MANUSCRIPT ID: 000000

NOAA Daily Weather Data Analysis and Modeling for Charleston, South Carolina (2022)

John W. Smutny, Ben C. Johnson, Anagha Mudki, James A. Ensminger

***Abstract* — Predicting the weather is an act that influences society everyday of every year. A prediction influences how an individual may travel to work, whether a business can operate, or any other countless effects. This report aims to use linear regression and classification machine learning models to predict how much precipitation will occur around the Charleston International Airport in Charleston, South Caroline of the United States of America. These models will be trained using over 70 years' worth of atmospheric data provided by the National Oceanic and Atmospheric Administration (NOAA) of the United States of America's government.**

Both models are trained by analyzing a full day's worth of measurements. After each model is trained, they attempt to predict two items; 1) whether the next day will have a precipitation event (IE: there is non-zero rain, snow, hail, etc) and 2) how much the next day's precipitation will be. These two predictions will be made twice: first using only the current day's weather, then second using the current and previous day's weather.

After both sets of models were created; the influence of more data affected the classification model and linear regression models differently. As the amount of data used in the training set of the classification models increased, so did the model's testing accuracy and model's prediction confidence scores. Meanwhile, as more data was considered for the linear regression model, the model's ability to predict how much precipitation would occur did not noticeably improve.

Index Terms*—Decision Tree Classification, Linear Regression, Model Accuracy, Supervised Machine Learning, Weather Prediction*

I. Introduction

The goal of this paper and the research performed is to determine a pair of models to predict the next day's precipitation that will occur at the Global Climate Observing System (GHOS) probe located at the Charleston International Airport in Charleston, South Carolina using the current data provided from the USW000013880 csv file. The paper is divided into several sections: pre-data preparation, post-data preparation, decision tree models and linear regression models.

The pre-data preparation contains the raw data recorded from the Global Climate Observing System (GHOS) probe located at the Charleston International Airport in Charleston, South Carolina containing multiple measurements per day. The goal is to detect information that should be modified in order to be used in the models.

The post-data preparation is the raw data computed to fix any aberrant values, remove any unnecessary data, and merge the multiple measurements of each day down to one daily entry. The post-data preparation also contains the columns of PREV_PRECIPFLAG, PRECIPAMT, PRECIPFLAG and PRECIPAMT. PRECIPFLAG is set if there was any precipitation, either snow, rain etc on that day and PRECIPAMT is the total amount that occurred. NEXTPRECIPFLAG and NEXTPRECIPAMT are created as well, these will become the two target variables for the models.

The two decision tree models used to predict if there will be any precipitation the next day use either Entropy and Gini Index. These models are judged based on their calculated training score and their testing score (or accuracy) with the provided post-data preparation dataset. The analysis of these scores determine which data elements should be considered.

The Linear Regression and RidgeCV models are used to predict how much precipitation will occur on the next day. These models are judged based on their Mean Squared Error (MSE) and their R squared coefficient. In order to calculate the ground truth analysis of model performance the data was divided into training and test partitions, using a 70/30 split, with 70 percent being the training data while the ramianing 30 percent was used to verify the data. The reliability of the data is measured using MSE. With the MSE we can calculate the R squared, the R squared score is used to indicate the correlation between the input features and the target on a linear basis.

II. Station information

The station analyzed in this paper is the USW000013880 National Oceanic and Atmospheric Administration (NOAA) Global Climate Observing System (GHOS) probe located at the Charleston International Airport in Charleston, South Carolina, United States of America [1]. This probe is also used as a part of the World Meteorological Organization (WMO). Table SD1 shows more specific information about the probe.

TABLE SD1
NOAA Weather Station Details used in report

Latitude (degrees)	32.8986
Longitude (degrees)	-80.0403
Elevation (meters above sea level)	12.2
Location	Charleston international airport, SC, USA
WHO ID	72208

The data collected from M.J. Menne and their collaborators probe provided daily data (measured at four potential times of day (06:00, 12:00, 18:00 or 24:00)) between the dates of March 1st 1937 and February 27th 2022. The probe is capable of measuring over 70 different types of atmospheric data ranging from rainfall, soil temperature, evaporation, and wind speed.

III. Data Discovery

A. Original Dataset Collected

The NOAA atmospheric probe has the potential to measure a diverse range of metrics relevant to the weather and is complemented by a variety of sources. Over years of deployment, the probe's variety of metrics has increased to provide the potential for an elaborate collection of data with a varied amount of accuracy.

The original unedited dataset took several measurements per day and alongside several quantities describing each measurement. For each data entry measured includes the following fields: The station ID measured from, the Date collected, what Element (measurement) was collected, the measured Value (in unique units to each Element), a Measurement Flag providing additional context about the measurement value, a Quality Flag identifying if there were any errors collecting the measurement, a Source Flag identifying where the data came from, and the OBS Time that the measurement was taken.

Please see the appendix 'Description of field names, etc' for more information or the 'NOAA Global Historical Climatology Network Daily' website.

B. Pre-Processing Data Quality Report

Before performing any data analysis or model training; there needed to be a greater understanding of the dataset provided by the Charleston NOAA weather station. In response, a Data Quality Report describing each feature of the Charleston weather data was created to plan how the data will be manipulated to yield the best models.

Table DD1 below shows an initial data quality report of the original dataset and (if relevant) mathematical details about the particular fields. The dataset's fields are a combination of numeric (Value, ID), ordinal (Date, Time), and categorical (Element, Measurement Flag, Quality Flag, Source Flag) entries.

Please see the next page for the full section "Initial Data Quality Report"..

This initial report provides observations to lead the later steps in the data preparation process. Some observations of note are below.

a. (ID) There is only one listed value for the station ID. Therefore, all of the data used is from the same location.
b. (QualityFlag) Out of the original 443681 values, 443358 (99.927%) of them are not flagged with any measurement errors.
c. (QualityFlag) Of the 0.073% (323 instances) of data with QualityFlags; the majority (218 of 323 or 67.5%) of flagged measurements failed NOAA's bounds check (symbol 'X') [1]. Please see appendix '**Description of field names, etc**' or the NOAA '**readme-NOAA_Data.txt**' for more information on flag values.
d. (Date) On average; each date has 25 measurements.
e. (Value) Based on the descriptions of the physical measurements taken, there is at least one invalid measurement since the maximum measurement value is 32767 (IE 2^{15} = size of int).

TABLE DD1

DATA QUALITY REPORT OF THE USW00013880 DATA WITHOUT ANY MODIFICATIONS.

stat	ID	Date (yyyymmdd)	Element	Value	Measurement Flag	Quality Flag	Source Flag	OBS Time (hhmm)
count	443681	443681	443681	443681	443681	443681	443681	443681
cardinality	1	31046	61	1845	5	4	9	5
mean	*	19849159	*	185.2947	*	*	*	2113.852
median	*	19860127	*	75	*	*	*	2400
number at median	*	20	*	124	*	*	*	95217
mode	*	19950101	PRCP	0	W	X	0	2400
number at mode	*	25	31045	93642	17104	219	211231	95217
stddev	*	205247.7	*	363.5312	*	*	*	585.3441
min	*	19370301	*	-178	*	*	*	700
max	*	20220228	*	32767	*	*	*	2400
number of zeros	443681	N/A	*	93642	*	*	*	N/A
number missing	0	0	0	0	416519	443358	0	322862

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature

IV. Data Preparation

For each step outlined in the previous section, please see the corresponding section in the 'Data Pre-Processing Steps and Reasoning' appendix area.

*A. **Analysis of the Original Dataset***

At the time of writing, the NOAA Charleston probe took 443681 total measurements, of 61 different types over the course of 31046 days starting on March 1st of 1937. By using various data processing techniques and software, the original dataset of 443681 measurements is refined so that the linear regression and classification models can make better predictions. The following processing steps are listed below.

1. Removal of Invalid Data Measurements
2. Repurposing the Measurement Flag 'Trace' Category into PrecipitationFlag
3. Removal of Unnecessary Fields
4. Added Column Describing the Number of Events in a Day
5. Specify Element Measurements used on Modeling
6. Combine all measurements that occurred in a single day.
7. Populating Model Target Values

Each step of the data preparation process is outlined in greater detail below in the next few sections.

1. Removal of Invalid Data Measurements

The Quality Flag on a data entry provides insight into any potential issues with that data measurement. Some example issues are 'inconsistency check', 'failed bounds check', and 'failed duplicates check' [1]. In order to train the models using accurate data, any data entries with a Quality Flag were removed from the dataset.

Every data entry with a Quality Flag was removed, because removing the entries promoted confidence in the data used to train the models without any significant tradeoffs. First, removing the data entries with Quality Flags accounts for 0.073% of the dataset and does not leave any day without at least one data entry (see Figure DP2 in the appendix). Secondly, the 0.073% of data entries with Quality Flags are not concentrated in a particular source. The source with the highest concentration of Quality Flag data entries is the U.S. Cooperative Summary of the Day (source '7') with 0.8259% of its values having Quality Flags (see Table DP3 in the appendix). Finally, the 0.073% of data entries with Quality Flags are also not concentrated by a particular Element being measured. The Element 'Daily minimum temperature of water in an evaporation pan' has the highest concentration of flagged data entries with 0.01522% having a Quality Flag (see Figure DP4 in the appendix for the other non-zero percentage Elements).

Due to the negligible scope of data entries with QualityFlags, any entry with a Quality Flag was deleted to have better confidence in the models.

2. Repurposing the Measurement Flag 'Trace' Category into PrecipitationFlag

To increase the accuracy of the 'Next Day Precipitation' prediction of each of the models, the 'Trace precipitation' MeasurementFlag was repurposed to flag when any precipitation event occurred (even if all precipitation amounts were zero). See the logic 1 below for how NEXTPRECIPFLAG is set to True or False.

Logic 1
Conditional logic to set bollean NEXTPRECIPFLAG

$$If\ ((PRCP\ >\ 0\ ||\ SNOW\ >\ 0)\ ||\ MFlag\ ==\ 'T')$$
$$then\ (NEXTPRECPIFLAG\ =\ TRUE)$$
$$Else$$
$$then\ (NEXTPRECIPFLAG\ =\ FALSE)$$

From the original dataset, 12.2% (3779/31046) of the data was labeled with the 'T' (trace) 'MeasurementFlag' and had zero precipitation or snow measurements. This manipulation was done to accommodate the end user of the model in deciding how to approach a day if there is a chance that some precipitation would happen, even if negligible. An example case of a day with trace (but negligible) precipitation changing an individual's plans is whether to hold a picnic or a multi-hour outdoor event. Any precipitation in this case would require extra planning. It is worth noting that the original dataset defined by Menna describes the 'Trace' Measurement Flag including precipitation, snow and snow depth. However, this model is only concerned with predicting precipitation from the air, so any 'snow depth' indicators were not included in our logic. By assuming that a 'Trace' of precipitation counts, there will be a floor of

After the 'MeasurementFlag' was used to increase the accuracy of the 'Next Day Precipitation Flag'; the 'MeasurementFlag' column was removed since the other 'MeasurementFlag' categories do not change how the measurement values are interpreted. The categories elaborate on how the values were calculated. All values with 'MeasurementFlag' 'P' were assumed to be zero since there were no other reasons to doubt the researchers. The full definitions of each 'MeasurementFlag' is given below [1].

