Predicting Rainfall

ECS 171 Machine Learning

Group 21 Project Report

Malcolm Alvarez (Team Leader), Conar Abramson-Davis, Lakin Smith,
Konark Mangudkar, Lucas Chen, Valerie Sandoval, Raghuram Palaniappan
ECS 171, Spring 2022

May 17, 2022

Repository: github.com

Demo: Link

1 Introduction

Rain is a necessary component to everyday life. It affects a wide range of areas, and without it, many regions of the world would dry up and become inhospitable to live. Because of these impacts, being able to predict rain and precipitation amounts is a very important practice that has been finely tuned within the field of Meteorology, which predicts it by using past data and incoming weather fronts that are analyzed from satellite images and other local measuring devices. However, as we continue to increase our understanding of computers and make them more reliable and faster, we can potentially find more approaches to predicting precipitation. Through machine learning, we can even potentially build more reliable and faster methods of predicting rain and precipitation than we currently have.

Because of the great leaps and strides into developing Machine Learning applications within weather forecasting, many researchers have developed and found Machine Learning applications that result in a high accuracy in weather predictions, giving us an ample number of resources in solving the problem of the weather prediction. This does not mean that these models don't make mistakes, seeing that as global warming effect us, weather changes are making past data less useful within machine learning models. As a result, today's models may become less accurate as time moves on. Our goal is to develop a way of building models that are both reliable and easy to make, so that all potential users need to do is supply their local weather data and be able to use what ever model they create for themselves.

In many ways, predicting precipitation has many impacts on many local and global economies, affecting the following areas for their own specific regions:

- Agriculture
  - Because agriculture is inherently dependent upon water, being able to predict precipitation will allow farmers a chance to gauge potential water amounts for their crops during any particular year.
  - It will also allow agriculture bodies to keep prepare for times of drought if predicted precipitation amounts are under what they need to sustain their business.
- Local Government Agencies
  - Precipitation has a direct impact on the local water ways. As such having a larger amounts of precipitation over a shorter period of time would create an environment susceptible to flooding, which can damage local economies and the people living within those regions
  - A lack of precipitation on the other hand can result in droughts, which can lead to lower yields of crops and may affect local wildlife populations. Government agencies that focus on conservation could potentially gain an upper edge and prepare for droughts before they happen. This can also apply to other agencies like counties which may want to predict coming droughts sooner so that they can prepare their local populations in water conservation policies or begin preparations in stopping them before the droughts are already in motion.

2 Literature Review

As discussed, predicting the amount of rainfall on any given day is a relevant problem for various areas in the world, and as such, many papers have been published regarding this issue.

Through our readings, we discovered multiple techniques used to predict daily rainfall, including Neural Network Regression (NNR), Boosted Decision Tree Regression (BDTR), Bayesian Linear Regression (BLR)[1], Markov chain extended with rainfall prediction (MCRP)[2], historical data matching [3], Multivariable Linear Regression (MLR), and the list continues. Through this we found that some variation of a Neural Network, especially deep learners[4], generally outperformed other techniques, especially when the amount of data and attributes increased[5]. This led us to use an ANN with three layers for our own model.

When trying to predict the amount of rainfall on any given day, rather than just predicting whether it'll rain or not, the variables used to do so increase. The data used and its relevant attributes also varies depending on the location that the data was gathered. Then, depending on what model was being used to predict rainfall, we learned that different researchers concluded different important attributes for the prediction.[5] While we were able to build a base for what attributes we needed to include from our chosen data set, we needed to speak to our own sources in order to gain a better understanding of the data, and learn other important attributes specific to our location.

As well as looking into literature, we also had conversations with experts in the field of weather data gathering and meteorology. Glenn Hetchler, who is the Lead Integrated Systems Engineer for OneRain, and Lindsey Nytes, who is the Technical Sales Support Manager for OneRain, pointed us towards using data collected from METAR stations, an instrument that gather's weather related data, and data collected from radiosondes which are lifted to upper atmosphere levels via weather balloons. These two areas of recording weather data gave us a lot of different variables to look into that related to rainfall. Based on our discussion, as well as some data preprocessing, we found that the following attributes were relevant to rainfall for our chosen location: Temperature, Dew Point, Relative Humidity, Wind Speed, Atmospheric Pressure, Visibility, Cloud Type, and Month.

3 Dataset Description

The data sets we used originate from the Iowa State University's Environmental Mesonet department (link). This data was initially gathered from METAR stations located at the Napa Airport, the Santa Rosa Airport, and the Sacramento Executive Airport. This data was taken over a span of 20 years. Each of these data sets are stored in CSV files. The Sacramento data set had 206097 rows and 11 columns. The Napa data set had 165916 rows and 11 columns. The Santa data set had 173401 rows and 11 columns.

