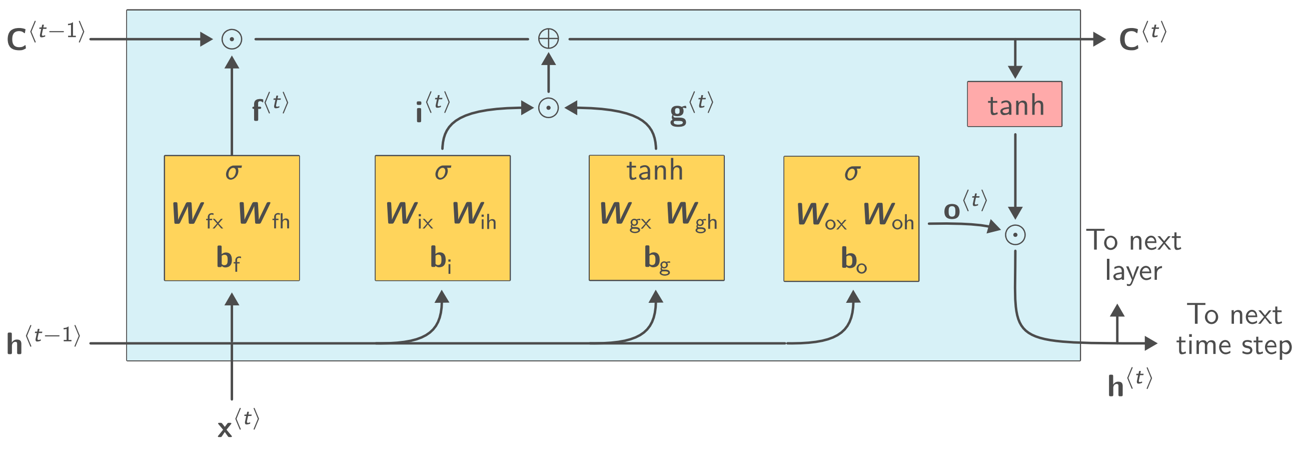
$\mathbf{C}^{\langle t-1\rangle}$
$\mathbf{C}^{\langle t\rangle}$
$\mathbf{f}^{\langle t\rangle}$
$\mathbf{i}^{\langle t\rangle}$
$\mathbf{g}^{\langle t\rangle}$
tanh
σ $\boldsymbol{W}_{\text{fx}}$ $\boldsymbol{W}_{\text{fh}}$ $\mathbf{b}_{\text{f}}$
σ $\boldsymbol{W}_{\text{ix}}$ $\boldsymbol{W}_{\text{ih}}$ $\mathbf{b}_{\text{i}}$
tanh $\boldsymbol{W}_{\text{gx}}$ $\boldsymbol{W}_{\text{gh}}$ $\mathbf{b}_{\text{g}}$
σ $\boldsymbol{W}_{\text{ox}}$ $\boldsymbol{W}_{\text{oh}}$ $\mathbf{b}_{\text{o}}$
$\mathbf{o}^{\langle t\rangle}$
To next layer
To next time step
$\mathbf{h}^{\langle t-1\rangle}$
$\mathbf{h}^{\langle t\rangle}$
$\mathbf{x}^{\langle t\rangle}$

Figure 2: Long Short-Term Memory unit | LSTM is characterized by a hidden state and a cell state. At each time step t, a forget gate controls hw much of the previous information is retained, and an input gate determines how much new information is written to the cell state. This mechanism allows keeping a long term memory throughout the network, resulting suitable for handling long time series. Modified after Raschka and Mirjalili, 2017

Candidate cell state

A candidate update for the cell state is computed as:

$$\tilde{\mathbf{C}}^{\langle t\rangle} = \tanh\left(\boldsymbol{W}_C\mathbf{x}^{\langle t\rangle} + \mathbf{b}_C\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_C\right). \tag{10}$$

Cell state update

The new cell state combines retained memory and newly acquired information:

$$\mathbf{C}^{\langle t\rangle} = \mathbf{f}^{\langle t\rangle} \odot \mathbf{C}^{\langle t-1\rangle} + \mathbf{i}^{\langle t\rangle} \odot \tilde{\mathbf{C}}^{\langle t-1\rangle} \tag{11}$$

where $\odot$ denotes element-wise multiplication.