H = How value was calculated (Temperatures)
P = Missing (presumed 0)
T = Trace (trace of precipitation, snowfall, or snow depth)
W = converted from 16-point WBAN code (for wind direction)

3. Removal of Unnecessary Fields

After reviewing the fields contained in each data entry, some fields were removed entirely given how the models are trained and the model's goals. The Source Flag was removed since its information does not affect model predictions since there is no data on source accuracy. Field OBS-Time was removed, because the models are trained using daily summaries of measurements, not the time of day that the measurements were recorded.

4. Added Column Describing the Number of Events in a Day

A new column was added to the final dataset to articulate how many measurements were taken each day. Since the resulting data used to generate all of the models are aggregations of individual measurements over the course of a day; the authors believe it is a useful statistic if the data needs to be more closely examined for any reason.

5. Specify Element Measurements used on Modeling

Every data element created after processing was used, because removing the entries would remove a source of information that could be used in calculating the accuracy of Next day precipitation amount.

6. Combine all measurements that occurred in a single day

The last processing step of already existing data from the NOAA station involved reducing the dataset in size. The dataset is reduced so that all weather measurements that were recorded for a day are summarized into one single data entry. This summarizing step is taken so that the models can focus on predicting the precipitation that occurs on a day, instead of considering 'when' in a day that precipitation could occur.

7. Populating model Target Values

Depending on the model type, the models outlined in this report are attempting to predict one of two features about the weather; 1) whether there will be any negligible precipitation the next day or 2) how much precipitation there will be in the next day. For each day in the resulting dataset, these target features are determined based on the day's other aggregate measurements using the relation & equation below. In summary; the NEXTPERCIPFLAG is a boolean quantity and is set to TRUE if throughout the day in question there was a non-zero amount of precipitation, snow or a 'Trace' of precipitation (see Logic 1 from in step 2). The NEXTPERCIPAMT is a numeric quantity that is a measure of the sum of precipitation and snowfall in a day (1).

$$NEXTPRECIPAMT = PRECIPITATION + SNOWFALL/8 \quad (1)$$

Both target values are created and generated to train all of the used models. All of the data processing steps resulted in a structure for each data entry in the dataset seen in appendix Figure DP5.

B. Post-Processing Data Quality Report

After all of the data processing steps outlined above, another Data Quality Report is created to analyze the new dataset that will be used to train the models. In the final dataset, there are columns not only for the date and our target variables (NEXTPERCIPFLAG and NEXTPERCIPAMT), but also include a new column for every unique value of the 'Element' feature in the original dataset.

One important note about the final Data Quality Report is that the final dataset's number of entries has decreased from 443681 entries to just 31046 data entries. This is due to the 'Combine all measurements that occured in a single day' processing step that merged all of a day's measurements together into one data entry.

Please see the appendix figures DQR2-a through DQR2-o for the full Data Quality Report after the data processing steps above were completed.

This final report before training models helps confirm that the data processing steps were successful.

a. (DATE) There is now only one line of information per date. This can be seen in the cardinality of the Date, since its value equals the count value.
b. (PREV_PRECIPAMT, PRECIPAMT, NEXTPRECIPAMT) All of their values are the same. This is because the data in each table is nearly identical since NEXTPRECIPAMT becomes PRECIPAMT on the next day and PRECIPAMT becomes PREV_PRECIPAMT (note this is the name for each respective flag, even though the table does not show this).

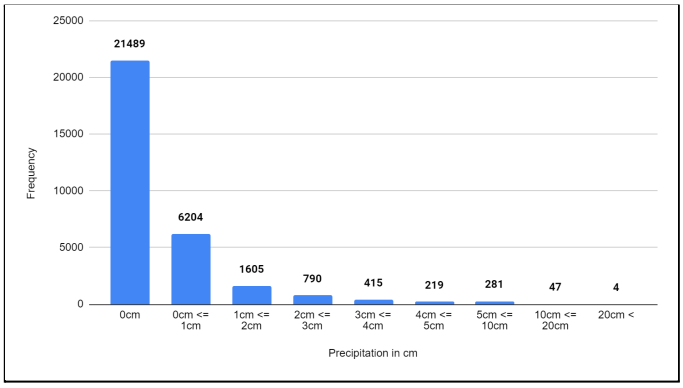
c. (TMAX and TMIN on Table DQR2-b) The average range of temperatures throughout the day in the processed dataset is 13.9*C (57.02*F) to 25.6*C (78.08*F) throughout the years measured. This range is approximately in the same range of yearly temperatures for North Charleston, South Carolina (see Table DQR2-p in the appendix for the range of temperatures for 2021). This helps build confidence that there was not a major issue with one of the core weather measurements.

C. Observations of the Target Variables Prior to Training

Before any models are trained; looking at the generated target variables of NEXTPERCIPFLAG and NEXTPERCIPAMT can provide insight into what to expect from each model. Looking at Figure DP6, the histogram of 'how much precipitation falls when precipitation occurs' helps manage expectations on how much precipitation the Linear Regression model should train for.

Based on the histogram, the highest occurring amount of precipitation is 0cm, which accounts for nearly 69.29% percent of the calculated days. This trend towards zero in the target feature means that the target is heavily skewed toward the origin. Table DP7 also supports the skew right behavior towards the origin since the first quartile and second quartile (median) of NEXTPRECIPAMT is set to a value of zero and the third quartile is being set to a value of 0.10 cm, while the max value is 29.21cm.

Graph DP6
Frequency of occurrence of the amount of precipitation



Due to the extreme density of 0cm occurrences in a day's precipitation (as seen in Graph DP6), the disruption of NEXTPRECIPAMT values are heavily skewed near 0.

TABLE DP7
This table shows the Min Max, Median, Average and the Quartiles of the predicted Next Day precipitation amount.

Max. Nextprecipamt	2,921
Min. Nextprecipamt	0
Median Nextprecipamt	0
Avg. Nextprecipamt	36
PERCENTILE([Nextprecip..	0
PERCENTILE([Nextprecip..	10

It is important to note that when NEXTPRECIPFLAG is set to TRUE, it does not guarantee that a non-zero amount of precipitation occurred that day. During data preparation, any data measurement with a 'Trace' measurement flag and zero total precipitation still sets the day's PRECIPFLAG as TRUE. As seen in Table DP8, the percentage of NEXTPERCIPFLAG and NEXTPERCIPAMT FALSE values are not the same. There are 11.73% more TRUE instances of NEXTPERCIPFLAG than NEXTPERCIPAMT.

TABLE DP8
Percentage difference of TRUE NEXTPRECIPFLAG values and non-zero (TRUE) NEXTPRECIPAMT values in the processed dataset.

NEXTPRECIPFLAG	%	NEXTPRECIPAMT	%
FALSE	**57.48%**	Zeros (FALSE)	**69.20%**
TRUE	**42.52%**	Non-Zeros (TRUE)	**30.80%**
Difference of NEXTPRECIPFLAG vs NEXTPRECIPAMT			
11.72%			

MANUSCRIPT ID: 000000

V. Prediction Models

A. Classification: Predicting if there will be any Precipitation

1. Summary of Approach

The first model used to predict whether there will be any precipitation the next day (NEXTPRECIPFLAG) is the Decision Tree algorithm. The Decision Tree algorithm belongs to the family of supervised learning algorithms that uses training data to create rules for decision making. The goal of using a Decision Tree is to create a training model that can be used to predict the class or value of a target variable by learning simple decision rules inferred from prior data (training data).

For the Decision Tree models discussed below, two metrics were used to create the rules that make up a tree; 1) Entropy (disorder of a dataset) and 2) Gini Index (probability of a wrong classification). The first rule is known as the 'Root Node' and is the most influential for any decision tree. When the root node is chosen then the tree expands to another branch and repeats the process of selecting a new rule to further expand the tree, going from node to node. How well the training data built decision tree classifies new data is determined by an accuracy score (See Equation (2) in the next section).

Various Decision Tree factors were considered when generating the classification model to best predict if there will be a negligible precipitation event the next day. The factors include 1) which Element types are considered for analysis, 2) the tree depth, and 3) which criterion is used in the model (Entropy or Gini Index). Each model's result is judged based on their accuracy score and if the model is over/under fitted to the training data.

$$Accuracy = (Test\ set\ correctly\ predicted)/(Test\ set) \quad (2)$$

To determine Accuracy classification score, metrics module was used from sklearn library. Using this module y_test values are compared with the predicted values and we get the accuracy of the model.

2. Classification Models

A total of six decision trees were created to predict if there will be any negligible precipitation event the next day. Three models were based on the Entropy criterion and the other three were based on the Gini Index. Each of the different criterion models considered a different number of Element measurements. Figures C1 and C2 below show the first models generated when considering only the five core Element measurements as defined by NOAA and MJ Menna: Temperature max/min, precipitation (rainfall), snowfall, and snow depth [1].

Figure C1
Gini Index Classification model with a tree depth of 5 and considering all the parameters.

Figure C2
Entropy Classification model with a tree depth of 5 and considering all the parameters.

3. Classification Model Results

After creating models with different combinations of decision tree settings, several trends began to emerge. Tables C3 & C4 show the accuracy scores for both criterions (Entropy & Gini Index) decision tree models based on which Element measurement features were considered.

MANUSCRIPT ID: 000000

Table C3
Model characteristics for Core5 Element Parameters

Classifier	Depth	Training score	Testing score or accuracy	Root node parameter
Entropy	5	66.67%	65.85%	PRECIPFLAG
Entropy	10	68.77%	65.25%	PRECIPFLAG
Entropy	15	74.31%	62.26%	PRECIPFLAG
Gini	5	66.81%	65.88%	PRECIPFLAG
Gini	10	69.31%	64.97%	PRECIPFLAG
Gini	15	74.50%	63.79%	PRECIPFLAG

Table C4
Model characteristics for All Element Parameters

Classifier	Depth	Training score	Testing score or accuracy	Root node parameter
Entropy	5	67.88%	66.59%	PRECIPFLAG
Entropy	10	72.20%	64.99%	PRECIPFLAG
Entropy	15	79.54%	64.57%	PRECIPFLAG
Gini	5	67.91%	67.59%	PRECIPFLAG
Gini	10	73.14%	66.36%	PRECIPFLAG
Gini	15	81.23%	65.16%	PRECIPFLAG

4. Discussion and Observations of Results

By observing the various models created; many decision trees had similar characteristics, particularly on what features to use for their earliest decisions. The earlier that a feature is used in the classification model, the more important and influential that feature is in that model's decision making. Based on the models created, the most important variable (root node) in the models is the current day's precipitation flag (PRECIPFLAG) that was added to the dataset. Other important features used in the beginning of the decision tree are 'amount of rain precipitation' (PRCP), the previous day's precipitation amount (PREV_PERCIPAMT) and the minimum temperature (TMIN).

Another visible observation of the created classification models is that there are various relationships between the models and accuracy; 1) how the number of parameters considered effects accuracy, 2) how tree depth effects accuracy, and 3) how various settings affects the confidence that the model predicts correctly if there will be any negligible precipitation event.

