Predicting Vermont Weather Data from Local Vermont Sensors

Alaina Birney, Vamsi Dondeti, Maya Griffith
CSYS 5870 Spring 2024 - Project Paper

Abstract

In this study, we investigate the viability of using various machine learning learning models to predict future snow depth in Vermont from data collected from meteorological sensors located at various geographical sites dispersed throughout the state. Additionally, we perform exploratory data analysis on the aforementioned data to help visualize and understand trends in weather factors. One of our main goals was to investigate whether we can achieve better performance when training models on aggregated data from multiple locations in order to make predictions for multiple locations or if more optimal performance is achieved by training models on data from individual locations in order to make predictions regarding snow depth for individual locations. Possessing the ability to formulate predictions about future snow depth with a high level of accuracy is increasingly important as our winters exhibit greater differences year-to-year in the face of climate change, it is our hope that this research can provide foundation for further exploration into this necessary endeavor. We were able to achieve excellent performance in predicting snow depth for a single location after training a Long Short-Term Memory (LSTM) neural network and a random forest on data from a single location, suggesting that it is possible to make accurate predictions about future snow depth using machine learning models and that optimal performance can be achieved through making models for specific locations, rather than aggregating data from multiple locations and predicting future snow depth for a variety of locations using a single model.

1 Introduction

In our project, we analyzed a dataset containing measurements from different weather sensors throughout Vermont and subsequently created various machine learning models to answer the following research questions:

- What are the trends in temperature and snow depth?
- Is there any correlation between temperature and snow depth?
- How can we predict snow depth for future dates in Vermont?
- Is it best to predict snow depth for one location at a time, or can we achieve similar results when predicting snow depth for multiple locations at once?

Specifically, we created two long short-term memory neural networks (LSTMs) and two Random Forests (RFs) for this purpose. For each model (LSTM and RF), one version was made to predict snow depth for one location at a time and a separate version was made to predict snow depth for all locations at one time.

This project should interest residents of Vermont who want to better understand current and future weather trends in their state. For example, many Vermont residents enjoy snow-related activities such as skiing and snowboarding or rely on income from snow-sports related tourism. Having accurate predictions of future snow depth can aid in recreational or financial planning for these reasons. Additionally, this project should interest anyone who has a desire to better understand future changes that can occur in the face of climate change; snow depth and subsequent melting can influence a variety of phenomena, including water resources and flooding.

Existing research has been performed to understand the viability of machine learning models as tools for predicting weather based on meteorological data, which was of interest to us as we utilized machine learning to predict snow depth

based on meteorological data. In Machine Learning in Weather Prediction and Climate Analyses- Applications and Perspectives [1], Bochenek and Ustrnul analyzed 500 scientific articles that each described using machine learning methods for climate and weather prediction. Many of the meteorological fields that we are interested in were also considered for this study: wind, precipitation, temperature and pressure, for example. Additionally, various machine learning models were considered: Deep Learning, Random Forest (RF), Artificial Neural Networks, Support Vector Machines, and XGBoost. The study concluded that machine learning models such as these can be successfully used to analyze meteorological data and make determinations regarding weather. This is particularly relevant because we utilized RF and Artificial Neural Networks (specifically, Long Short-Term Memory (LSTM) Artificial Neural Networks) for predicting future snow depth in our project.

In Predictive Models for Wind Speed Using Artificial Intelligence and Copula [2], Ehsan compares the performance of twelve different machine learning models in predicting wind speed from meteorological parameters. Model performance was evaluated on mean absolute error, mean squared error, median absolute error, and R2 score. All algorithms that were considered, with the exception of ridge regression, lasso regression, and convolutional neural networks (CNN), were found to have an R2 value greater than .9 when used to predict wind speed. This signifies that most models had a relatively high level of accuracy in predicting wind speed from meteorological data. Recurrent neural networks (RNN) were found to have the highest R2 score (.978) of all models considered. Random forest (RF) was also found to be a relatively high performer with an R2 score of .962. The high R2 scores observed for these models in Ehsan's study made our choice of using random forest and neural networks more justified; their robust performance in predicting wind speed from meteorological data suggest they could similarly perform well in forecasting related meteorological phenomena such as snow depth. Ehsan's study also helped us understand more about what type of neural network should be used for our purposes; due to the relatively low performance of the CNN in Ehsan's study, we preemptively understood that CNN might not be the best performer for this task and turned our attention to other types of neural networks such as RNN (LSTM is a type of RNN).

In Snow Depth Estimation and Historical Data Reconstruction over China Based on a Random Forest Machine Learning Approach [3], Yang et. al. researched the capabilities of using random forest (RF) as a model to reconstruct historical snow depth data for regions across China. Using a dataset including satellite passive microwave measurements as well as altitude, near-surface soil temperature, historical snow depth, and land cover from various sites located in regions throughout China, these researchers designed four RF models with different combinations of predictor variables. It was found that RF can be a great tool for snow depth estimation, but it has some flaws. One main flaw of the model lies in RF's ability to provide accurate predictions for geographical locations that differ significantly in terms of local climate and land cover. Because the model relies heavily on these local factors, its ability to generalize well is limited. The researchers conclude that employing deep neural networks may help to overcome these limitations. Although our project focused on predicting future snow depth rather than reconstructing historical data, this study still offered relevant insights as the researchers aimed to estimate snow depth using meteorological data through the application of machine learning models, which aligned with our goals. This study helped us to understand that it could be best to create separate models for each location for which we have data to make it easier for our models to generalize to new examples from individual locations, forming our fourth research question.

Overall, these studies told us that using machine learning models to make accurate predictions from meteorological data is feasible. These studies also helped to support our choice of model as Random Forest or Artificial Neural Networks (specifically LSTM) and helped us to further define the scope of our models. Our project can be seen as an extension of the Summit 2 Shore Project by CRREL; general concepts have been discussed amongst members of this lab, but analysis of this type has not been performed prior.

2 Methods

2.1 Data collection and preprocessing

Our data was collected from various sensors located across Vermont. Locations include Mansfield Summit (SUMM), Potash Brook (PTSH), Mansfield West Proctor (PROC), Jericho Forest (JRFO), Mansfield East Ranch (RB05, RB10, and RB11), and Spear St (SPER). The data contains information regarding temperature, precipitation, and wind parameters as well as snow depth information. This data was collected from the CRREL Laboratory of which Vamsi Dondeti is a part. Details can be found below:

Organization: U.S. Army Corps of Engineers (USACE)
Department: Engineer Research and Development Center (ERDC)
Laboratory: Cold Regions Research and Environmental Laboratory (CRREL)

Specific attributes collected include accumulated precipitation levels (real-time and non-real-time), air temperature (Celcius), ice content, snow depth, long wave and shortwave radiation (incoming and outgoing), relative humidity, snow water equivalent, snowpack density, snowpack temperature profile at various heights ranging from 0 cm to 290 cm (not all heights are included for all locations), wind direction, and maximum wind speed. This data contains a multi index; the top row indicates metadata information, the second row indicates the parameter for which values in the column correspond, the third row indicates attribute units, and the fourth row indicates the sampling measurement (for example, average or maximum). For each location, there are four separate data files holding data corresponding to precipitation information, snowpack information, temperature and humidity information, and wind information. Sample tables for each of these data files for the Jericho Forest location can be found in figures one through four below.

TOA5	JRFO	CR1000X	39705	CR1000X.Std.05.02	CPU:MetStation_JerichoForested.CR1X	11701	Precipitation	
TIMESTAMP	RECORD	Intensity_RT	Accu_RT_NRT	Accu_NRT	Accu_total_NRT	Bucket_RT	Bucket_NRT	Load_Temp
TS	RN	mm/min	mm	mm	mm	mm	mm	Deg C
		Smp	Smp	Smp	Smp	Smp	Smp	Smp
2023-01-19 14:30:00	0	0	0	0	121.8	164.9	164.9	0.9
2023-01-19 14:40:00	1	0	0	0	121.8	167.2	165	0.4
2023-01-19 14:50:00	2	0	0	0	121.8	196.5	218.1	0.5

Figure 1
Precipitation.