Output gate

The output gate regulates how much of the cell state is exposed to the hidden state:

$$\mathbf{o}^{\langle t\rangle} = \sigma\left(\boldsymbol{W}_{\text{ox}}\mathbf{x}^{\langle t\rangle} + \boldsymbol{W}_{\text{ox}}\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_{\text{o}}\right). \tag{12}$$

Hidden state update

The hidden state is finally updated as:

$$\mathbf{h}^{\langle t\rangle} = \mathbf{o}^{\langle t\rangle} \odot \tanh\left(\mathbf{C}^{\langle t\rangle}\right). \tag{13}$$

Vectorized formulation

For computational efficiency, the gate computations are often implemented in a single matrix operation:

$$\begin{bmatrix} \mathbf{f}^{\langle t\rangle} \\ \mathbf{i}^{\langle t\rangle} \\ \tilde{\mathbf{C}}^{\langle t\rangle} \\ \mathbf{o}^{\langle t\rangle} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \tanh \\ \sigma \end{bmatrix} \left(\boldsymbol{W}\begin{bmatrix} \mathbf{x}^{\langle t\rangle} \\ \mathbf{h}^{\langle t-1\rangle} \end{bmatrix} + \mathbf{b}\right). \tag{14}$$

A schematic view of LSTM functioning is shown in Figure 2,

2.2.d *Evaluation of the model: Root Mean Squared Error (RMSE)*

The performance of the model is evaluated by computing the discrepancy between forecasted and observed values of the target variable, after inverse scaling has restored physical units. Given observed values $\mathbf{y} = (y_1, \ldots, y_n)$ and corresponding forecasts $\tilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{\tilde{y}}_n)$, the Root Mean Squared Error (**aaaaaaaaaaaaaaaaaa**) is defined as:

$$\text{RMSE}(\tilde{\mathbf{y}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\tilde{y}_i - y_i)^2}. \tag{15}$$

For snow depth retrieval, particularly, a further information about model quality is given by the ratio of RMSE to total rime-averaged snowpack thickness, both expressed in cm, *i.e.* the Normalized Root Mean Squared Error (Moriasi et al., 2007):

$$R = \frac{\text{RMSE}}{\bar{H}} \times 100 \quad (\%), \tag{16}$$

with

$$\bar{H} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{17}$$

2.3 Implementation

2.3.a *Preprocessing*

The data from Pont weather station were loaded in in a Python 3.12 environment and organized in a dataframe using `glob` and `pandas` modules (Pandas Development Team, 2024; Python Software Foundation, 2024), with time steps as indexes and variables as columns. This part was performed in two steps, as we decided to include in the analysis wind speed only in a second moment, taking the occasion to enlarge the dataset timespan. The loaded dataframe contained NaNs. Some rows were entirely NaNs (missing values for every variable) –