1. The accuracy of the model, where only five parameters were considered, is less than the model where all the parameters were considered. The difference is extremely minimal. This has happened as in later cases; considering more elements were compared to previous. As more parameters were considered for training increases, the resulting model accuracy also increased.

2. When the depth of the tree increases, the training score increases and the accuracy (or the testing score) decreases. Initially; when the depth is 5, both scores are nearly the same. As the model's tree depth increases, the training score increases since a deeper tree can make more decisions to subset the dataset. Therefore, increasing tree depth increases the model's accuracy on the training set.

3. As the amount of measurement parameters considered increases across all of the other model settings, the confidence that the classification model makes accurate predictions slightly increases. This means that with more available parameters to make decisions, users of the model can be more confident in the model's ability to make the correct prediction (the model predicts precipitation and precipitation occurs or vice versa). The tables below (Table C5 & C6) detail how well the model predicts the target value of the boolean NEXTPRECIPFLAG (values of 0 or 1) by the true positive and true negative rates. The percent is the number of times the model has predicted the correct flag value for the next day precipitation flag. As the decision tree gets more data values the correct prediction percent also increases. However, the larger the decision tree, the lower the accuracy of the predictions.

Table C5

Confidence of the Classification model's predictions based on model settings for only the core 5 measurement parameters.

Classifier	**Depth**	**True Negative Accuracy that NEXTPERCIPFLAG will be 0**	**True Positive Accuracy that NEXTPERCIPFLAG will be 1**
Entropy	5	68%	61%
Entropy	10	68%	62%
Entropy	15	65%	59%
Gini	5	67%	63%
Gini	10	67%	61%
Gini	15	65%	58%

Table C6

Confidence of the Classification model's predictions based on model settings when using all of the measured parameters.

Classifier	**Depth**	**True Negative Accuracy that NEXTPERCIPFLAG will be 0**	**True Positive Accuracy that NEXTPERCIPFLAG will be 1**
Entropy	5	70%	62%
Entropy	10	69%	62%
Entropy	15	69%	59%
Gini	5	70%	63%
Gini	10	69%	62%
Gini	15	67%	58%

B. Linear Regression: Predicting how much Precipitation

1. Summary of Approach

Linear Regression (LR) modeling relies upon an iterative process of minimizing error to produce a set of weight coefficients for each input parameter. In general, most models follow the same basic process with various optimizations for speed and accuracy of the generated model. As the output of each of these models is an array of parameter weight coefficients, the speed of prediction across these models is constant. To modify the performance of output model accuracy, the selection of features and the modeling algorithm are the two most effective means of improvement. The data preparation section above was done with this modeling in mind. In addition, input data normalization was modified further to optimize for LR modeling with a final range of -1 to 1 used for all features. The target feature, NEXTPRECIPAMT, was left un-normalized for a more interpretable output.

Apart from model selection, there are a variety of parameters such as normalization biases, data formatting flags, and underlying algorithm configurations. As a step in the initial model investigation these parameters were modified in varying combinations to gain understanding of how these values affected the model performance. For both LinearRegression and RidgeCV the alpha parameter was increased and decreased proportionally to the data range. In the end, allowing the model to automatically generate values resulted in a lower MSE and higher R squared score. Other values such as forcing positive weights and algorithm selection had similar results and were correspondingly kept at their default ‘automatic’ values.

To test actual model performance, the processed daily readings of NOAA weather data were broken up into a training set and a test set with a 70-30 split using the SciKit Learn train test split function. A constant seed of 919 was used with random shuffling of data to create reproducible yet random sampling. This reproducibility enabled clear analysis of performance improvements as the models were altered. The 70% chunk was used for model training while the 30% block was reserved for model validation. All results in the coming sections are describing the performance of modeling on the test set.
For ground truth analysis of model performance, the Mean Squared Error (MSE) was used. The MSE takes the error, or difference between known test target values and predicted values, of all test entries and returns the mean of the sum of the square of each individual value. What this provides is a single value detailing the typical error for a prediction in a manner that drastically magnifies the importance of large individual errors.

With the algorithm parameterization and data ready for ingestion, the LinearRegression and RidgeCV models were implemented in python. The source code for these models is shown in the Appendices (*E. Python Code for Data Quality Analysis)* at the end of this report. The first step in modeling was to take the 31000 daily readings and modeling using the core five elements: amount of rain precipitation (PRCP), amount of snowfall (SNOW), snow depth (SNWD), minimum temperature (TMIN), and maximum temperature (TMAX). After the initial core parameters were analyzed, the entire list of readings or elements was used and compared to the core five. With both versions generated using LinearRegression and RidgeCV, the performance metrics were saved off for analysis.

2. Discussion and Observations of Results

For model analysis, the Linear Regression modeling was examined at three major milestones. First, the performance of the five core features was used as a baseline of a low complexity model’s performance. Table LR1 below shows the output of this stage. Following the core five, all elements were used as predictors for the models and the results of both LinearRegression and RidgeCV modeling libraries are shown in tables LR2 and LR3. Given the results and coefficient weights detailed in these tables, individual parameters were examined univariately against the target feature to gain a deeper understanding of model accuracy and correlation.

From table LR-1 shown below, we can see that all features were on the same order of magnitude, with the current day’s precipitation being the largest weight factor in predicting the next day’s amount. Apart from weights, the mean squared error of 12874 seems high, but reduced to a mean error of 1.134 cm which is within a decent margin of error. It can also be noted that the R squared score is extremely low indicating minimal correlation between the input features and the target on a linear basis. The following tables show the same results using all available NOAA readings.

MANUSCRIPT ID: 000000

Table LR1
Model performance and weight coefficients for Core Five elements in LinearRegression modeling.

Model Parameter	Value
MSE	12874.6581
R2 Score	0.035869927
Intercept	153.4313987
W_PRCP	127.5445139
W_SNOW	-12.33708721
W_SNWD	12.78584721
W_TMIN	68.62577116
W_TMAX	-50.24495686

The next two tables (LR2 and LR3) immediately reveal a few key takeaways. For starters, our mean squared error decreased by only 1.1% while the R squared score for both models increased by about 30%. As a result, the model performance improved marginally from 1.134 cm to 1.126 cm mean prediction error, but the overall model correlation between features and the target improved a noticeable amount. Beyond model accuracy, both models performed nearly identically with RidgeCV being only slightly more accurate for the given test set. The final takeaways is that although model performance was similar between the two libraries, the selected weights prioritized features differently. Knowing this it can be remarked that temperature of the given day, precipitation on the day prior to the prediction, and local weather conditions were commonly prioritized in all cases.

Table LR2
Model performance of LinearRegression with all ELEMENT values present for a single day prediction.

Model Parameter	Value
MSE	12725.58274
R2 Score	0.047033566
Intercept	-275.251091
W_WT11 (Thunder)	49.67996756
W_TMIN	47.33627611
W_SN12 (Min Soil Temp – Grass 10cm)	40.71846929
…	…
W_WT22 (Ice Fog)	0
…	…
W_SX12 (Max Soil Temp – Grass 10cm)	-52.01192715
W_SNOW	-298.9480694
W_PRCP	-66600.376

Table LR3
Model performance of RidgeCV with all ELEMENT values present for a single day prediction.

Model Parameter	Value
MSE	12722.57
R2 Score	0.047259
Intercept	-275.251091
W_PRCP	48.37254428
W_TMIN	46.46172943
W_WT11 (Thunder)	43.89643219
…	…
W_WT15 (Freezing Drizzle)	0
…	…
W_SX12 (Max Soil Temp – Grass 10cm)	-30.06173414
W_WSF1 (Fastest Wind Speed 1-minute)	-35.18791546
W_WSFI (Fastest Wind Speed Direction)	-40.71263185

MANUSCRIPT ID: 000000

Images LR-1 through LR-4 below show the normalized, univariate relationship between input model features and the target feature NEXTPRECIPAMT. LR-1 through LR-3 represent the largest weight magnitude with LR-4 representing a feature with zero weight in the model output.

Image LR-1
Scatter plot of cumulative precipitation (PRCP) reading vs. Next Day's Precipitation Amount

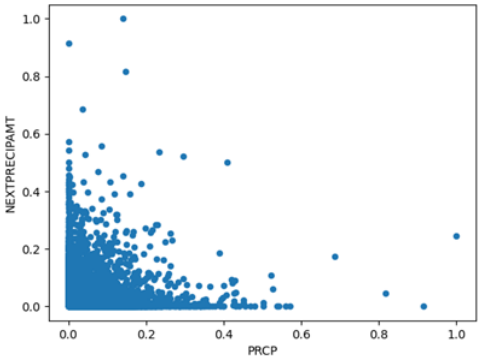


Image LR-2
Scatter plot of cumulative snowfall (SNOW) reading vs. Next Day's Precipitation Amount

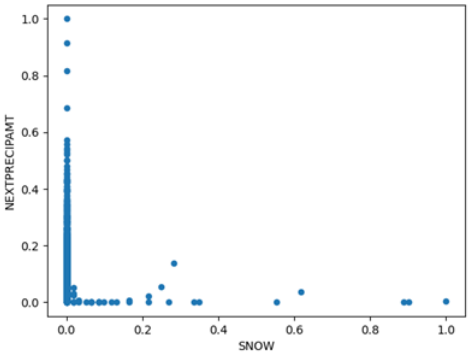


Image LR-3
Scatter plot of maximum soil temperature for grass covered ground at 10cm (SX12) reading vs. Next Day's Precipitation Amount

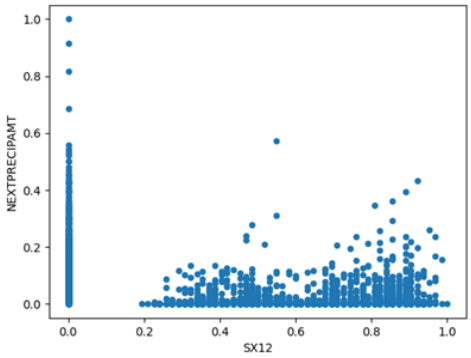
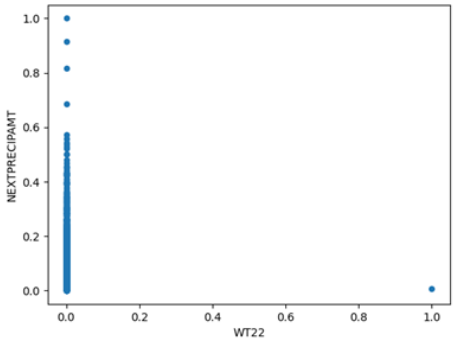


Image LR-4
Scatter plot of ice fog present at site (WT22) reading vs. Next Day's Precipitation Amount



After comparing the two SciKit Learn linear regression models, the next step was to analyze why certain variables may have been selected. The five images above show some common examples of relationships between features and the target variable NEXTPRECIPAMT. It should be noted that both the feature and the target were min-max scaled between 0 and 1 to make the plots more clear for this report.

It can be observed that all distributions are skewed right with larger reading being far less common. Values with a more even distribution across their min-max range tended to have a larger impact on the output, while those like WT22 with minimal presence in the data had little to no impact. For all variables, it can be seen that a general negative trend in the precipitation amount as the data goes from minimum to maximum values. Overall, the lack of a clear linear relationship for any variables gives a justification for such a low R squared scores with these models.