TOA5	JRFO	CR1000X	39705	CR1000X.Std.05.02	CPU:MetStation_JerichoForested.CR1X	11701
TIMESTAMP	RECORD	T107_C_0cm_Avg	T107_C_10cm_Avg	T107_C_20cm_Avg	T107_C_30cm_Avg	T107_C_40cm_Avg
TS	RN	Deg C	Deg C	Deg C	Deg C	Deg C
		Avg	Avg	Avg	Avg	Avg
2023-01-19 15:00:00	0	-0.13	-17.54	-17.47	-0.021	-86
2023-01-19 16:00:00	1	-0.397	-17.74	-17.67	-0.312	-83.5
2023-01-19 17:00:00	2	-0.728	-17.97	-17.9	-0.651	-86.8

Figure 2
Snowpack. Note: Some columns were excluded from this sample image in the interest of saving space.

TOA5	JRFO	CR1000X	39705	CR1000X.Std.05.02	CPU:MetStation_JerichoForested.CR1X	11701	Wind			
TIMESTAMP	RECORD	WindDir	WS_ms_Max	WS_ms_TMx	WS_ms	WS_ms_S_WVT	WindDir_D1_WVT	WindDir_SD1_WVT	WS_ms_Min	WS_ms_TMn
TS	RN	degrees	meters/second	meters/second	meters/second	meters/second	Deg	Deg	meters/second	meters/second
		Smp	Max	TMx	Smp	WVc	WVc	WVc	Min	TMn
2023-01-19 14:30:00	0	328.7	0.422	2023-01-19 14:30:00	0.422	0.053	327.1	0.826	0	2023-01-19 14:23:00
2023-01-19 14:40:00	1	290.5	0.367	2023-01-19 14:34:00	0	0.109	318.2	13.62	0	2023-01-19 14:32:00
2023-01-19 14:50:00	2	313.1	0	2023-01-19 14:41:00	0	0	0	0	0	2023-01-19 14:41:00

Figure 3
Wind.

TOA5	JRFO	CR1000X	39705	CR1000X.Std.05.02	CPU:MetStation_JerichoForested.CR1X	11701	Table1		
TIMESTAMP	RECORD	Batt_volt_Min	PTemp	AirTC_Avg	RH	shf	Soil_Moisture	Soil_Temperature_C	SW_in
TS	RN			Deg C	%	W/m^2	wfv	°C	W/m^2
		Min	Smp	Avg	Smp	Smp	Smp	Smp	Smp
2023-01-19 14:30:00	0	12.77	0.747	0.105	78.45	-7.453455	0.073	0.6	0
2023-01-19 14:40:00	1	12.77	0.683	0.082	84.8	-7.45011	0.073	0.6	0
2023-01-19 14:50:00	2	12.76	0.58	0.03	84.9	-7.45033	0.073	0.6	0

Figure 4

Temperature and Humidity. Note: Some columns were excluded from this sample image in the interest of saving space.

To manage our data effectively, we employed an (extract, transform, load) ETL pipeline. Data was first extracted from the physical sensors around Vermont and calculations were performed to find values for and create the snow depth column ("DBTCDT"):

Target_Depth = Target_Depth * 100

TCDT = Target_Depth * SQR(AirTC+273.15)/273.15)

DBTCDT = 180.45 - TCDT

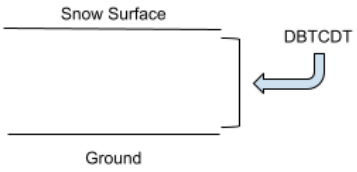

Figure 5.

Visualization of DBTCDT relative to the ground and snow surface.

This resulted in the tables displayed above ("DBTCDT" was within the temperature and humidity data, but is not pictured in the sample table). Minor initial data cleaning was performed at this point. First, the multi-index was unnecessary for our purposes; only the second row containing the parameter for which values in the column correspond was necessary, so we imported each of the four tables for each location using the skiprows parameter to skip the top row indicating metadata information, the third row indicating attribute units, and the fourth row indicating the sampling measurement. Second, we checked for duplicate rows for each table for each location based on the TIMESTAMP column, as there should only be one row in each table corresponding to a specific timestamp. Few duplicates were detected, but some were present. When duplicates of this kind were found, only the first instance was kept as the first instance would represent the initial and thus most valid recording of the weather data at that moment. As the third step of our ETL pipeline, we set up database tables using SQL statements, further transformed the data and loaded it to the database (Phpmyadmin- MySQL). Data was transformed and loaded to the database using one Python script. This script handles data for all four tables from one location at a time. The following actions are performed for one location at a time within the script:

1. Create local database connection
2. Process each CSV (there is one for each of the four tables)
    a. Read the CSV into a DataFrame with pandas' read_csv function.
    b. Check for null values in the "TIMESTAMP" column using pandas' .isnull() method followed by .sum() to count the number of null entries.
        i. This is a safeguard, TIMESTAMP should not and did not have missing values.
    c. Convert values within columns "TIMESTAMP", "WS_ms_TMx", and "WS_ms_TMn" to datetime using pandas' to_datetime function with the parameter errors='coerce' to handle any conversion errors by setting invalid parsing as NaT (Not a Time).
        i. Any NaT values resulting from these conversions were replaced with None using pandas' .replace() method.

    d. Drop the columns for snowpack temperature heights that aren't present in all locations using pandas' .drop() method.
    e. Replace 0s in columns "SW_in", "SW_out", "LW_in" and "LW_out" with None using pandas' .replace() method.
    f. Replace NaN values and NAN strings with None using pandas' .replace() method.
3. Add a new column to each DataFrame representing the corresponding location.
4. Filter each DataFrame's columns so that they must only include the column names within the corresponding table (precipitation, snowpack, temperature & humidity, or wind) within the database.
5. Handle missing values in the transformed DataFrames.
    a. Calculate the number of missing values in each column of each DataFrame using pandas' .isna() method followed by .sum().
    b. Check for NAN strings in the Target Depth column of DataFrames where it is present and find the total number of NAN strings using pandas' .sum() method.
6. Load data to the database.

After this point, two separate methodologies began to take place. Methodology one involved handling data in order to create a model that would be trained on and predict snow depth ("DBTCDT") for a single location at a time. Methodology two involved handling data in order to create a model that would be trained on and predict snow depth ("DBTCDT") for all locations at once. Each methodology involved extracting data from the database, merging or concatenating tables, then cleaning and preprocessing the combined data.

Methodology 1

Our first methodology for our experiment involved only analyzing one location of our data at a time. This was our first step in analyzing whether we could create an accurate machine learning model that is able to predict the generalized weather in Vermont. By analyzing one location at a time, we could compare our results to methodology 2 and determine which was more accurate.