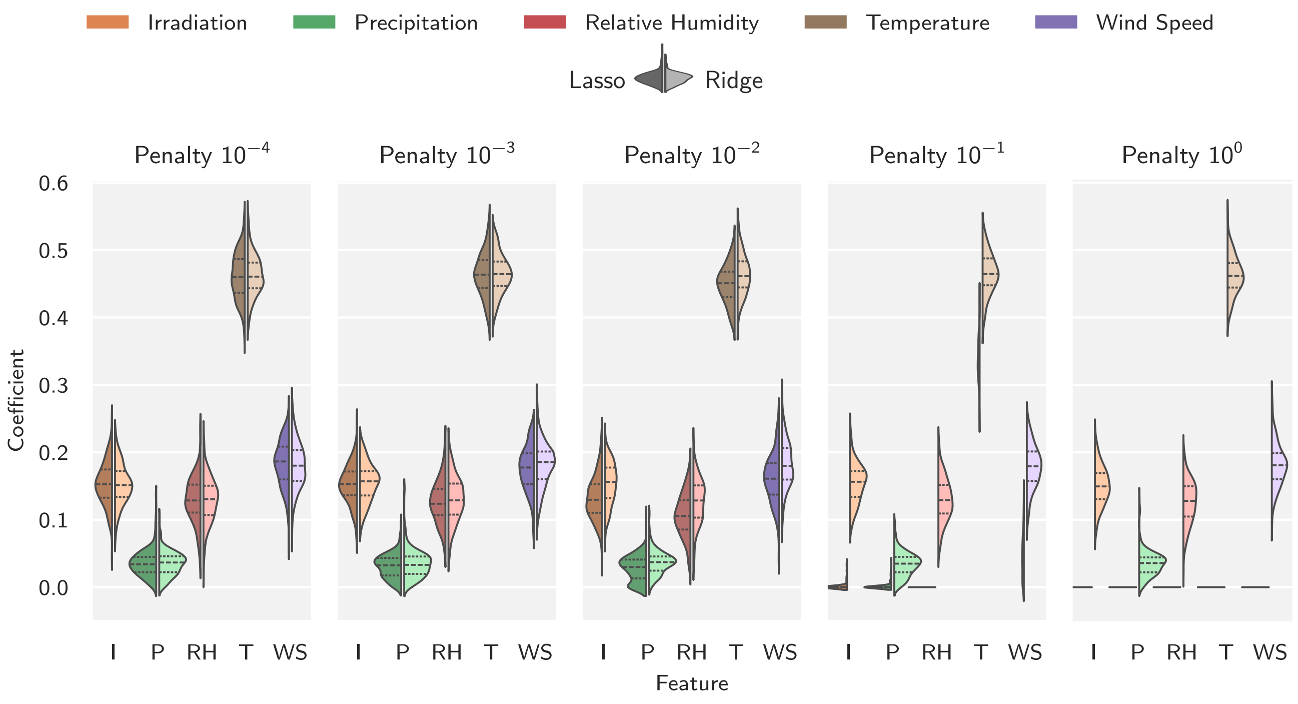

Figure 3: Coefficients distributions from Ridge/Lasso feature selection | Bootstrapping for different values of penalties ranging from 10^{-4} to 1; each distribution contains 200 coefficients. Violins are splitted – the right half corresponds to Ridge fits, the left to Lasso. The three hashed lines in each violins denotes the first interquartile, the median and the third interquartile respectively. In general, Ridge yields slightly higher values, and Lasso shrinks all the coefficients to 0 as applied penalties are in the order of magnitude of 10^{-1} or higher.

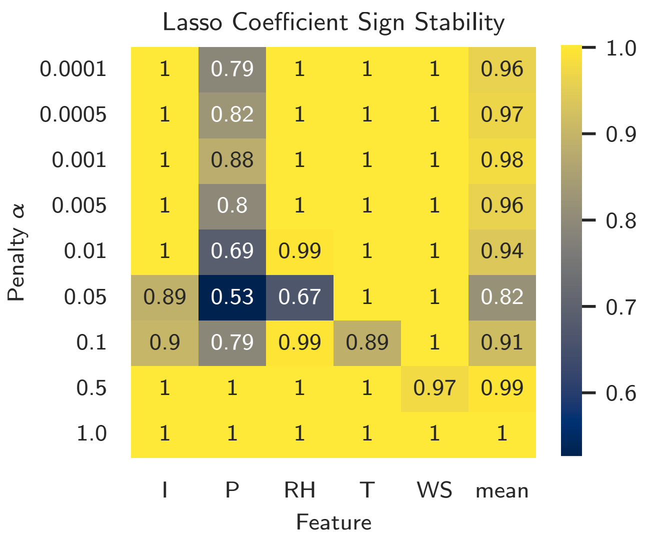

Figure 4: Sign–stability of Lasso fits over 200 bootstraps as function of penalty | Values between 0.9 and 1 indicate consistent sign in the same coefficient distribution. In general, this result is achieved for $\alpha \geq e-2$.

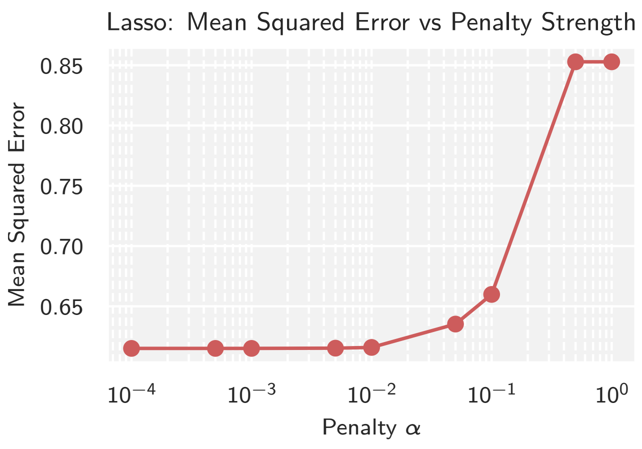

Figure 5: MSE from Lasso with different regularizations | The error between observed and modeled snow depths exponentially increases for $\alpha \geq 1$.

we directly excluded these rows from further computations. For single or small gaps, we applied forward and backward filling, that consist in replacing missing values at time step (index) `t` with the value recorded at `t-1` or `t+1`. This gap filling method was confirmed to be one of the most efficient when handling small gaps meteorological data, since it is simple, stable and accurate based on RMSE results (Siabi et al., 2022).

Most of the preprocessing was performed using `scikit-learn` utilities (M. Kumar et al., 2023; scikit-learn developers, 2024). Column-wise scaling was carried out with `scikit-learn` `StandardScaler` class to ensure numerical stability, as previously explained.