Overall, the use of a single day's weather readings to predict the next day's precipitation amount is at best a rough estimate with little to no quantitative benefit from a statistical standpoint. The R squared scores of both modeling libraries strongly indicate the lack of a linear relationship between features and the target variable. In general, the more data available, the better the models performed. With that said, the lack of exponential improvement based on data formatting, imputation, or feature selection suggest that the NOAA daily readings for a single location does not provide enough information to accurately predict precipitation for that region one day in advance. It is likely that considering weather measurements from multiple days in the past from multiple locations will yield a more accurate linear regression model for the next day.

Figure MDC-1
Entropy model for 2 days data was considered where 5 parameters were considered

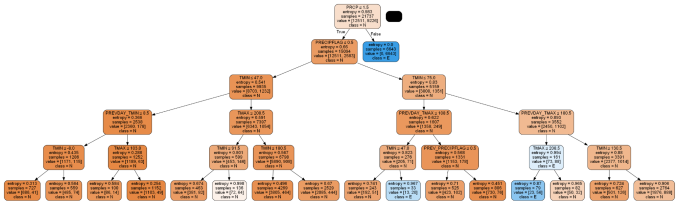
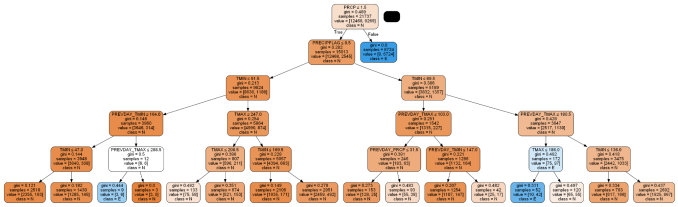


Figure MDC-2
Gini for 2 days where 5 parameters of data were considered



C. Influence of Multiple Days when Training Models

1. Summary of Approach

After a first round of models were created that only considered the current day's weather; new models were created that also considered the weather measurements from the previous day to see if model prediction accuracy could be improved. For this model change, the dataset used to train each model was changed so that the previous day and the current day's data measurements were considered to determine the precipitation flag. To accommodate this change; the dataset created from the 'Data Processing' section earlier in this report was changed. In each data entry, a new feature column was appended that represented each type of weather measurement from the previous day. Therefore, when the models review each data entry for training, there are features for the current day's weather and the weather events from the previous day.

The excel sheet was updated where five parameters were considered for the previous two days. There were two datasets used to train different sets of models. The first dataset used the 'core 5' parameters: Max Temperature (TMAX), Minimum Temperature (TMIN), Amount of Rain Precipitation (PRCP), Amount of Snowfall (SNOW), and Snow Depth (SNWD). The second dataset of two day data used all of the available parameters to predict if any precipitation will occur on the next day (NEXTPRECIPFLAG).

2. Classification Model Two Day Prediction

Below in Figure MDC-1 and Figure MDC-2 are two decision tree classification models that take into account two days worth of information instead of one. The dataset used to create these trees are based on the extra data preparation step described in the previous section. In these models, the decision could use either a feature from the current day or the previous day.

When the classification models trained with two days of weather measurements, the model's accuracy increased compared to when the model trained with one. As the amount of data considered by the decision tree increased, so did the model's accuracy. As seen in table MDC-6 below, the accuracy of the two day models were nearly 20% higher than the one day trained models. The training score increases as the depth of the tree increases whereas the testing score or accuracy decreases with the tree depth.

The MDC-3 table below shows how well the model predicts the next day's precipitation flag. The decision classifier's accuracy is based on if the value predicted is the actual training dataset. Since the True Positive & Negative percentages are above 80%, it is possible to conclude that as the data and depth of the tree increases, the classification model predicts with increased accuracy.

Table MDC-3
Classification model accuracy for two-day data of the core five readings

Classifier	Depth	True Negative Accuracy that NEXTPERCIPFLAG will be 0	True Positive Accuracy that NEXTPERCIPFLAG will be 1
Entropy	5	83%	100%
Entropy	10	84%	94%
Entropy	15	85%	87%
Gini	5	84%	100%
Gini	10	85%	94%
Gini	15	86%	87%

Table MDC-4
Classification model accuracy for two-day data of the core five readings

Classifier	Depth	True Negative Accuracy that NEXTPERCIPFLAG will be 0	True Positive Accuracy that NEXTPERCIPFLAG will be 1
Entropy	5	96%	99%
Entropy	10	96%	99%
Entropy	15	97%	98%
Gini	5	96%	99%
Gini	10	96%	99%
Gini	15	96%	98%

When modeling with all data elements, the resulting training scores showed slightly higher accuracy than the ‘core five’ models at depths of ten and fifteen. The training score for a model depth of five was slightly lower than the ‘core five’ accuracy, but both showed nearly perfect fit to the training data. Table MDC-5 shows a 30% increase in model accuracy on the training data for models using two days worth of readings as compared to a single day.

Table MDC-6, however, shows that testing scores fall nearly 12% short of training scores with larger deficits as tree depth increases. This discrepancy suggests a clear case of overfitting as a result of the much larger feature list in the training dataset. The additional data resulted in more consistent true-negative and true-positive readings at all depths, suggesting that the elements contained relevant predictors. Given the 100% accuracy of ‘core five’ classifications at a depth of five, the information gained from the larger set of readings is outweighed by the negative impact of overfitting to the larger training data.

Decision tree is very sensitive to the data. If a single row is changed then the accuracy will change. This sensitivity occurs because the decision tree tries to recreate every individual test case in the training data (also known as overfitting). This overfitting is reflected in the near 100% training test score seen in table MDC-5. As seen in table MDC-6, the model with less data has a significantly lower testing score than the model that considers two days worth of measurements. By adding an extra day, the accuracy has drastically increased from 65% to 85% approximately.

Table MDC-5
Difference in Model Training Performance based on Days Trained

Classifier	Tree Depth	Single Day Training score (accuracy)	Two Day Training score (accuracy)
Entropy	5	67.88%	97.51%
Entropy	10	72.20%	97.88%
Entropy	15	79.54%	98.68%
Gini	5	67.91%	97.59%
Gini	10	73.14%	98.15%
Gini	15	81.23%	98.92%

65 (all parameters) Element Types were considered for each model.

MANUSCRIPT ID: 000000

Table MDC-6
Difference in Model Test Performance based on Days Trained

Classifier	Tree Depth	Single Day Testing score (accuracy)	Two Day Testing score (accuracy)
Entropy	5	67.18%	88.35%
Entropy	10	66.49%	88.14%
Entropy	15	65.8%	86.92%
Gini	5	67.15%	87.95%
Gini	10	65.86%	88.14%
Gini	15	63.75%	85.23%

65 (all parameters) Element Types were considered for each model.

3. Linear Regression Two Day Prediction

For the linear regression modeling of the two previous days' data, the approach of using all available data points was followed. For each column in the single day data, a second column was added as the previous day's recording for that feature. This resulted in a data set that was about twice the size of the previous entry. Model training performance with SciKit Learn's linear regression tool sets remained the same with only a 5% increase in processing time. The table below details the high-level changes in accuracy for the LinearRegression and RidgeCV libraries as compared to their one-day counterparts.

MDL-1
Two-day model accuracy compared to previous single day values

Parameter	**1-day LR**	**1-day RidgeCV**	**2-day LR**	**2-day RidgeCV**
MSE	12725.58274	12722.57	11427.21243	11425.96855
R2 Score	0.047033566	0.047259	0.04799800331	0.04810163103

From the values shown above in MDL-1, it becomes obvious that more historical data has a far greater impact on model performance than the inclusion of any single feature variable. While the R squared scores for both models saw only a slight increase, the Mean Squared Error (MSE) improved by over 10% for both SciKit Learn libraries. These findings also further the assessment that RidgeCV typically outperforms the base LinearRegression model by a small margin for this data set. Experimentation with different seeds for data partitioning also supported this finding. The average error of these models dropped from about 1.13cm to 1.06cm resulting from the 10.1% increase in MSE.

It should be noted that the MSE from these models still lacks enough consistency for use in reliable precipitation prediction. The real value gained from this multi-day analysis is the implication that sequences of weather spanning multiple days greatly increases model accuracy. Future improvements may involve iteratively increasing the window of dates used in calculations in order to determine the optimal time window for modeling local weather trends. The final note to make about these models is that linear regression algorithms can perform poorly when feature variables are heavily correlated. Seeing as many columns, such as weather type and precipitation amount, are heavily correlated or codependent, follow-on investigation should also examine correlation and removal of features in relation to each other.

VI. Conclusion

Predicting the weather is something that researchers have been attempting to perfect for centuries by using a variety of tools and techniques. Supervised machine learning led classification and linear regression models have a place when determining if any precipitation will occur and how much. This report provides an example of how to prepare a large dataset into individual features for these models to efficiently process into meaningful predictions. The created classification model details that the amount of training data used makes a significant difference in how well it predicts if any precipitation event occurs. When all of the element parameters were considered in the model, the testing accuracy rose 20%. However, predicting the weather is an imperfect science. The linear regression model outlines that increasing the amount of used training data does not guarantee that the model's accuracy (R squared score) will also increase.

The combination of the classification and linear regression models describe how important it is to plan what a model will predict. For a boolean, yes or no question, more data increases the performance of classification models. However, open ended similarity models might suffer from larger low quality datasets. Quality and not quantity of data may make or break a model. With more varied data and data from different sources, these models could become more accurate for the Charleston area. The next researcher can take these models and provide more data from different weather stations to help increase the prediction's accuracy. In conclusion; these supervised machine learning models can help advise if precipitation will or will not occur the next day, but they cannot accurately predict the amount of precipitation that may occur if any does. Always bring an umbrella.

MANUSCRIPT ID: 000000

APPENDIX

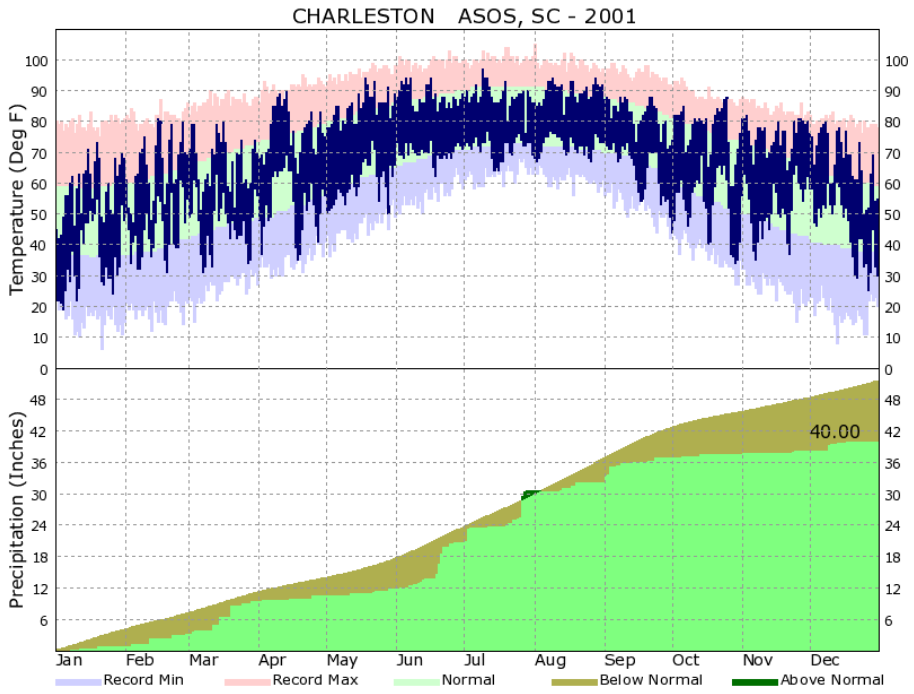
A. Description of Element Field Names from original Dataset

Below is a reference to the direct website of NOAA documentation created by M.J. Menne and their collaborators. The documentation provides a list of the potential data samples that the USW00013880 station is capable of measuring.