1. Export Data
    a. Execute an SQL query to extract the cleaned records from the database only from the specified location. This query filtered data across the wind and temperature tables.
    b. At this point, the database contained four tables: precipitation, wind, temperature and humidity, and snowpack. After running the models on all of the tables and getting nearly perfect accuracy, it was determined that precipitation and snowpack were not needed for our models as there are likely similar correlations in comparison to the wind and temperature and humidity data that the model can simply 'memorize'. Additionally, there is overlap between many of the features in the precipitation and snowpack tables and many of the features in the wind and temperature and humidity tables, so only wind and temperature and humidity tables were extracted from the database in order to ensure there was no data leakage.
2. Data Preparation and Cleaning:
    a. Load the extracted data into a DataFrame using pandas' read_csv function.
    b. Identify and sum missing values in the dataset using pandas' .isna().sum() method.
    c. Identify blank strings with pandas' .sum() method to detect empty entries across columns.
    d. Add Laplace noise to "Target_Depth" creating a new column "Target_Depth_noisy", utilizing Numpy's random.laplace function.
        i. Including Target_Depth as a feature without adding this noise would result in data leakage as DBTCDT (our target) is the result of a calculation based on Target_Depth as described above.
3. Data Analysis and Feature Selection:
    a. Columns such as TIMESTAMP were broken down into multiple columns (year, month, day, hour, minute, second) and converted into floating point data types.

i. As LSTM can only handle floating point values, “TIMESTAMP” had to be broken into multiple columns (year, month, day, hour, minute, second) and converted to float, so that the timestamp information could all be retained and used in the model.
ii. First, the “TIMESTAMP” column was saved to a separate variable so that the information could be retained to later display the timestamps along with the model’s predictions.
iii. The original "TIMESTAMP" column was then converted into datetime format with pandas’ to_datetime() function. Subsequently, individual components of the datetime (year, month, day, hour, minute, and second) were extracted into separate columns and converted to a floating-point data type using pandas’ .astype() method with a float parameter.
iv. Lastly, the “TIMESTAMP” column was dropped from the DataFrame using pandas’ .drop() method.

b. Implement cyclical encoding for time-related features such as hours and months to maintain the cyclical nature using sin and cos transformations from Numpy, leveraging numpy.pi for calculations.
i. New columns, “hour_sin”, “hour_cos”, “month_sin”, and “month_cos” were created to capture the cyclical nature of these time units. This approach was intended to help the model recognize and utilize the continuity inherent in time data, such as the transition from 23 hours to 0 hours or from December to January, which are important for understanding daily and seasonal patterns in the dataset.

c. Explore the range of feature and label values using pandas’ .max() and .min() methods and handle negative values.
i. This was done to detect if there are negative values for “DBTCDT”, “SW_in”, “SW_out”, “LW_in”, “LW_out”, “RH”, “Soil_Moisture”, “WS_ms”, “WindDir”, and “Target_Depth_noisy”. This was important because negative values should not be present for these columns as that would be physically impossible in the context of the measurements they represent.
ii. For columns that shouldn’t be negative, values below 0 but close to it were likely meant to be 0 while values far below 0 were likely erroneous. So, if a value within these columns was within -5 of 0, it was replaced with a 0. If a value within one of these columns was negative and less than -5, the row containing that value was dropped.
iii. Values within -5 of 0 were found and replaced with Numpy’s .where() function, then rows containing values negative and further from 0 than -5 were dropped using conditional statements to filter out the rows.

d. Conduct a correlation analysis to determine which predictors would be most effective for the models, selecting those with a higher correlation with the target. The predictors that were selected are described in methodology 2 below.

4. Data Splitting:
a. Perform a train-test split using Sci-Kit Learn's train_test_split function, designating 80% of the data for training and the remaining 20% for testing.

5. Normalization and Reshaping for LSTM
a. Specifically for LSTM modeling, reshape predictors into the required format (batch_size, sequence_length, number_of_predictors) to align with the LSTM's input requirements.
b. Adjust labels and timestamps for LSTM to correspond to the sequence ends, ensuring each label and timestamp accurately reflects the sequence it represents.

Methodology 2

Once extracted, the wind table was combined with the temperature and humidity table using an outer merge on columns “TIMESTAMP” and “location”. Data was then cleaned and preprocessed in the following way:

1. Read the merged file as a DataFrame with pandas' read_csv function.
2. Check for missing values using pandas' .isna() method followed by .sum()
3. Check for blank strings using pandas' .sum() method to find the number of blank strings in each column.
4. Add LaPlace noise to "Target_Depth" to create a new column, "Target_Depth_noisy" to allow for target depth to be used as a feature without resulting in data leakage, like we did in our previous method..
5. Drop columns with missing values that will not be used as predictors or labels using pandas' .drop() method.
    a. This was done in order to preserve as much data as possible for model training.
    b. Predictors were determined based on domain expertise and correlation analysis; predictors with a higher correlation with the target were considered to be well-suited to use in our models. Predictors and their meaning are displayed in the table below.

Column Name	Meaning	Units (if applicable)
TIMESTAMP Note: this is not the actual column that was ultimately used as a feature. The process of and rationale behind breaking the values of this column down to be stored in new columns (year, month, day, hour, minute, second) for use in the models is described below.	The date and time that the measurements in the corresponding row were recorded.	
AirTC_Avg	Average air temperature	℃
Soil_Temperature_C	Soil temperature	℃
SW_in	Incoming shortwave radiation	W/m^2
SW_out	Outgoing shortwave radiation	W/m^2
LW_in	Incoming longwave radiation	W/m^2
LW_out	Outgoing longwave radiation	W/m^2
RH	Relative humidity	%
Soil_Moisture	The amount of water contained in the soil	wfv
WS_ms	Wind speed (meters per second)	m/s
WindDir	Wind direction	Degrees
Target_Depth_noisy	The distance from the current snow surface to the previous year's maximum snow height.	cm

    c. Labels include "DBTCDT" as it represents snow depth in cm, the value we'd like to predict.

6. Explore the range of feature and label values using pandas' .max() and .min() methods and use this information to handle negative values, like in method 1.
7. Drop rows with missing values.
    a. To safely remove rows with missing values from the DataFrame, a copy of the DataFrame was first created. After creating the copy, pandas' .dropna() method was employed with the inplace=True parameter on the copied DataFrame to directly remove rows with missing values.
8. Extract aspects of "TIMESTAMP" values so that they can be stored as floating point values in new columns, then drop the "TIMESTAMP" column, as we did in Methodology 1.
9. Perform cyclical encoding for hour and month.
10. Extract predictors and labels based on column names. The predictors and labels are described above.
11. Perform a train test split using Sci-Kit Learn's train_test_split function.
    a. 80% of data was used for training, 20% was used for testing.
    b. Timestamps from the train and test set were also returned from this function so that timestamps from the test set could be displayed with the model predictions to improve the interpretability of the results.
12. Perform a train validation split using Sci-Kit Learn's train_test_split function.
    a. 80% of the training data was used for training, the other 20% was used for validation.
13. Perform one-hot encoding on the "location" column
    a. At this point, all values were floating point aside from those corresponding to the "location" column, which were categorical values represented as strings.
    b. Sci-Kit Learn's OneHotEncoder was fit to the training set, then the training, testing, and validation sets were transformed.
    c. Following the one-hot encoding of the location column within the train, test, and validation sets, the encoded predictors were converted to a DataFrame, then the original location column in each set (training, testing, and validation) was replaced with the new, encoded column using pandas' .drop() method and .concat() function.
14. Normalize the data.
    a. Sci-kit Learn's MinMaxScaler was used to normalize the test, train, and validation sets. Two separate MinMaxScaler objects were initialized (one for predictors, scaler_X and one for labels, scaler_y), then scaler_X was fit to the training predictors while scaler_y was fit to the training labels. Scaler_X was then used to transform the training, testing, and validation predictors while scaler_y was used to transform the training, testing, and validation labels.
15. For the LSTM model only, reshape predictors.
    a. LSTM requires that input predictors are of shape (batch_size, sequence_length, number_of_predictors), where sequence length is the number of consecutive data points (time steps) that the model will consider in each training example.
    b. Predictors were reshaped into sequences that adhere to this format, ensuring each sequence encompasses the specified number of time steps. Once reshaped, these sequences were stored as arrays of floating point numbers using Numpy's .array() function and .astype() method.
16. For the LSTM model only, adjust labels and timestamps.
    a. Labels and timestamps were adjusted according to sequence length. Specifically, they were trimmed to ensure each label corresponded to the end of a sequence created from the input predictors. This adjustment was necessary because the creation of sequences from time-series data as we did leads to a reduction in the length of the dataset, as sequences encompass multiple time steps. Timestep adjustment was done so that the timestamps from the test set could later be displayed along with predictions to make results more interpretable.