Train-test splitting was handled separately and differently for feature selection ad LSTM modeling, hence it is not included in this part of the workflow.

2.3.b *Feature selection*

Since snow depth is our focus, feature selection was carried out to investigate which other parameters had the strongest influence on it.

In the order:

1. the dataset was split with `scikit-learn` `train_test_split` function, setting all the steps before December 2 as training set;
2. for different values of penalties α ranging from 10^{-5} to 1:

   – Ridge ad Lasso fits were computed with bootstrapping (200 bootstraps), resulting in a distribution of coefficients for each meteorological variable;

   – the sign-stability and MSE were computed from these distributions;
3. the penalty value that yielded the best combination of MSE and sign–stability was used to select the essential features for snow depth forecasting, again bootstrapping Lasso fit over 400 iterations and filtering with the threshold criterion (Equation (6)).

This tuning workflow was carried out by creating custom functions, one for the metrics (`stability_mse()`) and one for the actual feature selection (`feature_selection()`). Note that, for feature selection, the MSE is computed using snow depth values modeled over the same time period in which the meteorological observations are available; the forecasting task is addressed separately in the following section.

2.3.c Forecasting

Splitting of the dataset in a test and training sets was done manually by setting a split time (December 2 at 00:00).

The original hourly meteorological time series were first transformed into supervised learning samples through a sliding-window sequencing approach. The `create_sequence` custom function transforms raw time series data of shape `(n, p)` , with `n` the number of samples and `p` the number of features, into input-output pairs suitable for sequence models such as LSTMs. For a given sequence length, in our case typically 48 hours, the function extracts overlapping sequences of past observations as model inputs, and for a forecast horizon (24 hours), it extracts the corresponding future values of the target variable as outputs. If the input is a `pandas` DataFrame, the specified feature columns are stacked alongside the target column, with the target always placed last to simplify slicing. The function iterates over the dataset, generating arrays of shape `(n, sequence_length, p)` and Y of shape (n, forecast_horizon).

This setup allows the LSTM to learn temporal dependencies from past features to predict multiple future steps of the target variable, effectively framing time series forecasting as a supervised learning problem.

The implementation of LSTM was straight-forward with `tensorflow` (https://www.tensorflow.org/), a machine learning-oriented Python ecosystem. Its library `keras` is particularly suitable for sequential models such as RNNs (https://keras.io/). We structured our code using `keras.models.Sequential()` as base model, and `keras.layers.LSTM()` and `keras.layers.Dense()` as layers. The workflow we followed to build this model is largely inspired to this notebbok (TensorFlow Developers, 2023, 2025). A typical implementation in Python would be the following;

```
model = Sequential([
  LSTM(units=32,
    input_shape=(timesteps, n_features)
    ),
  Dense(horizon)
  ])
model.compile(
    optimizer='adam',
    loss='mse',
)
model.fit(X_train, y_train,
    epochs=50,
    batch_size=16,)
```

The LSTM layer receives input sequences of shape `(timesteps, n_features)` and learns temporal dependencies across the input window, compressing the sequence information into a latent representation of size units. The subsequent Dense layer maps this representation to the forecast horizon, producing a multi-step prediction of the target variable. The model is trained using the Adam optimizer, since it is the most efficient according to report from Project 2, and by minimizing the MSE loss. Training is performed over multiple epochs with mini-batches, allowing the network to iteratively adjust its parameters to capture the temporal dynamics of the data.

We organized forecasting by modeling both snow depth and temperature in the future 24 hours (*i.e.*, `horizon`). In fact, even is our focus is primarily snow pack, temperature patterns have a more 'predictable' behavior as more influenced from diurnal cycles of solar radiation; it can therefore be a reference benchmark for evaluating the network, whereas snow pack is more experimentally complicated to forecast (Blandini et al., 2025; DeWalle & Rango, 2008; Lehning et al., 2002). Since features were not selected for temperature, this was fitted using all the variables. Snow depth was fitted both with the full features set (snow depth itself, irradiation, precipitation, relative humidity, wind speed and temperature) *and* with the features previously selected only. We end up with we three sets to model, in the end.