Each dataset included 11 columns representing the following categories: *station*, *valid*, *tmpc*, *dwpc*, *rehl*, *sped*, *alti*, *mslp*, *p01m*, *vsby*, *skyc1*. *station* contains the name of the station that the data came from. *valid* is a string representing the time and date the the row was recorded. *tmpc* is the air temperature in Celsius at that moment in time. *dwpc* is the dew point temperature in Celsius. *rehl* is the relative humidity as a percentage. *sped* is the wind

speed at that time in mph. *alti* is the pressure altimeter in inches. *mspl* is the sea level pressure in millibar. *p01m* is the amount of precipitation in one hour measured in meters. One thing to note about *p01m* is that it isn't just recorded every hour, but it is also recorded every 5 minutes when a change occurred. However, this recording is reset every hour, so if you wanted to find the precipitation in one day, you would have to add the precipitation amount for each hour in that day. *vsby* is the visibility radius around the station in miles. *skyc1* is the cloud coverage in level 1. This is broken up into 8 categories: BKN (broken), CB (cumulonimbus clouds), CLR (clear), FEW (few), OVC (overcast), SCT (scattered), SKC (sky clear), TCU (towering cumulus).

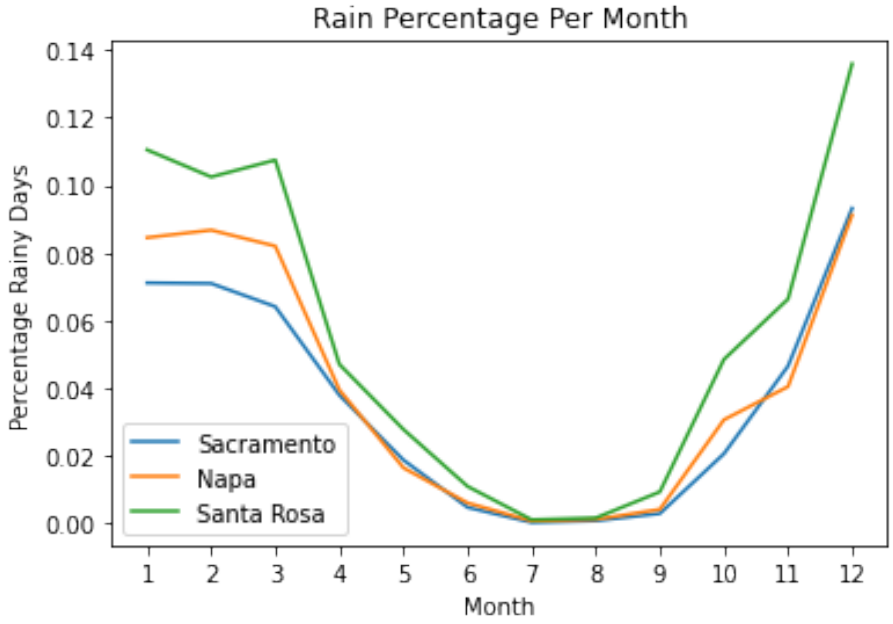

Figure 1: Percentage of Rain Each Month

Based on **Figure1** we can see that between the months of January to April and October to December is the periods that rain the most. One thing to note is that rain only represents 5% of our data.

These data sets are useful for our project because they contain relevant information regarding surface level weather measurements. Because the weather as a whole picture does include precipitation, we can easily use our machine learning model to identify which variables do and don't have significance to precipitation. These data sets will have to be modified to a degree to encompass the output that we expect for our model.

From the above figure, it is clear that while all features are somewhat correlated to rainfall, some are more strongly correlated than others. The strongest correlations overall are relative humidity, visibility, wind speed, altitude, and atmospheric pressure, in order from most correlated to least correlated.

It is important to note that for any given dataset, the altitude will never change. This is because each dataset is a set of chronological weather readings from a single weather station.