2.2 Exploring the data

EDA was performed on the data for multiple locations and the data for a single location, respectively. For each, a heatmap displaying the correlation matrix of features as well as plots of air temperature average over time, soil moisture over time, and snow depth (DBTCDT) over time were generated.

The heatmaps below (figures 6 and 7) display the Pearson correlation coefficients between various features, including the target variable DBTCDT. Both heatmaps show the following: certain features, like AirTC_Avg, PTemp, and Soil_Temperature_C have strong correlations with each other, which is expected given that they are all temperature-related measurements that should reflect ambient environmental conditions. SW_in, SW_out, LW_in, and LW_out also show strong correlations, which is expected given that each of these variables are related to radiation and energy transfer. Additionally, DBTCDT shows relatively significant negative correlation with TCDT and Target_Depth. This is expected as DBTCDT is a calculation based on TCDT and Target_Depth. The strong correlation between DBTCDT and Target_Depth supports the inclusion of Target_Depth as a predictor in our models. Other environmental factors such as RH, Soil_Moisture, and Batt_volt_Min show moderate to low correlations with the target variable DBTCDT. Generally, it appears that when comparing the correlation coefficients for the data for multiple locations with the correlation coefficients for the data for a single location, some correlations appear more pronounced while other correlations are weaker. This could be due to the way data is being aggregated; correlations may become stronger if the relationships between the variables are consistent across locations, while correlations may weaken if the relationships are not consistent across locations.

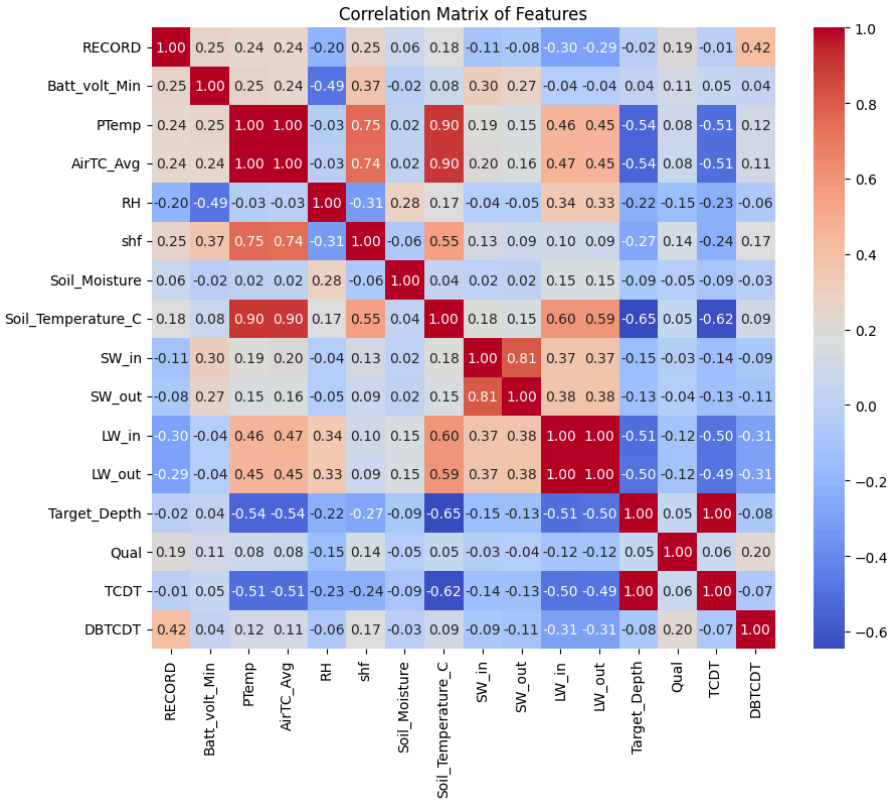

Figure 6: Heatmap of Pearson correlation coefficients for all features for one location (JRFO).

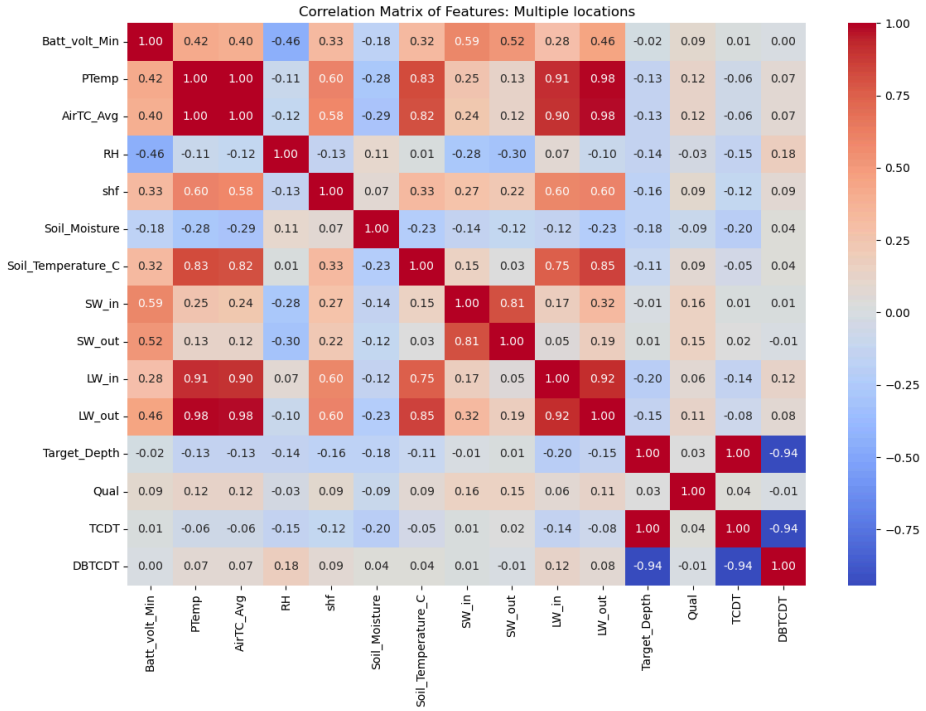

Figure 7: Heatmap of Pearson correlation coefficients for all features for multiple locations (PROC and SUMM).