Model tuning was carried out by gradually seeking MSE minima in simulation results obtained with different ranges of selected hyperparameters. For the three target sets, we evaluated MSEs between the observed and forecasted values by exploring the dependence of fit quality on:

(1) the choice of activation function for the `LSTM` layer and the number of epochs;

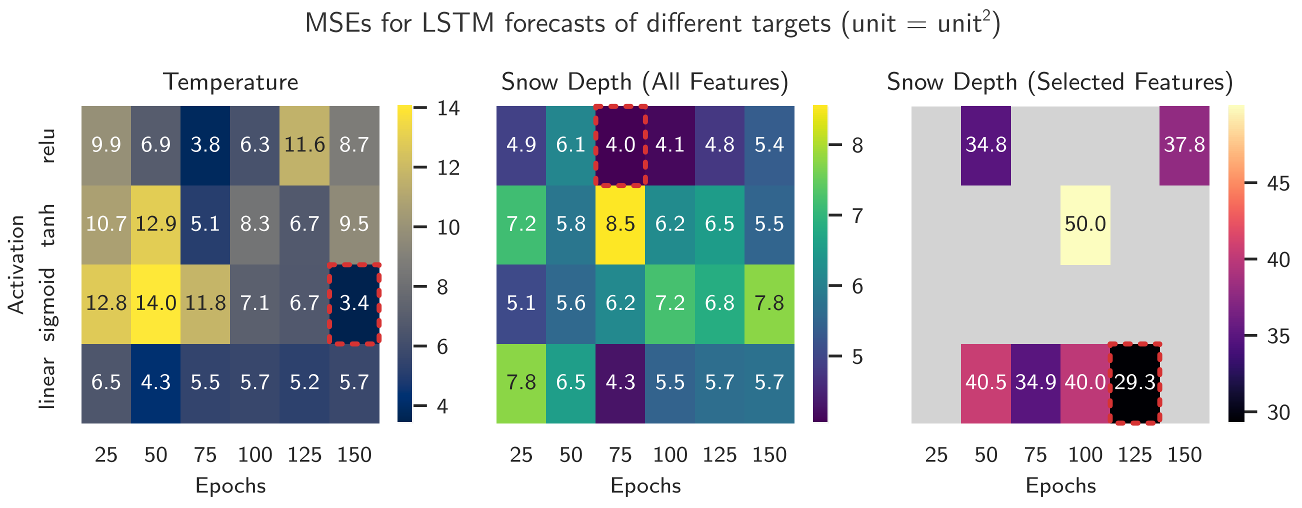

Figure 6: MSE of LSTM forecasts for different activation functions and number of epochs | The bars refer to MSE values expressed in squared predictor unit. Values above 50 are displayed in light gray. The best (minimum) MSE is highlighted in red dashed line.

(2) the number of units in the `LSTM` layer and the use of a bias initializer, employing the optimal activation and epochs found in (1).

The bias initializer can be passed as an argument to the `Dense` layer, for instance by using the mean of the last 12 hours of observations in the pre-forecasting period. Potentially, this may improve the fit by biasing the forecast toward a more realistic value than the default random bias.

Finally, we performed a forecast with the optimal combination of the four hyperparameters (activation function, epochs, units, and bias initializer) and computed RMSE and NRMSE between the modeled and observed series.

Table 2: Lasso coefficients for snow depth prediction.

Predictor (unit)	Coef. (cm unit^{-1})	Std. error (cm unit^{-1})
Irradiation (W m^{-2})	0	5.9×10^{-3}
Precipitation (mm h^{-1})	0	7.0×10^{-3}
Relative humidity (%)	0	1.1×10^{-3}
Wind speed (m s^{-1})	4.2×10^{-2}	3.0×10^{-2}
Temperature (°C)	-3.3×10^{-1}	2.7×10^{-2}

2.4 Use of Artificial Intelligence

ChatGPT service from Open AI was used both in browser version and VSCode AI Chat. It was mainly employed to fix, debug or improve custom functions and fairly complex plots (such as Figure 3). Whether this tool was used, a comment line in the code specifies it, both in GitHub-available `functions.py` file and in the analysis notebook. The employed online chat can be visited at CHATGPTCHAT. For bibliography, Google Scholar Labs provided a fast and efficient way to find studies comparable to this work.

3 Results and discussions

3.1 Feature selection

The results of the bootstrapped Ridge and Lasso fits are shown in Figure 3. Ridge generally yields slightly larger coefficient values than Lasso and, as expected, does not shrink any of them to zero, whereas Lasso drives several coefficients to zero for penalties $\alpha \geq 10^{-1}$. Temperature exhibits by far the strongest effect, with mean coefficients 3–10 times larger than those of the other predictors in both methods. This outcome is consistent with previous studies that relied solely on temperature to estimate snowpack or ice dynamics, for example Degree-day approaches used to model melt rates (Braithwaite, 1995; Hachem et al., 2009; Hock, 1999). Sign–stability and MSEs computed from these fits suggest selecting features using $\alpha = 0.1$, as it represents an appropriate tradeoff between penalization and accuracy, with a sign–stability of 0.91 (Figure 4) and an MSE of 0.65 (Figure 5). Hence, with $\alpha = 0.1$, Lasso regression yields the coefficients in Table 2, indicating that temperature and wind speed should be selected as the primary predictors. This result is partially consistent with existing literature: temperature remains the major regulator of snowpack evolution. Although wind can be considered an informative predictor, its influence is generally weaker than that of precipitation: correlation coefficients between snow pack thickness and snowfall intensity are usually ~ 2 times larger than snow depth/wind speed ones. (Dadic et al., 2010; Durand et al., 1993; Ma et al., 2023). The relatively small weight assigned to irradiation is likely related to the topographic context of the Pont weather station: the narrow, flat val-