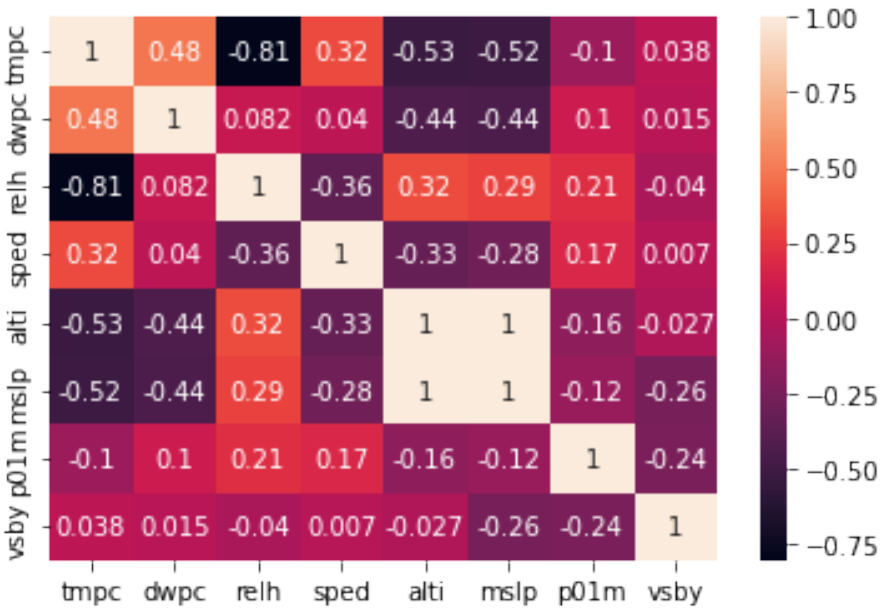

Figure 2: Correlation Matrix (Sacramento Data)

Because weather stations don't move, it is safe to say that the altitude will be static. Because of this, it is safe to omit the altitude feature when training the model.

One thing that we didn't use in our dataset was including measurements that were recorded from the radiosondes devices. Because these devices are only rarely measured (twice a day to be exact), we decided to leave these measurements out of the data we used because of a lack in data for a whole day. This makes it very difficult to incorporate it into the data we got from the METAR stations.

4 Proposed Solution

This problem is a binary classification problem. Because we are only dealing with numbered values after running some prepossessing on our data, we decided to implement a Artificial Neural Network (ANN). This is going to make it easier to predict whether or not it is going to rain on any specific day. In our case, we built a deep neural network with three hidden layers, each containing 12, 10, and 4 number of nodes respectively. We were able to focus on those values by using a Grid search algorithm on the model, though due to time constraints we couldn't do a wide range of values for these hyper parameters.

In terms of prepossessing our data, we went ahead and removed any data that contained values indicated "missing", which were distinguished by the characters 'M' and 'T' in the data set. We also removed columns like 'station' and 'valid' which we knew had no meaning to our model and were just constants anyways. We also scaled our data using a min max scalar so that we could easily pass our data into our neural network.

In preparing our data, we realized that because it only rains on roughly 5 percent of days, any model trained on the raw data alone would be incredibly biased towards classifying any given day as not rainy, because it can achieve a 95 percent accuracy rate by classifying any

day as not rainy. We fixed this by utilizing undersampling by mutating the dataset such that it is composed of 50% rainy days and 50% not rainy days. Undersampling was chosen over oversampling because when using oversampling the increase in accuracy was only around one percent and the time to train was greatly increased. This was done by first splitting the original data into two sets, based on whether or not each day is rainy. Then we recombined the two, while undersampling the majority set such that the result is equally composed of rainy and non rainy observations.

One thing to note is that we implemented a different model for each dataset. The reason for this is because each area that we gathered data from contains a different micro climate, so rainfall in those areas may different, even if they are geographically located in similar areas. For example, Santa Rosa, because of its relative closeness to the coast, experiences more rain than Sacramento, even if they are only about 80 miles apart. So creating models that focus on a particular area is going to give better results than just clumping all of these areas together into one model.

5 Experiment Results

5.1 Hyperparameter Tuning

In designing our neural network, we had to do some form of hyperparameter tuning in order to decide which hyperparameters would produce the most efficient and accurate model. We decided on conducting a grid search. The decision to use grid search over random search was made because we were not testing a wide range of different hyperparameter values. Therefore, random search, while faster, could possibly give suboptimal results. For the purposes of hyperparameter tuning, we elected to keep the number of hidden layers constant at 3. Then, we varied the number of nodes in each layer to find the optimal arrangement of nodes within hidden layers. Our grid search utilized 3-fold cross validation.

Layer 1	Layer 2	Layer 3
12	8	4
15	9	5
18	10	6

Table 1: Grid Search Parameters

After running the grid search, we wrote the results of every different model trained by the grid search into a file. The combination of hyperparameters that yielded the highest score were layer 1 containing 12 nodes, layer 2 containing 10 nodes, and layer 3 containing 4 nodes. The accuracy score of the model trained on these hyperparameters was roughly 0.9254. Below is a bar graph containing the top 5 models with the highest accuracy ratings.