Please see NOAA website: https://www.ncei.noaa.gov/products/land-based-station/global-historical-climatology-network-daily

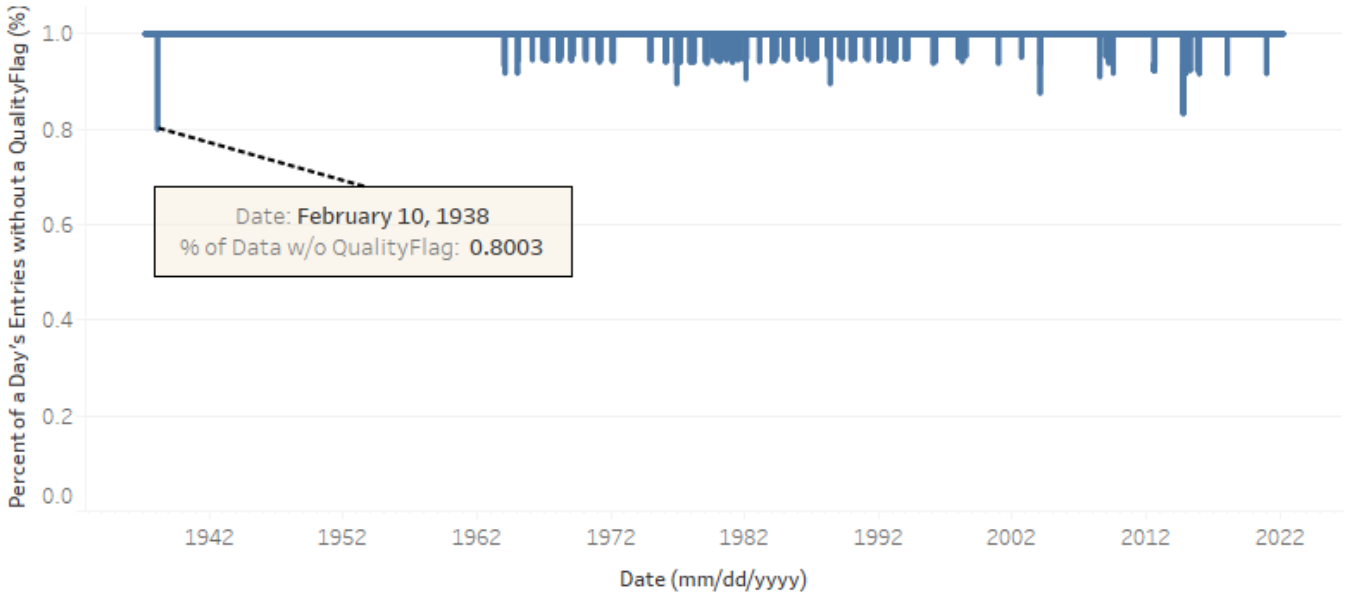
Figure SD2

Average Temperature in Charleston during 2001 [2]



B. Data Processing: Justifications to Remove all Data Entries with a Quality Flag

Figure DP2
Descriptive statistics of each day's data quality efficiency



*(how many data entries do not have a QualityFlag).
**The day with the lowest quality efficiency is 02/10/1938 with ~80%.

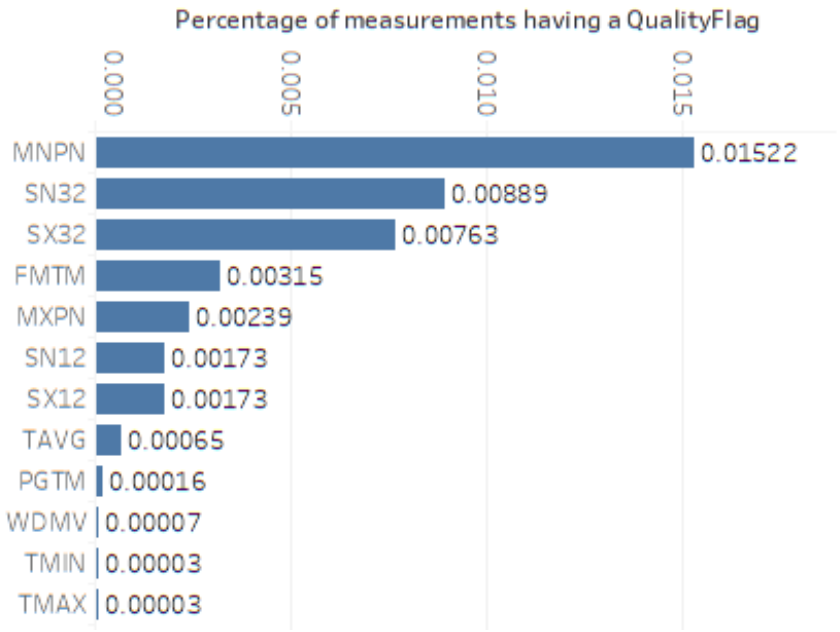
TABLE DP3
CONCENTRATION OF DATA ENTRIES WITH QUALITY FLAGS BY SOURCE.

SourceFlag	**InstancesOfQualityFlag / InstancesOfSourceFlag**	**Percentage (%)**
(0) U.S. Cooperative Summary of the Day (NCDC DSI-3200)	232/211231	0.1098%
(7) U.S. Cooperative Summary of the Day – Transmitted via WxCoder3 (NCDC SI-3207)	52/6296	0.8259%
(B) U.S. ASOS data for October 2000-December 2005 (NCDC DSI-3211)	0/2	0.00%
(D) Short time delay US National Weather Service CF6 daily summaries provided by the High Plains Regional Climate Center	0/292	0.00%
(K) U.S. Cooperative Summary of the Day data digitized from paper observer forms (from 2011 to present)	0/730	0.00%
(S) Global Summary of the Day (NCDC DSI-9618)	0/422	0.00%
(W) WBAN/ASOS Summary of the Day from NCDC's Integrated Surface Data (ISD).	2/46219	0.0043%
(X) U.S. First-Order Summary of the Day (NCDC DSI-3210)	35/172235	0.0203
(Z) Datzilla official additions or replacements	2/6254	0.032%

MANUSCRIPT ID: 000000

TABLE DP3

CONCENTRATION OF DATA ENTRIES WITH QUALITY FLAGS BY ELEMENT MEASURED*



*ANY DATA ELEMENT NOT SHOWN HAS A CONCENTRATION OF 0%

FIGURE DP5

RESULTING STRUCTURE OF A DATA ENTRY IN THE DATASET USED TO TRAIN EACH MODEL

Date	#Events	Element Feature1	Element Feature2	…	PRECIP FLAG	PRECIPAMT	NEXTPRECIP FLAG	NEXTPRECIPAMT

*THE '...' ELLIPSE REPRESENTS THE REMAINING ELEMENT FEATURES.

C. Post-Processing Data Quality Report

Table DQR2-b
Data Quality Report of the USW00013880 data after all modifications

stat	TMAX Max Temperature (+/- 0.1*C)	TMIN MinTemperature (+/- 0.1*C)	PRCP Precipitation (+/- 0.1 mm)	SNOW Snow Fall (+/- 0.1 mm)	SNWD Snow Depth (+/- 0.1 mm)
count	31054	31054	31054	31054	31054
cardinality	81	91	375	21	12
mean	244.7074129	129.7407741	35.90561602	0.03648483287	0.05751271978
median	256	139	0	0	0
number at median	736	563	21490	31018	31028
mode	311	222	0	0	0
number at mode	1142	1275	21490	31018	31028
stddev	73.85014509	82.78636855	109.5762633	1.721127368	2.44649887
min	-67	-144	0	0	0
max	406	283	2921	152	203
number of zeros	7	389	21490	31018	31028
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-c
Data Quality Report of the USW00013880 data after all modifications

stat	WT05 Weather Type (Hail)	WT08 Weather Type (smoke/haze)	WT16 Weather Type (Rain)	WT01 Weather Type (Fog)	WT03 Weather Type (Thunder)
count	31054	31054	31054	31054	31054
cardinality	2	2	2	2	2
mean	0.01236555677	0.2648612095	0.338152895	0.4261286791	0.1408836221
median	0	0	0	0	0
number at median	30670	22829	20553	17821	26679
mode	0	0	0	0	0
number at mode	30670	22829	20553	17821	26679
stddev	0.1105126376	0.4412663815	0.4730884926	0.4945208822	0.3479070635
min	0	0	0	0	0
max	1	1	1	1	1
number of zeros	30670	22829	20553	17821	26679
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-d
Data Quality Report of the USW00013880 data after all modifications

stat	WT04 Weather Type (Ice Pellets)	WSFG Highest instantaneous wind speed (+/- 0.1 m/s)	WT18 Weather Type (Snow Pellets)	PGTM Peak gust time (hours & minutes)	WDFG Direction of peak wind gust (degrees)
count	31054	31054	31054	31054	31054
cardinality	2	58	2	1326	17
mean	0.002093128099	43.10736137	0.003316802988	784.8084949	86.4885361
median	0	0	0	936	0
number at median	30989	17804	30951	43	17813
mode	0	0	0	0	0
number at mode	30989	17804	30951	12815	17813
stddev	0.04570354667	54.1030015	0.0574970283	742.452279	119.4870014
min	0	0	0	0	0
max	1	437	1	2359	360
number of zeros	30989	17804	30951	12815	17813
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