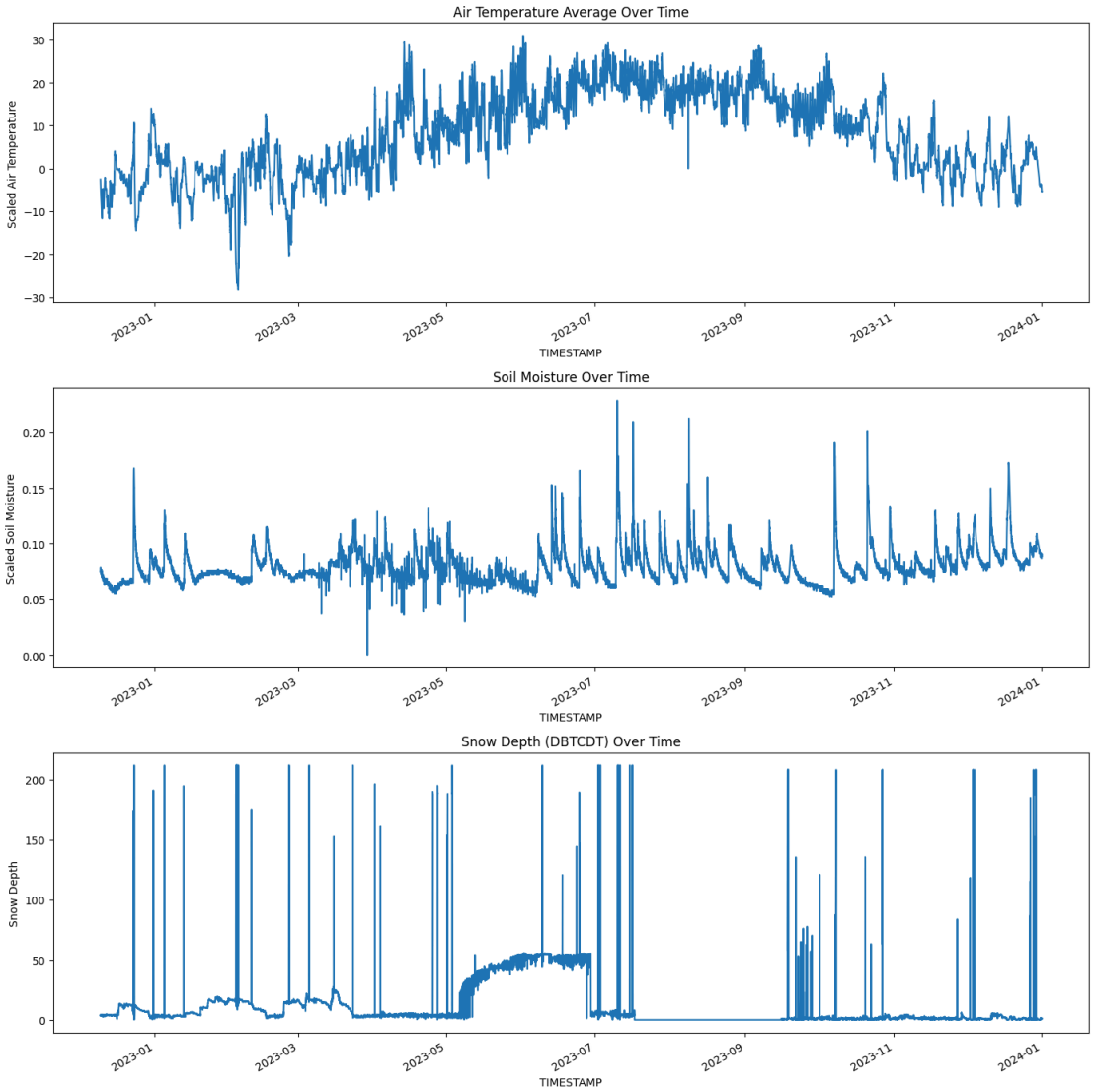

Figure 8: A collection of three subplots corresponding to a single location (JRFO). From top to bottom, the plots display air temperature average over time, soil moisture over time, and snow depth (DBTCDT) over time.

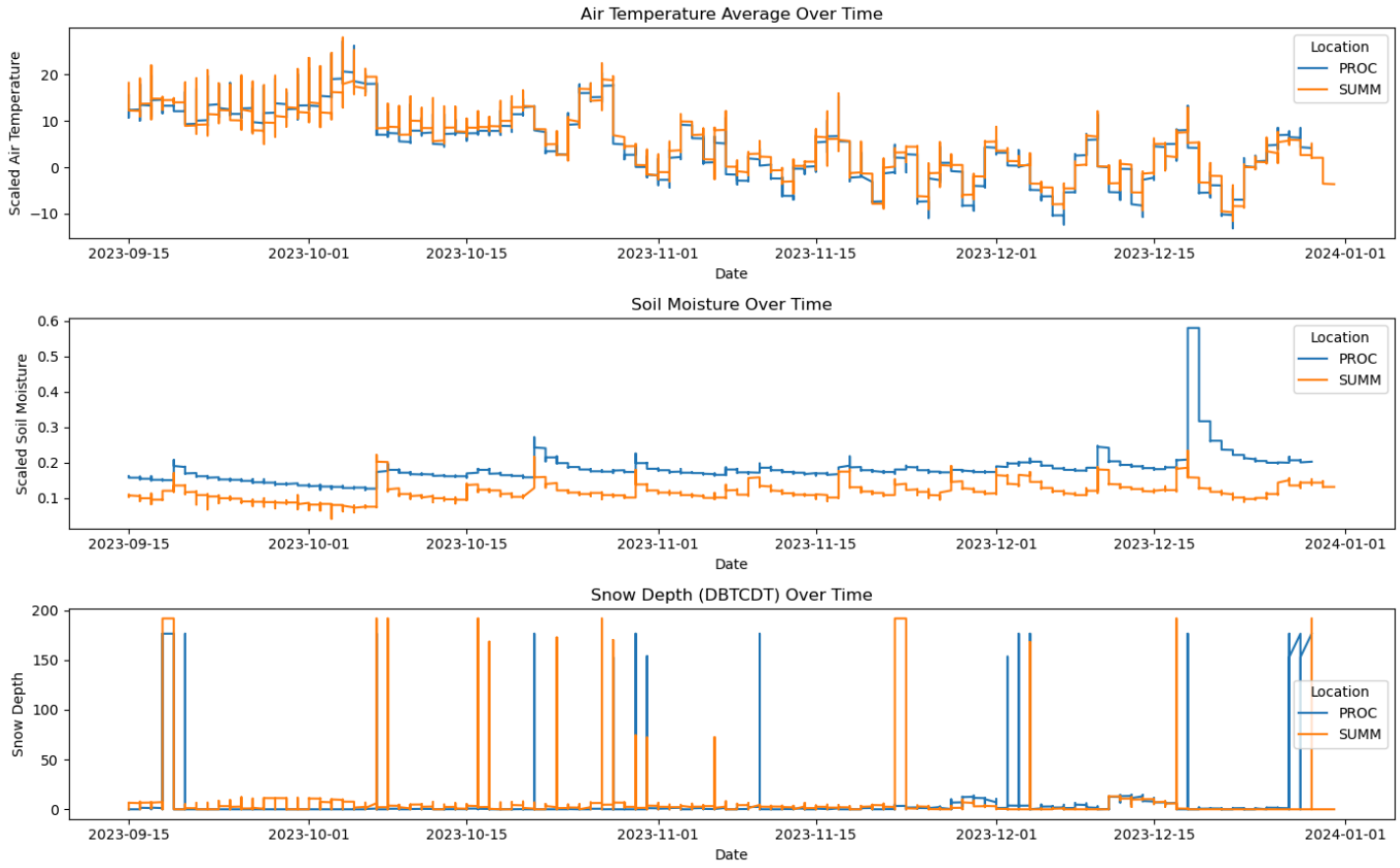

Figure 9: A collection of three subplots corresponding to multiple locations (PROC and SUMM). From top to bottom, the plots display air temperature average over time, soil moisture over time, and snow depth (DBTCDT) over time.

The plots above (figures 8 and 9) provide an overview of how the air temperature average, soil moisture, and snow depth (DBTCDT) vary over time. Similar trends were found for the data corresponding to a single location and the data corresponding to multiple locations; the plots of air temperature average over time show seasonal variations and possibly daily fluctuations, which is typical for climate data. Soil moisture over time appears relatively stable, but shows some variations that could correspond to precipitation events or other environmental factors. Lastly, snow depth (DBTCDT) fluctuates more dramatically, showing spikes that likely correspond to snowfall events and subsequent melting or settling.

2.3 Analysis

As stated prior, our project involved multiple models: two LSTM neural networks were created, one to make predictions regarding snow depth for a single location based on data from that location and another to make predictions regarding snow depth for multiple locations based on data from multiple locations. Additionally, two random forests were created, one to make predictions regarding snow depth for a single location based on data from that location and another to make predictions regarding snow depth for multiple locations based on data from multiple locations. The architecture for each model is described below.

LSTM: Single Location

The LSTM model for predicting snow depth for a single location consisted of an LSTM layer with 6 units. A dropout layer with a rate of 0.2 followed the LSTM layer to prevent overfitting by randomly setting input features to zero during training at each update cycle. This dropout layer was added to enhance the model's ability to generalize, which is often necessary for complex data and complex models such as ours. The output layer was a Dense layer with a single unit.