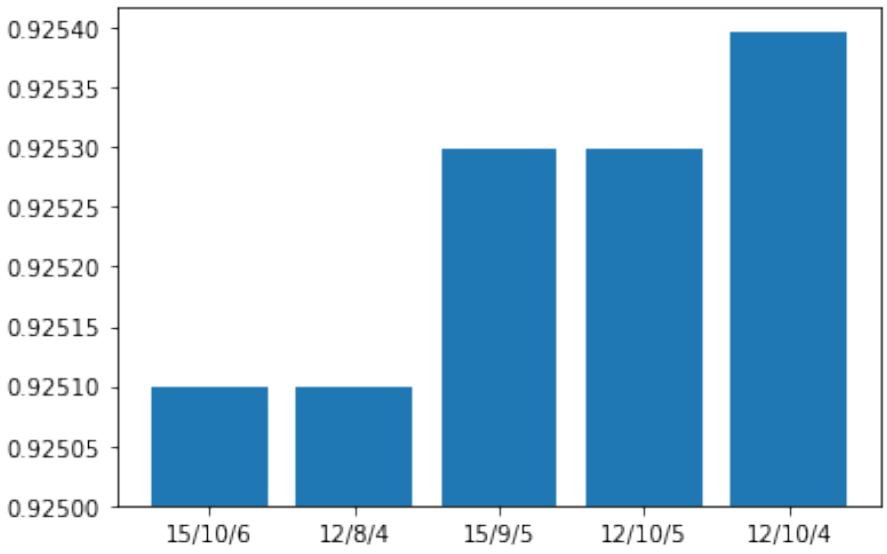

Figure 3: Five Top Layer Configurations

5.2 Model Design

The final model design that we decided on is a neural network with three hidden layers, containing 12, 10, and 4 nodes respectively. The dimensions of the hidden layers were discovered through the aforementioned grid search. The hidden layers use the relu activation function, while the output layer uses the sigmoid activation function. The relu activation function was chosen as the activation function for the hidden layers because it is common practice to do so, and it decreases the training time. We chose to use adam as our optimizer, which is an implemenation of stochastic gradient descent. The difference between adam and normal sgd is that adam utilizes momentum to converge more quickly. The chosen number of epochs was 400, and it was sufficient as the model converged quickly, and numbers of epochs higher than 400 gave diminishing returns.

The inputs, or features that we decided to use for our model are temperature (*tmpc*), dew point (*dwpc*), relative humidity (*relh*), wind speed (*sped*), atmospheric pressure (*mslp*), visibility (*vsby*), cloud class (*skyc1*), and month (parsed from *valid*). Once again, altitude (*alti*) was omitted because, when training a model on a dataset from a single station, the altitude will remain constant, making it useless when it comes to prediction. Because we are training a single model per area, altitude is not useful. We opted to feed in the month as a number from 1 to 12 because, as can be seen in 1, month is correlated to rainfall.

5.3 Frontend Design

The frontend for our model is a web app built using the ReactJS framework. It is a simple demonstration of using our pre-trained models to classify rainfall based on other weather conditions. First, it asks the user to provide the required inputs to the model in their browser. The web app queries the user for the temperature, dew point, atmospheric pressure, relative humidity, wind speed, visibility, cloud classification, and month. After

collecting the required model inputs, the user is required to choose which model to run. The user has a choice of either the Sacramento, Napa, or Santa Rosa model. Depending on which one the user selects, the inputs will be passed to the corresponding model, and the outputs may be different depending on the observed weather patterns in each city. After selecting the desired model, the web app makes a request back to the server, which is running an ExpressJS server. The server then passes the model inputs to a python script which runs the model. The server then waits for the python script to finish execution, and subsequently sends the prediction back to the web app in an http response. On receiving the response, the web app displays the result on the screen, completing the interaction. The repo containing the frontend, backend, and model source code is located here, on github.