Table DQR2-e
Data Quality Report of the USW00013880 data after all modifications

stat	WT06 Weather Type (Glaze)	WT07 Weather Type (Dust)	DAEV Number of days included in multiday evaporation	EVAP Evaporation from Pan (+/- 0.1 mm)	MDEV Multiday evaporation total (tenths of mm; use with DAEV)
count	31054	31054	31054	31054	31054
cardinality	2	2	5	106	28
mean	0.001803310363	0.002608359632	0.003187995105	21.47449604	0.06955625684
median	0	0	0	0	0
number at median	30998	30973	31018	17683	31018
mode	0	0	0	0	0
number at mode	30998	30973	31018	17683	31018
stddev	0.04242777866	0.05100627285	0.09970238592	34.40067739	2.435829959
min	0	0	0	0	0
max	1	1	6	907	155
number of zeros	30998	30973	31018	17683	31018
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-f
Data Quality Report of the USW00013880 data after all modifications

stat	WDMV 24-hour wind movement (km)	FRGT Top of frozen ground layer (cm)	THIC Thickness of ice on water (tenths of mm)	DAWM Number of days included in the multiday wind movement (MDWM)	MDWM Multiday wind movement (km)
count	31054	31054	31054	31054	31054
cardinality	244	2	3	5	29
mean	47.14036839	0.01552134991	0.008984349842	0.002704965544	0.2284407806
median	0	0	0	0	0
number at median	16721	31052	31052	31024	31024
mode	0	0	0	0	0
number at mode	16721	31052	31052	31024	31024
stddev	63.65079926	1.934043657	1.448328487	0.1073375969	8.982450931
min	0	0	0	0	0
max	1249	241	254	13	731
number of zeros	16721	31052	31052	31024	31024
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-g
Data Quality Report of the USW00013880 data after all modifications

stat	MNPN Daily minimum temperature of water in an evaporation pan (tenths of degrees C)	MXPN Daily maximum temperature of water in an evaporation pan (tenths of degrees C)	ACMH Average cloudiness midnight to midnight from manual observations (percent)	ACSH Average cloudiness sunrise to sunset from manual observations (percent)	PSUN Daily percent of possible sunshine (percent)
count	31054	31054	31054	31054	31054
cardinality	54	76	11	11	101
mean	58.19965222	105.59535	19.84060024	21.59431957	14.38999807
median	0	0	0	0	0
number at median	18821	18538	20770	20931	24537
mode	0	0	0	0	0
number at mode	18821	18538	20770	20931	24537
stddev	84.96700392	137.6833606	32.5913182	35.40181376	30.43808758
min	0	0	0	0	0
max	300	428	100	100	100
number of zeros	18821	18538	20770	20931	24537
number missing	0	0	0	0	0

All measurements annotated with '*' represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-h
Data Quality Report of the USW00013880 data after all modifications.

stat	TSUN Daily total sunshine (minutes)	WDFI Direction of highest instantaneous wind (degrees)	WSFI Highest instantaneous wind speed (tenths of meters per second)	WT02 Weather Type (Heavy Fog)	WDFM Fastest mile wind direction (degrees)
count	31054	31054	31054	31054	31054
cardinality	830	9	54	2	9
mean	181.5659818	23.88387969	12.46676757	0.04485734527	0.9984221034
median	0	0	0	0	0
number at median	19612	27344	27344	29661	30902
mode	0	0	0	0	0
number at mode	19612	27344	27344	29661	30902
stddev	268.3764171	73.20735497	35.54618352	0.2069940666	15.74528039
min	0	0	0	0	0
max	859	360	317	1	360
number of zeros	19612	27344	27344	29661	30902
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-i
Data Quality Report of the USW00013880 data after all modifications.

stat	WSFM Fastest mile wind speed (tenths of meters per second)	WT09 Weather Type (Drifting snow)	WDF1 Direction of fastest 1-minute wind (degrees)	WSF1 Fastest 1-minute wind speed (tenths of meters per second)	WESD Water equivalent of snow on the ground (tenths of mm)
count	31054	31054	31054	31054	31054
cardinality	27	2	37	39	5
mean	0.4611966252	0.001449088684	43.78018935	16.3023121	0.1472274103
median	0	0	0	0	0
number at median	30902	31009	23689	23689	31050
mode	0	0	0	0	0
number at mode	30902	31009	23689	23689	31050
stddev	6.89383997	0.03803991882	91.6880034	30.68205439	24.65735098
min	0	0	0	0	0
max	197	1	360	232	4343
number of zeros	30902	31009	23689	23689	31050
number missing	0	0	0	0	0

All measurements annotated with '*' represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-j
Data Quality Report of the USW00013880 data after all modifications.

stat	AWND Average daily wind speed (tenths of meters per second)	FMTM Time of fastest mile or fastest 1-minute wind (hours and minutes, i.e., HHMM)	WT14 Weather Type (Drizzle)	WT10 Weather Type (Tornado)	WT15 Weather Type (Freezing drizzle)
count	31054	31054	31054	31054	31054
cardinality	98	1238	2	2	2
mean	15.75426676	458.726251	0.01391125137	9.66E-05	9.66E-05
median	0	0	0	0	0
number at median	17112	20940	30622	31051	31051
mode	0	0	0	0	0
number at mode	17112	20940	30622	31051	31051
stddev	19.4662605	707.2538991	0.1171245927	0.009828514144	0.009828514144
min	0	0	0	0	0
max	132	2359	1	1	1
number of zeros	17112	20940	30622	31051	31051
number missing	0	0	0	0	0

All measurements annotated with '*' represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-k
Data Quality Report of the USW00013880 data after all modifications.

stat	WT17 Weather Type (Freezing Rain)	WDF2 Direction of fastest 2-minute wind (degrees)	WDF5 Direction of fastest 5-second wind (degrees)	WSF2 Fastest 2-minute wind speed (tenths of meters per second)	WSF5 Fastest 5-second wind speed (tenths of meters per second)
count	31054	31054	31054	31054	31054
cardinality	2	37	37	43	60
mean	0.0004186256199	55.37998325	54.53371546	24.98544471	31.59357893
median	0	0	0	0	0
number at median	31041	21405	21443	21405	21442
mode	0	0	0	0	0
number at mode	31041	21405	21443	21405	21442
stddev	0.02045638892	98.77334579	98.30515546	39.39647236	50.23261947
min	0	0	0	0	0
max	1	360	360	233	309
number of zeros	31041	21405	21443	21405	21442
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-l
Data Quality Report of the USW00013880 data after all modifications.

stat	WT19 Weather Type (unknown)	WT13 Weather Type (Mist)	WT11 Weather Type (High or damaging winds)	TAVG Average temperature (tenths of degrees C)	WT21 Weather Type (Ground fog)
count	31054	31054	31054	31054	31054
cardinality	2	2	2	231	2
mean	0.0004508275906	0.1046886069	0.000805049269	19.31548271	0.01246216268
median	0	0	0	0	0
number at median	31040	27803	31029	27956	30667
mode	0	0	0	0	0
number at mode	31040	27803	31029	27956	30667
stddev	0.02122825609	0.3061566933	0.02836242354	62.611341	0.1109380616
min	0	0	0	-22	0
max	1	1	1	339	1
number of zeros	31040	27803	31029	27956	30667
number missing	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

MANUSCRIPT ID: 000000

Table DQR2-m

Data Quality Report of the USW00013880 data after all modifications.

stat	WT22 Weather Type (Ice fog)	WV03 Weather in the Vicinity (Thunder)	SN12 Minimum soil temperature (tenths of degrees C) (Grass - 10cm)	SX12 Maximum soil temperature (tenths of degrees C) (Grass - 10cm	SN32 Minimum soil temperature (tenths of degrees C) (bare ground - 10 cm)	SX32 Maximum soil temperature (tenths of degrees C) (bare ground - 10 cm)
count	31054	31054	31054	31054	31054	31054
cardinality	2	2	49	52	47	54
mean	3.22E-05	0.001191472918	18.03159013	20.67936498	19.49584595	22.60208025
median	0	0	0	0	0	0
number at median	31053	31017	28175	28174	27933	27931
mode	0	0	0	0	0	0
number at mode	31053	31017	28175	28174	27933	27931
stddev	0.005674678031	0.03449770476	60.07676098	67.83827513	61.86343219	70.66581782
min	0	0	0	0	0	0
max	1	1	306	344	294	367
number of zeros	31053	31017	28175	28174	27933	27931
number missing	0	0	0	0	0	0

All measurements annotated with ‘*’ represents an invalid mathematical operation with a non-numeric feature.

Table DQR2-n

Data Quality Report of the USW00013880 data after all modifications.

stat	Previous Day's Precipitation Flag (1/0)	Previous Day's Precipitation (+/- 0.1 mm)	Previous Day's Precipitation Flag (1/0)	Previous Day's Precipitation (+/- 0.1 mm)	Next Day's Precipitation Flag (1/0)	Next Day's Precipitation (+/- 0.1 mm)
count	31054	31054	31054	31054	31054	31054
cardinality	3	393	2	392	3	393
mean	0.4254339355	35.91124851	0.4254202357	35.9100921	0.4254339355	35.91124851
median	0	0	0	0	0	0
number at median	17842	21488	17843	21489	17842	21488
mode	0	0	0	0	0	0
number at mode	17842	21488	17843	21489	17842	21488
stddev	0.4944165996	109.5835317	0.494414533	109.5819567	0.4944165996	109.5835317
min	0	0	0	0	0	0
max	1	2921	1	2921	1	2921
number of zeros	17842	21488	17843	21489	17842	21488
number missing	1	1	0	0	1	1

All measurements annotated with '*' represents an invalid mathematical operation with a non-numeric feature.

D. Classification - Supporting Figures & Tables

Figure C3
Entropy model with all parameters considered for 1 day data and depth 10

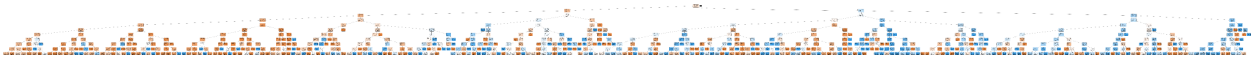


Figure C4
Gini model with all parameters considered for 1 day data and depth 10

Figure C5
Entropy model with 5 parameters considered for 1 day data and depth 10

Figure C6
Gini model with 5 parameters considered for 1 day data and depth 10

Figure MDC-6
Entropy model with all parameters considered for 2 days data and depth 10

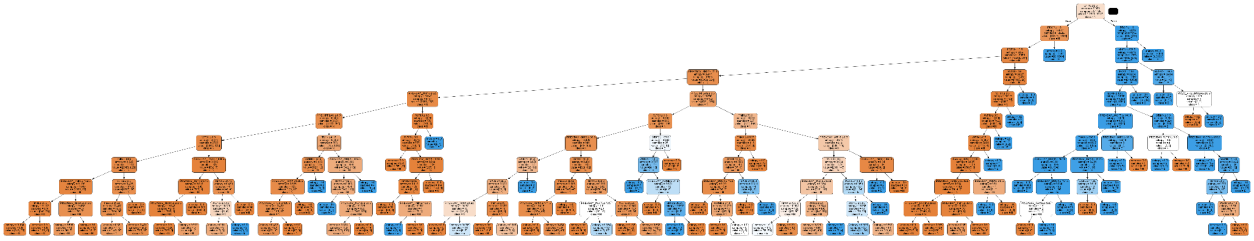
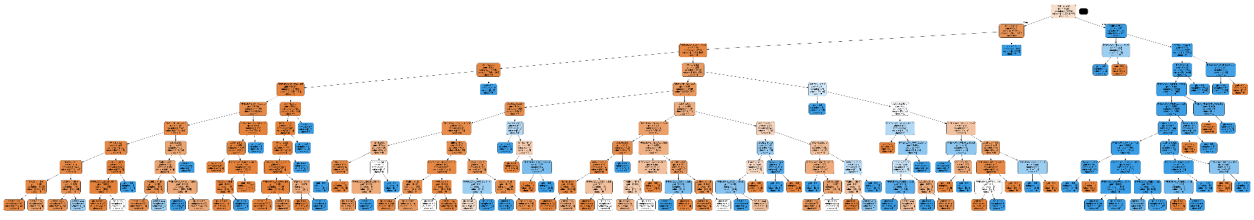