The model was compiled with the Adam optimizer. Adam was selected because it allows for faster convergence than vanilla gradient descent, but has less risk of overshooting than momentum updates. Through combining the benefits of RMSProp and momentum updates, Adam allows for more effective minimization of the loss function, particularly in complex datasets like ours.

An essential aspect of the LSTM model's configuration was the selection of the sequence length. For our study, a sequence length of 144 was chosen. This decision was based on the fact that within our data, approximately 144 samples constitute one day (24 hours). Selecting a sequence length that covers a full day was crucial for capturing patterns in snow depth, which are influenced by environmental factors that can fluctuate over the course of a day. Additionally, a batch size of 32 was chosen to balance computational efficiency with the stability of the model's learning process.

RF: Single Location

The Random Forest model for predicting snow depth at a single location was designed with robustness and predictive accuracy in mind. This model utilized 100 trees (estimators), a choice driven by the need to balance between overfitting and the model's ability to generalize across diverse data conditions typical in meteorological datasets. We experimented with the number of trees in our model, and decided that 100 was enough to make the model accurate, while not taking up too much processing power.

Each tree in the forest was built from a bootstrap sample, which is a randomly selected subset of the training set, allowing each tree to be trained on slightly different data. This technique introduces randomness into the model, which helps in improving the accuracy and robustness. At each split in the learning process, a random subset of features was considered, which ensures diversity among the trees in the model and contributes to better generalization. The model naturally controls overfitting as well by averaging multiple decision paths, and the inherent randomness introduced in both features and samples during the tree building process prevents any single pattern from overly influencing the outcome.

The selection of these parameters and the model's architecture were grounded in the understanding that snow depth patterns, while influenced by immediate and past weather conditions, do not require the temporal sequence processing of models like LSTMs but can benefit from the robust decision-making capability provided by a forest of decision trees.

LSTM: Multiple Locations

The architecture of our LSTM model for predicting snow depth for multiple locations was also carefully designed to address the specific challenges of our project and was similar in many ways to the LSTM for predicting snow depth for a single location. However, the multiple location LSTM required more complexity as the model had to learn from data derived from multiple sites. The model consisted of an LSTM layer with 250 hidden units and 0.01 kernel and recurrent L1 (Lasso) regularization, followed by a dropout layer with a rate of 0.2 and lastly a Dense layer with a single unit and a Relu activation function.

Similarly to the single location LSTM, dropout was implemented to increase the model's ability to generalize. However, it was also necessary to add regularization to the LSTM layer in the multiple location model due to the increased depth and

thus increased model complexity. Relu was chosen as the activation function for the output layer to prevent the model from outputting negative values. Despite limiting values that should not be negative to have only positive values during data pre-processing, the model consistently output many negative values without the presence of either a Relu or Softplus activation function on the output layer. Relu was chosen over Softplus because Softplus showed a tendency to squash all predicted values to be within a very small range, which was less representative of reality than output when Relu activation was used.

This model was also compiled with the Adam optimizer, had a sequence length of 144, and had a batch size of 32. The same justification provided for the use of Adam, a sequence length of 144, and a batch size of 32 in the single location LSTM applied to the multiple location LSTM.

RF: Multiple Locations

The Random Forest model developed for predicting snow depth across multiple locations is an expansion of the single location model, designed to effectively manage the increased complexity and variability of data from diverse geographical settings. In this enhanced model, the number of trees was increased to 150 estimators. This adjustment was made to better capture the broader range of patterns and noise inherent in data from multiple locations, ensuring robust generalization capabilities.

Each tree within this larger forest continues to utilize bootstrap samples from the training data, a method essential for maintaining the model's resilience against overfitting and ensuring that each tree learns from a slightly different subset of the data. This approach is particularly advantageous when the dataset encompasses a variety of climatic conditions, as it prevents the model from becoming overly biased towards any single location's specific characteristics.

In terms of handling features, this model maintains the random selection of features at each split, promoting diversity in the decision paths and enhancing the overall accuracy. The increase in the number of trees also aids in smoothing out anomalies and reducing variance without compromising the trees' individual depths, which are crucial for capturing detailed data insights.

3 Results

LSTM: Single Location

The LSTM model dedicated to predicting snow depth for a single location performed quite well, achieving a mean squared error (MSE) of approximately 0.0022 and a mean absolute error (MAE) of approximately 0.0147 on the test set. Considering that the typical snow depths observed ranged from 0 to 211.8cm, these error metrics indicate the model is performing quite well in making predictions regarding snow depth, this finding is further clarified by the plot of actual versus predicted snow depth seen in figure 10 (below).

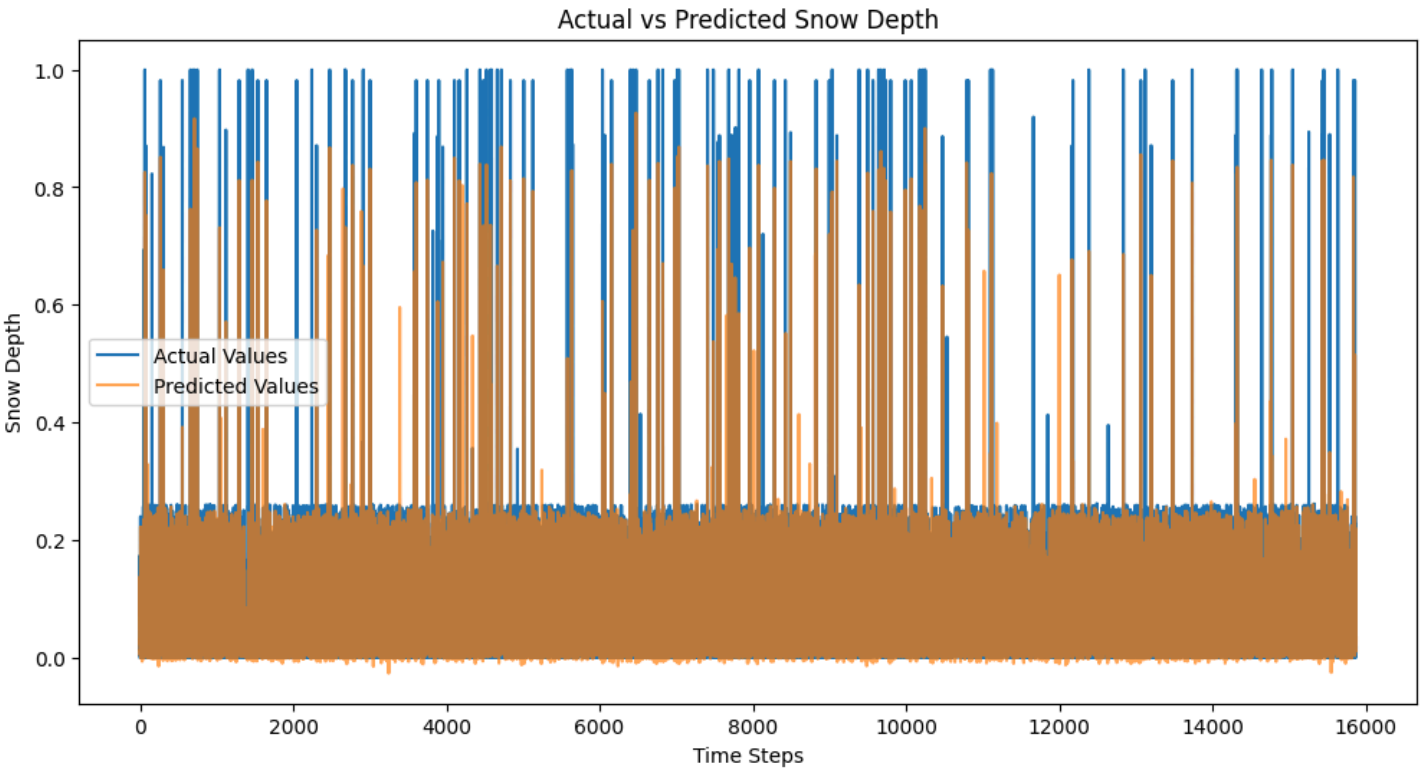

Figure 10: Actual versus predicted snow depth values for the single location LSTM.