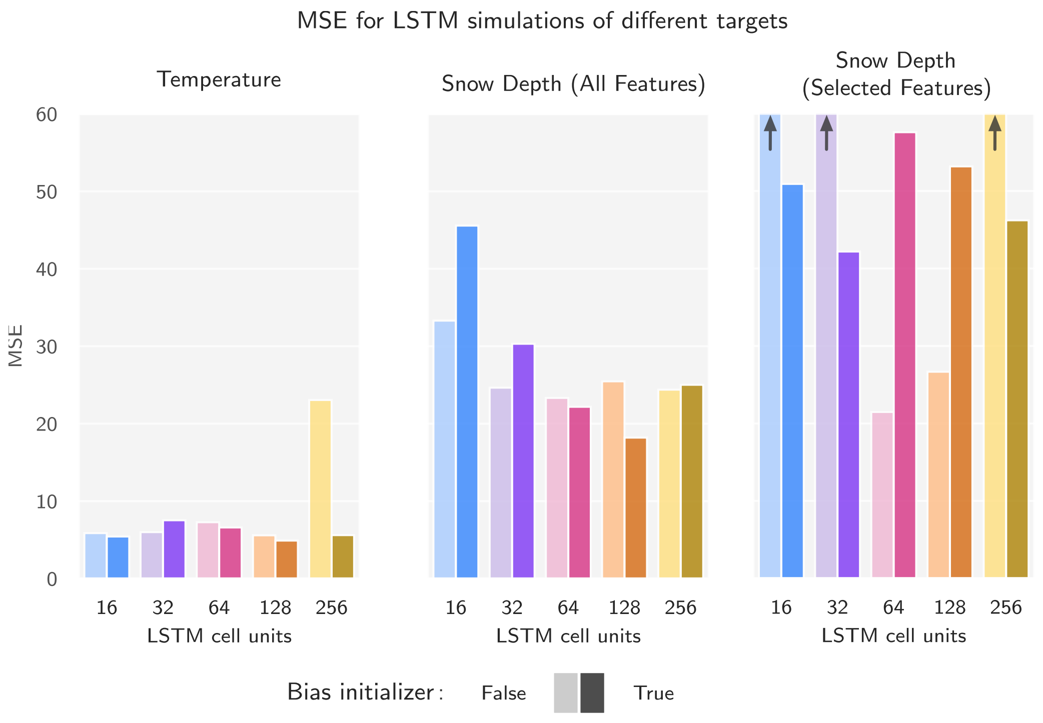

Figure 7: MSE of LSTM forecasts for different units amount and with/without bias initialization | For each units range, left and pastel bins refer to random bias initialization (`None`); right, saturated bins refer to bias initialization using the mean of last 12 hours of training series (before the start of forecasting period). *y*-axis is truncated at MSE = 60, longer bins are displayed with an upward arrow. Optimal combinations of activation function and epochs were set after the first tuning phase (Figure 6).

ley bottom exhibits limited contrasts in daytime irradiation, and the amount of solar radiation received at the surface in winter is reduced.

3.2 LSTM forecasting

The results of LSTM model tuning are showed in Figures 6 and 7. In general, all the MSEs are fairly high, always included in the range $3.4 - 10^3$. This is a first indicator of model incapacity to correctly reproduce patterns in the future, especially when a little amount of predictor is set. We would rather expect MSE below 1, to potentially compare our model architecture with the ones of similar, but more effective studies, such as Blandini et al., 2025; Wang et al., 2022. In fact, trying to forecast snow depth using temperature and wind speed solely leads to poor quality predictions. When all the predictors are included in the model, the 24 hours-forecasted snow depth approximate the finally observed values much better. LSTM still yields best results with temperature forecasts, with the lowest recorded MSEs, both for the first tuning phase (activation function – epochs dependency) and the second (LSTM units – bias initializer dependency).