Figure MDC-7
Gini model with all parameters considered for 2 days data and depth 10



MANUSCRIPT ID: 000000

Figure MDC-8
Entropy model with 5 parameters considered for 2 days and depth 10

Figure MDC-9
Gini model with 5 parameters considered for 2 days and depth 10

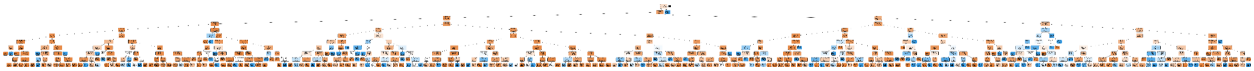


Table MDC-1
Classification model accuracy for two-day data of the core five readings

Classifier	**Depth**	**Training score**	**Testing score or accuracy**	**Root node parameter**
Entropy	5	88.30%	88.60%	PRCP
Entropy	10	90.21%	87.56%	PRCP
Entropy	15	93.04%	86.27%	PRCP
Gini	5	88.45%	88.31%	PRCP
Gini	10	89.91%	88.14%	PRCP
Gini	15	94.3%	85.79%	PRCP

Table MDC-2
Classification model accuracy for two-day data of the core five readings

Classifier	**Depth**	**Training score**	**Testing score or accuracy**	**Root node parameter**
Entropy	5	97.51%	97.66%	WT16
Entropy	10	97.88%	97.1%	WT16
Entropy	15	98.68%	96.85%	WT16
Gini	5	97.59%	97.37%	WT16
Gini	10	98.15%	97.40%	WT 16
Gini	15	98.92%	96.90%	WT 16

E. Python Code for Data Quality Analysis

```
'''
@package    project1.preparedata
@author     Group K
@info       Script for parsing NOAA daily readings summary and generate a
            summary of all possible readings for each day.

            Data Quality Reports are generate for:
               - Raw NOAA data
               - Unmodified daily summaries
               - Cleaned and normalized daily summaries

            Control flags at the top of this file set the format of output
            data and where to collect/deposit inputs and outputs.

            See the README and const.py for more information.
'''

# Python Libraries

# Third Party libraries
import pandas as pd
import numpy as np
from pprint import pprint
import copy
import sys

# Group K project 1 libraries
from lib.dataqualityreport import DataQualityReport
import lib.constants as constants

## Control flags and constants
################################################################################
DEBUG               = False
MODEL_VERSION       = 1
PERCENT_DAYS        = 100
INPUT_FILE          = 'data/USW00013880.csv'
OUTPUT_FILE         = 'processed/USW00013880-SINGLEDAY.csv'

# Validate control flags
_ALLOWABLE_MODELS   = [0, 1, 2]
assert DEBUG in [0, 1, True, False, None], 'DEBUG flag must be a valid true-false value'
assert MODEL_VERSION in _ALLOWABLE_MODELS, 'MODEL_VERSION supports {_ALLOWABLE_MODELS}'
assert PERCENT_DAYS <= 100 or PERCENT_DAYS == None, 'PERCENT_DAYS <= 100 or None'
assert PERCENT_DAYS >= 0 or PERCENT_DAYS == None, 'PERCENT_DAYS >= 0 or None'

# Definitions of feature/column groups
CALCULATED_FEATURES = [
   'PRECIPFLAG',
   'PRECIPAMT'
]

TARGET_VARIABLES = ['NEXTPRECIPFLAG', 'NEXTPRECIPAMT']

## Helper Functions
################################################################################
def wait():
   ''' Function to pause for user-interaction before continuing
```

```
    Disable/Enable with DEBUG flag
    '''
    if DEBUG:
        input()

def summarize(df: pd.DataFrame):
    ''' Print summaries for a dataframe '''
    if not DEBUG:
        return
    print(df.describe())
    print()
    print(df.info())

## Data Handling Functions
################################################################################
def LoadNOAAData(filename: str) -> pd.DataFrame:
    return pd.read_csv(filename, header=None, names=constants.NOAA_DAILY_HEADER)

def cleanRawData(df: pd.DataFrame) -> pd.DataFrame:
    ''' Clean undesired data from the raw NOAA data frame '''
    # 1) Remove all data points that have a non-NULL quality_flag
    df = df[df.quality_flag.isnull()]

    # 1b) TODO - Get indexes of all MeasurementFlag 'T' values?

    # 2) Drop the quality_flag & source_flag column
    df = df.drop(columns=['quality_flag', 'source_flag'])
    return df

## Main Data Processing
################################################################################
print('\nRaw Data Characteristics:\n=========================================')
raw_df = LoadNOAAData(INPUT_FILE)
summarize(raw_df)
wait()

## Create an organized data set summary for the console using a data frame.
report1 = DataQualityReport()
for thisLabel in raw_df.columns:
    thisCol = raw_df[thisLabel]
    if thisLabel == 'element'\
            or thisLabel == 'measurement_flag'\
            or thisLabel == 'quality_flag'\
            or thisLabel == 'source_flag':
        report1.addCatCol(thisLabel, thisCol)
    else:
        report1.addCol(thisLabel, thisCol)
print('\n\nRaw Data DQR - 1/3:\n=====================================================')
print(report1.to_string())
report1.to_csv(OUTPUT_FILE.replace('.csv', '-RAWDQR.csv'))
wait()

## Clean data before generating proceseed data DQR
print('\n\nCleaned Raw Data DQR:\n=====================================================')
clean_df = cleanRawData(raw_df)
summarize(clean_df)
wait()

## Create the processed, unnormalized dataframe of single-day entries
```

```
# Initialize the new dataframe and get the target days
UNIQUE_ELEMENTS = clean_df['element'].unique()
PROCESSED_HEADER = constants.OUTPUT_ID_COLUMNS
PROCESSED_HEADER.extend(UNIQUE_ELEMENTS)
PROCESSED_HEADER.extend(CALCULATED_FEATURES)
PROCESSED_HEADER.extend(TARGET_VARIABLES)
proc_df = pd.DataFrame(columns=PROCESSED_HEADER)
days = clean_df['date'].unique()

# Allow the user to limit data if desired -- really only good for debugging
if PERCENT_DAYS:
    days = days[0-int(PERCENT_DAYS*(days.size/100)):]

## Create the multi-day target dataframe
DOUBLE_COLUMNS = proc_df.drop(columns=['NEXTPRECIPFLAG', 'NEXTPRECIPAMT']).columns.to_list()
two_df = pd.DataFrame(columns=DOUBLE_COLUMNS)
DUB_COL = proc_df.drop(
    columns=['NEXTPRECIPFLAG', 'NEXTPRECIPAMT']
).columns.to_list()
for column in DUB_COL:
    DOUBLE_COLUMNS.append('PREV' + column.upper())
DOUBLE_COLUMNS.extend(['NEXTPRECIPFLAG', 'NEXTPRECIPAMT'])

# Loop through NOAA readings and add values to target output
#   => This will include entries for all elements -- drop(columns) to filter
progress = 0
progress_step = 100 / days.size
print(f'Processing {progress:.2f}%\r', end='')
for didx, day in enumerate(days):
    # Initialize variables for a single day
    events = clean_df.loc[clean_df['date'] == day]
    eventCount = len(events.index)
    if eventCount == 0:
        raise RuntimeError(f'{day} had no applicable events. Please correct data.')
    blankValues = [np.nan] * len(PROCESSED_HEADER)
    entry = dict(zip(PROCESSED_HEADER, blankValues))
    entry['date'] = events.iloc[0]['date']
    datestr = str(entry['date'])
    entry['year'] = int(datestr[:4])
    entry['month'] = int(datestr[4:6])
    entry['day'] = int(datestr[6:8])
    entry['event_count'] = eventCount
    precipFlag = 0
    precipAmount = 0

    # Zero out element readings since they're all numerical
    for key in UNIQUE_ELEMENTS:
        entry[key] = 0

    # Summarize all events from the day into a single entry
    for i in range(0, eventCount):
        # Dissect each event in the current day.
        eidx = events.index[i]
        entryList = events.loc[eidx, :].values.tolist()
        element = entryList[2]

        # Check measurement flag for trace precip
        if clean_df['measurement_flag'][eidx] == 'T':
            precipFlag = 1
```

```
        # SWITCH statement to process the ELEMENT + VALUE
        if (entry[element] not in [0, np.nan]) and (entry[element] != entryList[3]):
            raise ValueError(f'{day} had two values for {element}')
        elif element == 'PRCP':
            entry[element] = entryList[3]
            precipAmount = precipAmount + entryList[3]
        elif element == 'SNOW':
            entry[element] = entryList[3]
            precipAmount = round(precipAmount + entryList[3] / 8)
        elif element == 'SNWD':
            entry[element] = entry[element] + entryList[3]
        elif element == 'EVAP':
             entry[element] = entry[element] + entryList[3]
        else:
            entry[element] = entryList[3]

    # After going through all of the days events; fill in precipitation info.
    if precipAmount > 0:
        precipFlag = 1
    entry['PRECIPFLAG'] = precipFlag
    entry['PRECIPAMT'] = precipAmount

    # Add yesterdays precipitation to today as next_precip...
    if didx > 0:
        df_PrevDay = proc_df.index[proc_df['date'] == days[didx-1]]
        proc_df.at[df_PrevDay[0], 'NEXTPRECIPFLAG'] = precipFlag
        proc_df.at[df_PrevDay[0], 'NEXTPRECIPAMT'] = precipAmount
        two_df.at[df_PrevDay[0], 'NEXTPRECIPFLAG'] = precipFlag
        two_df.at[df_PrevDay[0], 'NEXTPRECIPAMT'] = precipAmount

        # Update dubby boi
        dub_entry = copy.deepcopy(entry)
        for column in DUB_COL:
            dub_entry['PREV'+column.upper()] = proc_df.loc[df_PrevDay[0], column]
        two_entry = pd.DataFrame([dub_entry])
        two_df = pd.concat([two_df, two_entry], ignore_index=True)


    # Convert the day summary from a list too a dataframe.
    df_entry = pd.DataFrame([entry])

    # 5) Add a new TOTAL DAY entry of weather events in a data frame.
    proc_df = pd.concat([proc_df, df_entry],
                                ignore_index=True)

    # Update user with progress
    progress += progress_step
    print(f'Processing {progress:.2f}%\r', end='')

print('Data Processing Complete')
print('====================================================')

## Generate the processed, non-normalized DQR
print(f'\n\nGenerating processed, unnormalized DQR')
print('====================================================')
print(f'Processing {progress:.2f}%\r', end='')
progress = 0
progress_step = 100 / len(PROCESSED_HEADER)
report2 = DataQualityReport()
for thisLabel in PROCESSED_HEADER:
```

```
    report2.addCol(thisLabel, proc_df[thisLabel])
    progress += progress_step
    print(f'Processing {progress:.2f}%\r', end='')

print('\n\nProcessed Data DQR - 2/3\n==================================================')
print(report2.to_string())
report2.to_csv(OUTPUT_FILE.replace('.csv', '-PROCDQR.csv'))

## Write the processed, unnormalized data to file
print(f'\n\nWriting processed data to {OUTPUT_FILE}')
print('==================================================')
if OUTPUT_FILE not in ['', None]:
    proc_df.to_csv(OUTPUT_FILE, index_label='id')
    two_df.to_csv(OUTPUT_FILE.replace('.csv', '-DUB.csv'), index_label='id')
print('All processing complete!')
print('==================================================')

## Normalize and write to file
NORMALIZED_FEATURES = UNIQUE_ELEMENTS.tolist()
NORMALIZED_FEATURES.extend(CALCULATED_FEATURES)
norm_df = copy.deepcopy(proc_df)
for key in NORMALIZED_FEATURES:
    if norm_df[key].min() == norm_df[key].max():
        continue
    norm_df[key] = (norm_df[key]-norm_df[key].min())/(norm_df[key].max()-norm_df[key].min())

print(f'\n\nWriting normalized data to {OUTPUT_FILE}')
print('==================================================')
if OUTPUT_FILE not in ['', None]:
    norm_df.to_csv(OUTPUT_FILE.replace('.csv', '-NORMAL.csv'), index_label='id')
print('All processing complete!')
print('==================================================')

## Generate the processed, non-normalized DQR
print(f'\n\nGenerating processed, normalized DQR')
print('==================================================')
print(f'Processing {progress:.2f}%\r', end='')
progress = 0
progress_step = 100 / len(PROCESSED_HEADER)
report3 = DataQualityReport()
for thisLabel in PROCESSED_HEADER:
    report3.addCol(thisLabel, proc_df[thisLabel])
    progress += progress_step
    print(f'Processing {progress:.2f}%\r', end='')
print('\n\nNormalized Data DQR - 3/3\n==================================================')
print(report3.to_string())
report3.to_csv(OUTPUT_FILE.replace('.csv', '-NORMDQR.csv'))
```