RF: Single Location

The Random Forest model for predicting snow depth across multiple locations has demonstrated good performance, achieving a low Mean Squared Error (MSE) of 0.008 and an R-squared value of 0.99 on the test set. These metrics indicate a high level of accuracy in the model's predictions, suggesting that the model is well trained and can predict snow depth in a singular location very accurately.

The near-perfect R-squared value close to 1 suggests that the model explains nearly all the variability in the snow depth data through the features it uses. The MSE, being significantly low, points towards minimal deviation between the model's predictions and the actual data. Specific results from the test dataset further support this, with predictions like 19.0762 cm against an actual of 19.10 cm, and 21.2080 cm against an actual of 21.22 cm, demonstrating the model's precise predictive capability.

Additionally, the training data metrics, with an MSE of 0.013 and an R-squared of 0.99, indicate that the model is neither overfitting nor underfitting. This means that this random forest model is just about as accurate than the LSTM for a single location, if not more accurate. Figure 11 below shows predicted values vs. actual values of the model.

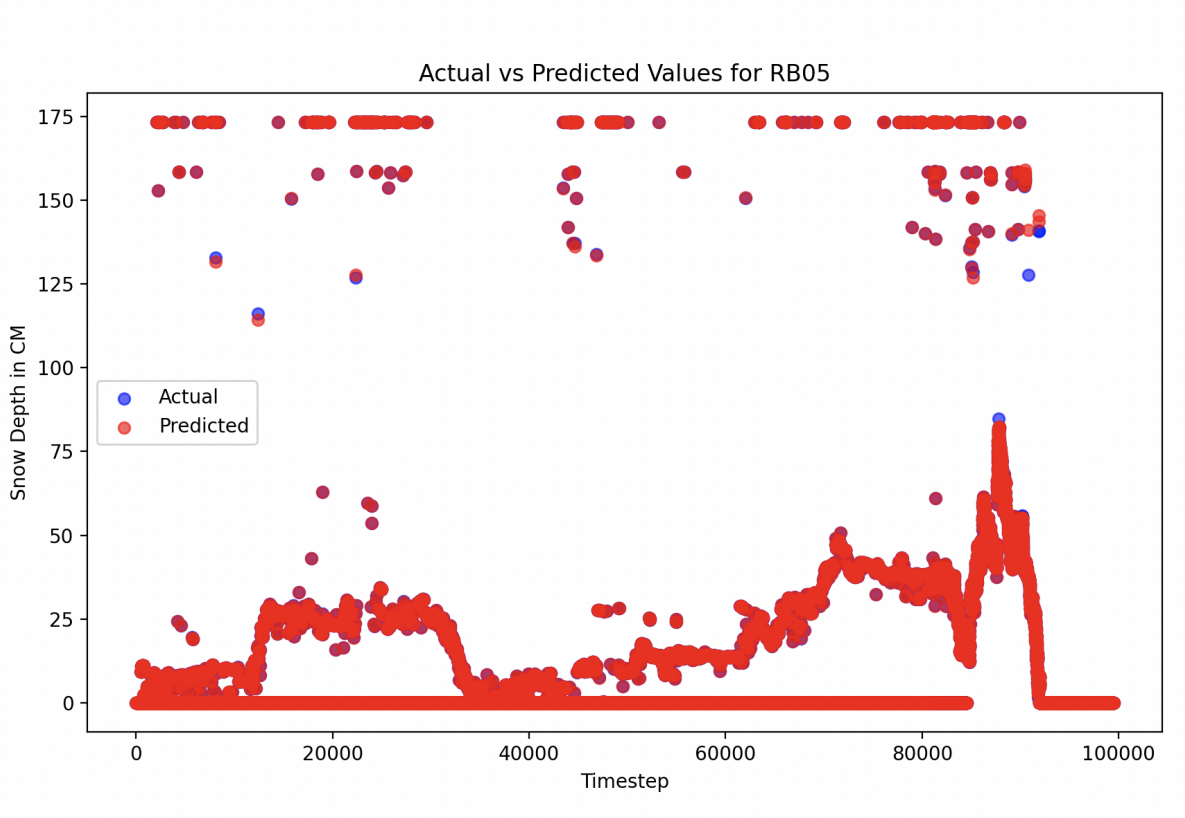

Figure 11: Actual versus predicted snow depth values for a single location (RB05) for the single location RF.

LSTM: Multiple Locations

The multiple location LSTM exhibited considerably poorer performance in comparison to the single location LSTM, resulting in an MSE of approximately 52.0752 and an MAE of approximately 2.7904 on the test set. Although the range of observed snow depth was a bit wider within the data for multiple locations as it varied from 0 to 224.0 cm, the MSE in particular is much higher than expected. The relatively high MSE suggests that the model may be struggling to handle outliers within the dataset. From figure 11, the plot of actual versus predicted snow depth for the multiple location LSTM, one can gain a visual understanding of what kind of mistakes the model is making.

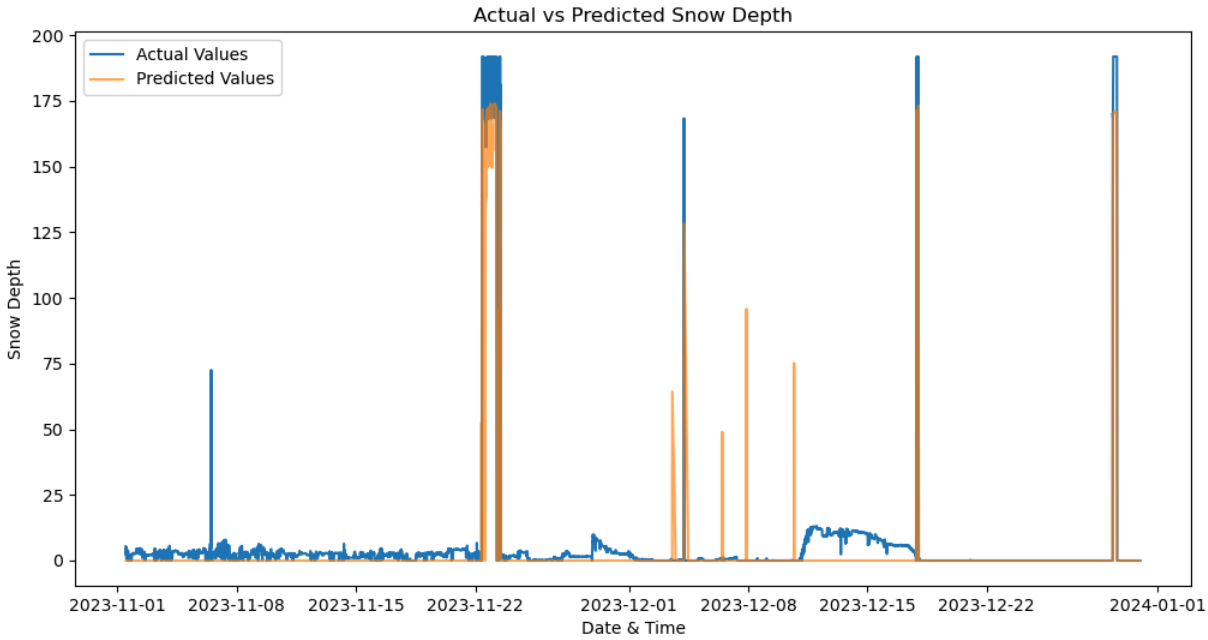

Figure 12: Actual versus predicted snow depth values for a single location (PROC) for the multiple location LSTM.