From Figure 6, we can assess the optimal combination of activation function and epochs range: (Sigmoid, 150 epochs) for temperature prediction, (ReLu, 75 epochs) for snow depth with all predictors, (Linear, 125) for snow depth with selected features. Since trends in MSEs are barely visible (*i.e.* , no strong dependencies on epochs range), it also is difficult to assess whether these combinations are effective or just random effects-related. Still, we set them as constant hyperparameters for the following part.

Figure 7 is

References

Bishop, C. M. (2006). *Pattern recognition and machine learning (information science and statistics)*. Springer-Verlag.

Blandini, G., Avanzi, F., Campo, L., Gabellani, S., Aalstad, K., Girotto, M., Yamaguchi, S., Hirashima, H., & Ferraris, L. (2025). Learning to filter: Snow data assimilation using a Long Short-Term Memory network. *The Cryosphere*, *19*(10), 4759–4783. https://doi.org/10.5194/tc-19-4759-2025

Braithwaite, R. J. (1995). Positive degree-day factors for ablation on the Greenland ice sheet studied by energy-balance modelling (2017/01/20). *Journal of Glaciology*, *41*(137), 153–160. https://doi.org/10.3189/S0022143000017846

Dadic, R., Mott, R., Lehning, M., & Burlando, P. (2010). Wind influence on snow depth distribution and accumulation over glaciers. *Journal of Geophysical Research: Earth Surface*, *115*(F1).

Deng, H., Chen, Y., & Li, Y. (2019). Glacier and snow variations and their impacts on regional water resources in mountains. *Journal of Geographical Sciences*, *29*(1), 84–100.

Déry, S. J. (2010). Snow and climate: Physical processes, surface energy exchange and modeling. *Polar Research*, *29*(3), 461–462. https://doi.org/10.1111/j.1751-8369.2010.00181.x

DeWalle, D. R., & Rango, A. (2008). *Principles of Snow Hydrology*. Cambridge University Press. https://doi.org/10.1017/CBO9780511535673

Durand, Y., Brun, E., Merindol, L., Guyomarc'h, G., Lesaffre, B., & Martin, E. (1993). A meteorological estimation of relevant parameters for snow models (2017/01/20). *Annals of Glaciology*, *18*, 65–71. https://doi.org/10.3189/S0260305500011277

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning - Chapter 10 "Sequence Modeling: Recurrent and Recursive Nets". MIT Press. https://www.deeplearningbook.org/contents/rnn.html

Graves, A. (2012). Supervised Sequence Labelling. In A. Graves (Ed.), *Supervised Sequence Labelling with Recurrent Neural Networks* (pp. 5–13). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-24797-2_2

Gupta, H. V., Kling, H., Yilmaz, K. K., & Martinez, G. F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. *Journal of hydrology*, *377*(1–2), 80–91.

Gustafsson, D., Stähli, M., & Jansson, P.-E. (2001). The surface energy balance of a snow cover: Comparing measurements to two differentsimulation models. *Theoretical and Applied Climatology*, *70*(1), 81–96. https://doi.org/10.1007/s007040170007

Hachem, S., Allard, M., & Duguay, C. (2009). Using the MODIS land surface temperature product for mapping permafrost: An application to northern québec and labrador, canada. *Permafrost and Periglacial Processes*, *20*(3), 223–233. https://doi.org/10.1002/ppp.672

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer.

Hewage, P., Behera, A., Trovati, M., & Pereira, E. (2019). Long-short term memory for an effective short-term weather forecasting model using surface weather data, 382–390.

Hochreiter, S. & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hock, R. (1999). A distributed temperature-index ice- and snowmelt model including potential direct solar radiation (2017/01/20). *Journal of Glaciology*, *45*(149), 101–111. https://doi.org/10.3189/S0022143000003087

Hou, Y., Yang, Y., Han, J., Woods, R., & Yan, Z. (2025). Impacts of snow changes on hydropower potential under a changing climate. *Journal of Hydrology*, 133747.

Karevan, Z., & Suykens, J. A. (2020). Transductive LSTM for time-series prediction: An application to weather forecasting. *Neural Networks*, *125*, 1–9. https://doi.org/10.1016/j.neunet.2019.12.030

Katipoglu, O. M., Pekin, M. A., & Akil, S. (2023). The Impact of Preprocessing Approaches on Neural Network Performance: A Case Study on Evaporation in Adana, a Mediterranean Climate. *Indonesian Journal of Earth Sciences*, *3*(2), A821. https://doi.org/10.52562/injoes.2023.821

Kumar, M., Telenczuk, M., & Hug, N. (2023). Feature Selection — Model-based and sequential feature selection. *scikit learn Developers*. https://scikit-learn.org/stable/auto_examples/feature_selection/plot_select_from_model_diabetes.html

Kumar, U., & Sharma, N. (2025). Comparative analysis of various state-of-the-art models used for weather prediction. *Available at SSRN 5193462*.