F. Python Code for Classification

```
#!/usr/bin/env python
# coding: utf-8

from sklearn import tree
import pandas as pd
from sklearn.model_selection import train_test_split
get_ipython().system('pip install graphviz')
get_ipython().system('pip install pydot')
import pydot
import graphviz
import pydotplus
import collections
from IPython.display import Image
import numpy as np

model= "C:/Users/anagh/OneDrive/Desktop/ML/Project.csv"
CRITERION = “entropy”
TREE_DEPTH = 5

dataFrame1= pd.read_csv(model)
X = dataFrame1.drop(["YEAR", "MONTH", "DAY","NEXTPRECIPFLAG"], axis=1)

y1= dataFrame1['NEXTPRECIPFLAG']

X_train, X_test, y1_train, y1_test = train_test_split(X, y1, test_size=0.30)

clf= tree.DecisionTreeClassifier(criterion=CRITERION, random_state= 1123, max_depth= TREE_DEPTH)
clf3= clf.fit(X_train, y1_train)
predictions = clf.predict(X_test)
predictions 1

print("training set score= ", clf.score(X_train, y1_train))
print("Test set score= ", clf.score(X_test, y1_test))

dot_data= tree.export_graphviz(clf, out_file=None,
                               feature_names=X.columns,
                               class_names= 'NEXTPRECIPFLAG',
                               filled=True,
                               rounded=True,
                               special_characters=True)
(graph,) = pydot.graph_from_dot_data(dot_data)
graph.write_png("modelDatatree2.png")
display(graph)

from sklearn.metrics import confusion_matrix

cmatrix = confusion_matrix(y1_test,predictions 1,
labels=y1_test.unique())
pd.DataFrame(cmatrix, index=y1_test.unique(), columns=y1_test.unique())

from sklearn.metrics import classification_report
report = classification_report(y1_test,predictions 1)
print(report)

from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y1_test, predictions 1))
```

G. Python Code for Linear Regression

```
#!/usr/bin/env python3

'''
@package    project1.linreg
@info       Perform sklearn LinearRegression and RidgeCV modeling on processed
            NOAA daily weather data. Currently using data for Charleston, SC
@author     Group K
'''

# Python libraries
import sys

# External libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression, RidgeCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
import matplotlib
matplotlib.use('Qt5Agg')
from matplotlib import pyplot as plt

# Constants
RANDOM_SEED = 919
DATA_FILE = 'processed/USW00013880-SINGLEDAY.csv' if len(sys.argv) < 2 else sys.argv[1]
#DATA_FILE = 'processed/USW00013880-SINGLEDAY-DUB.csv' if len(sys.argv) < 2 else sys.argv[1]
TARGET = ['NEXTPRECIPAMT']

# Load the processed data frame
df = pd.read_csv(DATA_FILE)

# Drop the last row as it will not have a valid NEXTPRECIPAMT
df = df.drop(labels=(len(df.index)-1), axis=0)

# Generate the training and test data split
FEATURES = df.drop(columns=TARGET+[
    'id',
    'date',
    'event_count',
    'PRECIPFLAG',
    'NEXTPRECIPFLAG'
], axis=1).columns
targets = df[TARGET]
features = df[FEATURES]
print(features.describe())

X_train, X_test, Y_train, Y_test = train_test_split(
    features,
    targets,
    test_size=0.3,
    random_state=RANDOM_SEED,
)

# Scale the data
scaler = MinMaxScaler(feature_range=(-1, 1))
```

```
scalertrain = scaler.fit(X_train)
scalertest = scaler.fit(X_test)
xtrain = scalertrain.transform(X_train)
xtest = scalertest.transform(X_test)
ytrain = Y_train.to_numpy()
ytest = Y_test.to_numpy()

# pandas quick DQR
print(X_train.describe())
print(Y_train.describe())
print(pd.DataFrame(xtrain, columns=X_train.columns).describe())

# Run the LinearRegression test
print('\n\nLinearRegression:\n==========================================')
lmodel = LinearRegression()
lmodel = lmodel.fit(xtrain, ytrain)
print(f'R2 Score:\t{lmodel.score(xtrain, ytrain)}')
print(f'W:\t\t{lmodel.intercept_}')
lt_predict = lmodel.predict(xtest)
lcoefficients = pd.concat(
    [
        pd.DataFrame(X_train.columns, columns=['Feature']),
        pd.DataFrame(np.transpose(lmodel.coef_), columns=['Coefficient'])
    ],
    axis = 1,
).sort_values(by=['Coefficient'])
print(f'Coefficients:\n{lcoefficients}')
r2 = lmodel.score(xtest, ytest)
print(f'Prediction Score: {r2}')
mse = mean_squared_error(ytest, lt_predict)
print(f'MSE:\t\t{mse}')

# Write results to CSV
coef = lcoefficients.to_dict()
values = {}
values['W_0'] = lmodel.intercept_[0]
for key in coef['Feature'].keys():
    values[f'W_{coef["Feature"][key].upper()}'] = coef['Coefficient'][key]
values['MSE'] = mse
values['R2'] = r2
results = {'Parameter': list(values.keys()), 'Value': list(values.values())}
results = pd.DataFrame(results, columns=results.keys())
results.to_csv('processed/linear_regression_one_day_results.csv', index_label=None)


#!/usr/bin/env python3

'''
@package    project1.ridge
@info       Perform sklearn LinearRegression and RidgeCV modeling on processed
            NOAA daily weather data. Currently using data for Charleston, SC
@author     Group K
'''

# Python libraries
import sys

# External libraries
import numpy as np
import pandas as pd
```

```
from sklearn.linear_model import LinearRegression, RidgeCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
import matplotlib
matplotlib.use('Qt5Agg')
from matplotlib import pyplot as plt

# Constants
RANDOM_SEED = 919
DATA_FILE = 'processed/USW00013880-SINGLEDAY.csv' if len(sys.argv) < 2 else sys.argv[1]
#DATA_FILE = 'processed/USW00013880-SINGLEDAY-DUB.csv' if len(sys.argv) < 2 else sys.argv[1]
TARGET = ['NEXTPRECIPAMT']

# Load the processed data frame
df = pd.read_csv(DATA_FILE)

# Drop the last row as it will not have a valid NEXTPRECIPAMT
df = df.drop(labels=(len(df.index)-1), axis=0)

# Generate the training and test data split
FEATURES = df.drop(columns=TARGET+[
    'id',
    'date',
    'event_count',
    'PRECIPFLAG',
    'NEXTPRECIPFLAG'
], axis=1).columns
targets = df[TARGET]
features = df[FEATURES]
print(features.describe())

X_train, X_test, Y_train, Y_test = train_test_split(
    features,
    targets,
    test_size=0.3,
    random_state=RANDOM_SEED,
)

# Scale the data
scaler = MinMaxScaler(feature_range=(-1, 1))
scalertrain = scaler.fit(X_train)
scalertest = scaler.fit(X_test)
xtrain = scalertrain.transform(X_train)
xtest = scalertest.transform(X_test)
ytrain = Y_train.to_numpy()
ytest = Y_test.to_numpy()

# pandas quick DQR
print(X_train.describe())
print(Y_train.describe())
print(pd.DataFrame(xtrain, columns=X_train.columns).describe())

# Run the RidgeCV test
print('\n\nRidgeCV:\n=======================================')
lmodel = RidgeCV()
lmodel = lmodel.fit(xtrain, ytrain)
print(f'R2 Score:\t{lmodel.score(xtrain, ytrain)}')
print(f'W:\t\t{lmodel.intercept_}')
lt_predict = lmodel.predict(xtest)
```

```
lcoefficients = pd.concat(
    [
        pd.DataFrame(X_train.columns, columns=['Feature']),
        pd.DataFrame(np.transpose(lmodel.coef_), columns=['Coefficient'])
    ],
    axis = 1,
).sort_values(by=['Coefficient'])
print(f'Coefficients:\n{lcoefficients}')
r2 = lmodel.score(xtest, ytest)
print(f'Prediction Score: {r2}')
mse = mean_squared_error(ytest, lt_predict)
print(f'MSE:\t\t{mse}')

# Write results to CSV
coef = lcoefficients.to_dict()
values = {}
values['W_0'] = lmodel.intercept_[0]
for key in coef['Feature'].keys():
    values[f'W_{coef["Feature"][key].upper()}'] = coef['Coefficient'][key]
values['MSE'] = mse
values['R2'] = r2
results = {'Parameter': list(values.keys()), 'Value': list(values.values())}
results = pd.DataFrame(results, columns=results.keys())
results.to_csv('processed/linear_regression_one_day_results.csv', index_label=None)
```

MANUSCRIPT ID: 000000

Acknowledgment

References

[1] Menne, M.J., I.Durre, B. Korzeniewski, S. McNeal, K. Thomas, X. Yin, S. Anthony, R. Ray, R.S. Vose, B.E.Gleason, and T.G. Houston, 2012: Global Historical Climatology Network - Daily (GHCN-Daily), Version 3.26 , USW00013880 e.g. Version 3.12]. NOAA National Climatic Data Center. http://doi.org/10.7289/V5D21VHZ March 22nd 2022.

[2] National Weather Service, Charleston, SC. Weather Forecast Office. (2022, 3 20) Local Climate Data and Plots [Online]. Available: https://www.weather.gov/chs/climate