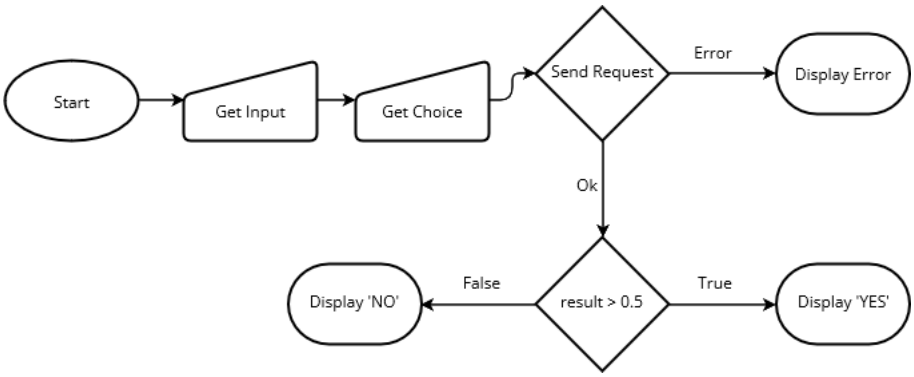

Figure 4: Frontend Flow Chart

5.4 Model Accuracy

Out of the three models that we trained, those being trained on data from Sacramento, Napa, and Santa Rosa respectively, all exhibited accuracy scores of at least 0.92 on the test dataset. The data was split 80:20 into training and testing data after having the aforementioned preprocessing, scaling, and undersampling applied to it.

6 Conclusion and Discussion

Using the data we gathered from the METAR stations in Santa Rosa, Napa, and Sacramento, we were able to build models with an accuracy of at least 92% using an Artificial Neural Network. These datasets contained eleven different parameters, and we applied some data preprocessing before applying the data to the model.

One thing to note is that our model may be still be inaccurate because we are missing data from upper atmosphere conditions. Because the data we had access to for upper atmosphere conditions was rather lacking, we decided to exclude it, but if upper atmosphere conditions were measured more frequently, then it may be possible that our model could contain rather different results.

	precision	recall	f1-score	support
False	0.95	0.90	0.93	1266
True	0.91	0.96	0.93	1254
accuracy			0.93	2520
macro avg	0.93	0.93	0.93	2520
weighted avg	0.93	0.93	0.93	2520

(a) Sacramento

	precision	recall	f1-score	support
False	0.92	0.89	0.91	1021
True	0.90	0.92	0.91	1045
accuracy			0.91	2066
macro avg	0.91	0.91	0.91	2066
weighted avg	0.91	0.91	0.91	2066

(b) Napa

	precision	recall	f1-score	support
False	0.92	0.86	0.89	1437
True	0.86	0.93	0.89	1427
accuracy			0.89	2864
macro avg	0.89	0.89	0.89	2864
weighted avg	0.89	0.89	0.89	2864

(c) Santa Rosa

Figure 5: Classification Reports

Another issue with our model is that it won't be able to be translated into other geographical regions. Even though the idea behind this project is to create models that agricultural centers can use to predict times of rain, different environmental factors and geographical location greatly affects what patterns weather has. However, in theory, different agricultural areas can potentially us the code we used to build their own models.

Later implementations of this model will probably move towards condensing all the models into one overall model that uses geographical location as another parameter. Because of the lack of time and hardware we had available, we couldn't implement this now, but given time and better hardware to create the model, a more abstract model can be built. Later implementations may also lean towards predicting precipitation amounts, which will be more useful towards parties who use the model as it will allow for more important issues like flooding predictions and water replenishment in regions that experience more devastating weather phenomena directly correlated to rain.

7 References

7.1 Literature

References

[1] W. M. Ridwan, M. Sapitang, A. Aziz, K. F. Kushiar, A. N. Ahmed, and A. El-Shafie, "Rainfall forecasting model using machine learning methods: Case study terengganu, malaysia," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 1651–1662, 2021. DOI: https://doi.org/10.1016/j.asej.2020.09.011.

[2] S. Cramer, M. Kampouridis, A. A. Freitas, and A. K. Alexandridis, "An extensive evaluation of seven machine learning methods for rainfall prediction in weather derivatives," *Expert Systems with Applications*, vol. 85, pp. 169–181, 2017. DOI: https://doi.org/10.1016/j.eswa.2017.05.029.

[3] S. Kachwala, M. Jha, D. Shah, U. Shinde, and H. N. Bhor, "Predicting rainfall from historical data trends," *SSRN Electronic Journal*, 2020. DOI: https://doi.org/10.2139/ssrn.3571738.

[4] D. Endalie, G. Haile, and W. Taye, "Deep learning model for daily rainfall prediction: Case study of jimma, ethiopia," *Water Supply*, vol. 22, no. 3, pp. 3448–3461, 2021. DOI: https://doi.org/10.2166/ws.2021.391.

[5] C. M. Liyew and H. A. Melese, "Machine learning techniques to predict daily rainfall amount," *Journal of Big Data*, vol. 8, 2021. DOI: https://doi.org/10.21203/rs.3.rs-801241/v1.

7.2 Dataset Reference

1. METAR station data - link
2. Radiosondes stations data - link

7.3 Interviewees

1. Lindsey Nytes: Technical Sales Support Manager for OneRain - link
2. Glenn Hetchler: Lead Integrated Systems Engineer for OneRain - link