Lehning, M., Bartelt, P., Brown, B., & Fierz, C. (2002). A physical SNOWPACK model for the Swiss avalanche warning: Part III: Meteorological forcing, thin layer formation and evaluation. *Cold Regions Science and Technology*, *35*(3), 169–184. https://doi.org/10.1016/S0165-232X(02)00072-1

Lines, L. R., & Treitel, S. (1984). A review of least-squares inversion and its application to geophysical problems. *Geophysical prospecting*, *32*(2), 159–186.

Ma, H., Zhang, G., Mao, R., Su, B., Liu, W., & Shi, P. (2023). Snow depth variability across the Qinghai Plateau and its influencing factors during 1980–2018. *International Journal of Climatology*, *43*(2), 1094–1111. https://doi.org/10.1002/joc.7883

Magnusson, J., Nævdal, G., Matt, F., Burkhart, J. F., & Winstral, A. (2020). Improving hydropower inflow forecasts by assimilating snow data. *Hydrology Research*, *51*(2), 226–237.

Male, D. H., & Granger, R. J. (1981). Snow surface energy exchange. *Water Resources Research*, *17*(3), 609–627.

Meinshausen, N., & Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*(4), 417–473.

Moriasi, D., N., Arnold, J., G., Van Liew, M., W., Bingner, R., L., Harmel, R., D., & Veith, T., K. (2007). Model Evaluation Guidelines for Systematic Quantification of Accuracy in Watershed

Simulations. *Transactions of the ASABE*, *50*(3), 885–900. https://doi.org/10.13031/2013.23153

Nishat, M. M., Aarvold, M. O., Hartvig, W. J., & Olsson, N. O. (2025). Investigating the Applicability of Long Short-Term Memory (LSTM) Algorithm in Project Decision-Making: A Case Study in ML-Driven Forecasting, 26–44.

Pandas Development Team. (2024). *Pandas Documentation*. manual. https://pandas.pydata.org/docs/

Picard, G., & Libois, Q. (2024). Simulation of snow albedo and solar irradiance profile with the Two-streAm Radiative TransfEr in Snow (TARTES) v2.0 model. *Geosci. Model Dev.*, *17*(24), 8927–8953. https://doi.org/10.5194/gmd-17-8927-2024

Poushi, M. R., & Chaki, S. (n.d.). Exploring A Machine Learning Model in Seasonal Prediction of Temperature over Dhaka.

Python Software Foundation. (2024). *Glob — unix style pathname pattern expansion*. manual. https://docs.python.org/3/library/glob.html

Raschka, S., & Mirjalili, V. (2017). *Python machine learning* (2nd ed.). Packt Publishing.

Roesch, A. (2006). Evaluation of surface albedo and snow cover in AR4 coupled climate models. *Journal of Geophysical Research: Atmospheres*, *111*(D15). https://doi.org/10.1029/2005JD006473

scikit-learn developers. (2024). *Scikit-learn Documentation*. manual. https://scikit-learn.org/stable/

Siabi, N., Sanaeinejad, S. H., & Ghahraman, B. (2022). Effective method for filling gaps in time series of environmental remote sensing data: An example on evapotranspiration and land surface temperature images. *Computers and Electronics in Agriculture*, *193*, 106619. https://doi.org/10.1016/j.compag.2021.106619

TensorFlow Developers. (2023). *Time series forecasting*. Retrieved on March 8, 2025, from https://www.tensorflow.org/tutorials/structured_data/time_series

TensorFlow Developers. (2025). *Sequential model guide*. https://www.tensorflow.org/guide/keras/sequential_model

Tibshirani, R. (1996). Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288. https://doi.org/10.1111/j.2517-6161.1996.tb02080.x

Wang, Y.-H., Gupta, H. V., Zeng, X., & Niu, G.-Y. (2022). Exploring the potential of long short-term memory networks for improving understanding of continental-and regional-scale snowpack dynamics. *Water Resources Research*, *58*(3), e2021WR031033.

Wasserman, L. (2004). Linear and Logistic Regression. In L. Wasserman (Ed.), *All of Statistics: A Concise Course in Statistical Inference* (pp. 209–229). Springer New York. https://doi.org/10.1007/978-0-387-21736-9_13

Zou, H., & Hastie, T. (2005). Regularization and Variable Selection Via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, *67*(2), 301–320. https://doi.org/10.1111/j.1467-9868.2005.00503.x