RF: Multiple Locations

The Random Forest model constructed to predict snow depth across various locations has yielded worse results than when a single location was analyzed, but seems to perform better than the LSTM on multiple locations. This model has a Mean Squared Error (MSE) of 3.66 and an R-squared value of 0.99 on the test set. The training metrics, with an MSE of 0.58 and an R-squared of 0.99, suggest that the model is overfitting; however, the high R-squared values for both training and test sets indicate that the model explains a substantial proportion of the variance in snow depth across different geographical areas.

Despite the low MSE in both training and testing phases, the increase in test MSE compared to the training MSE warrants attention, as it may suggest the model's performance could potentially be optimized further to close the gap between training and testing errors. This disparity might be attributed to the model capturing complex, location-specific patterns in the training data that do not generalize perfectly to unseen data.

While the R-squared value remains high, it's worth noting that this metric does not always capture the model's predictive performance accurately, especially with non-linear models or those that predict probabilities. Nevertheless, the Random Forest's ability to handle the intricacies of multiple locations with varying environmental factors has better results than that of the LSTM. The figure 13 below shows the results of this model.

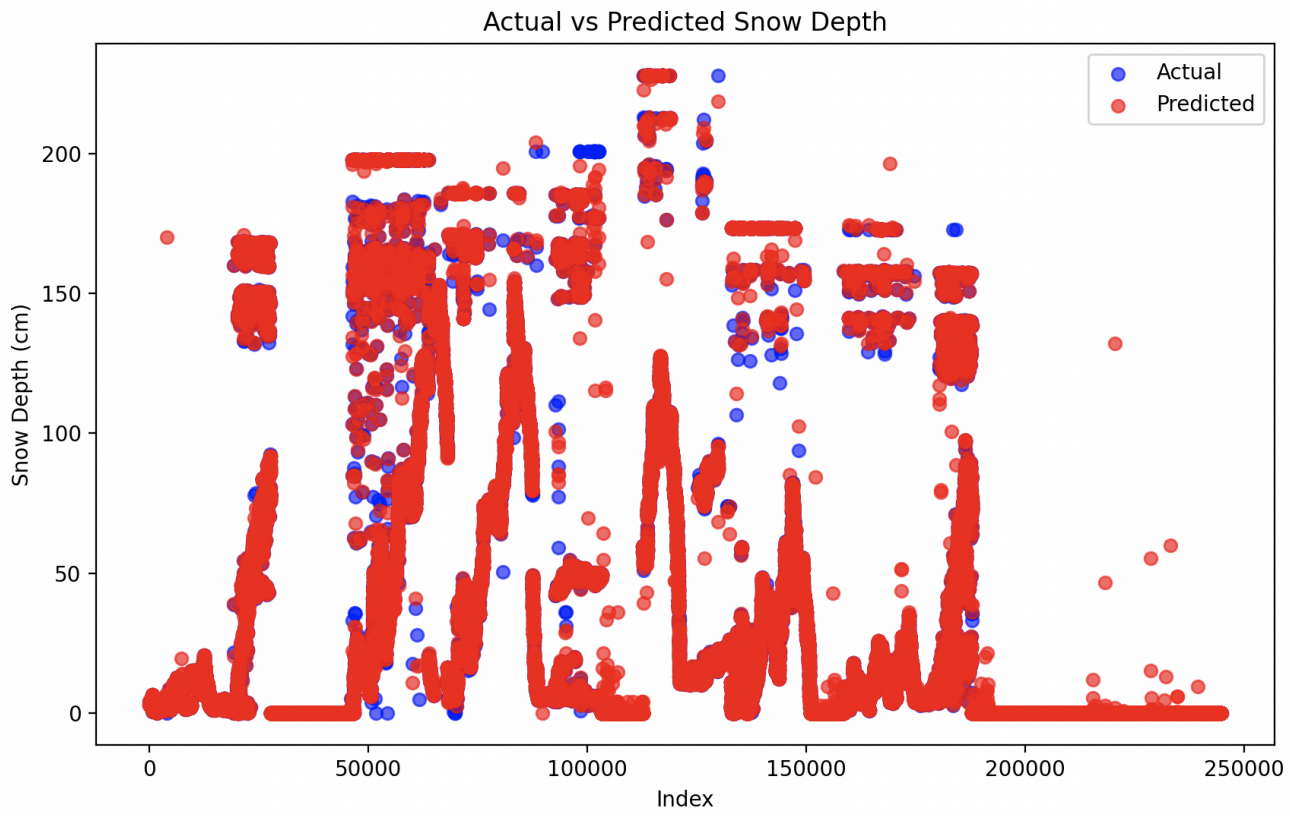

Figure 13: Actual versus predicted snow depth values for RF model trained on all locations

4 Discussion

Our findings confirm that temperature exhibits both seasonal variations and daily fluctuations over time, which aligns with typical meteorological expectations. Additionally, we found that snow depth fluctuates more dramatically, showing peaks that likely correspond to snowfall events and subsequent melting or settling. These results are illustrated in figures 8 and 9.

Figures 6 and 7, which display Pearson correlation coefficients for various predictors including snow depth, reveal that temperature (AirTC_Avg) and snow depth are not as strongly correlated as one might expect. Due to the impact temperature has on snow formation and melting, common intuition might suggest that there should be a strong inverse correlation between snow depth and temperature. However, a correlation coefficient of 0.11 was observed between these features in the data for a single location, and this correlation was found to be 0.07 in the data from multiple locations. The reduced correlation in data for multiple locations could suggest that the relationship between temperature and snow depth is not consistent across different geographical areas.

Far better performance was exhibited by the single location models in comparison to the multiple location models in both random forest and LSTM. This finding implies that perhaps single location models are best suited for predicting snow depth. This could be due to additional complexity that is introduced through aggregating data from various sites.

5 Conclusion

In conclusion, our comprehensive analysis of snow depth data using machine learning techniques yielded significant insights and provided answers to our initial research questions. Our exploration into temperature trends and correlations with snow depth underscored the intricate nature of these variables. While one might anticipate a strong inverse relationship between temperature and snow depth due to their direct impact on snow formation and melting, our findings highlighted a more complex interplay that is likely influenced by many factors beyond just temperature.

Regarding our main objective to discern whether individual location models outperform multi-location models, our results were quite revealing. Both the LSTM and Random Forest models trained on single-location data consistently outperformed their multi-location counterparts in terms of predictive accuracy. This suggests that while machine learning models are indeed capable of predicting future snow depth with a considerable degree of accuracy, the optimal performance is achieved through models tailored to specific locations rather than using a generalized approach across multiple locations.

The multi-location models, while still displaying a decent level of accuracy, were likely to overfit, as seen in the larger discrepancy between training and testing errors. The multi-location LSTM model, in particular, faced challenges in capturing the variable patterns across different locations, which was reflected in its higher mean squared error.

Both the single-location LSTM model and the single-location Random Forest Models demonstrated exceptional performance with a very low mean squared error and mean absolute error. The Random Forest's model's MSE was slightly higher than the LSTM, but had a near-perfect R-squared value. Overall, since the MSE difference between the LSTM and Random Forest was negligible, we believe that both a RF model and an LSTM trained on a single location can be accurate predictors of snow depth.

References

[1] Bochenek B, Ustrnul Z. Machine Learning in Weather Prediction and Climate Analyses—Applications and Perspectives. Atmosphere. 2022; 13(2):180. https://doi.org/10.3390/atmos13020180

[2] Ehsan, M. A. (2021). Predictive models for wind speed using artificial intelligence and copula. arXiv preprint arXiv:2107.06182.

[3] Yang, J., Jiang, L., Luojus, K., Pan, J., Lemmetyinen, J., Takala, M., & Wu, S. (2020). Snow depth estimation and historical data reconstruction over China based on a random forest machine learning approach. The Cryosphere, 14(4), 1763-1778. https://doi.org/10.5194/tc

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2.

McKinney, W., & others. (2010). Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56).

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.