PREDICTING RAIN IN AUSTRALIA

Owen Bartels

ASU 9/24 /23

Table of Contents

Executive Summary:

The problem: Rain prediction has somewhat obvious benefits. It could help with water consumption, agriculture and planning to not go outside when it is raining. Machine learning research has been devoted to this topic for a long time with mixed results. This brings us to the problem.

Which Machine Learning Models predict rain better?

Can rain amounts be accurately predicted?

The solution:

Using a Kaggle Dataset on Australian rain, I was able to design three machine models based on predicting whether or not it will rain tomorrow given a certain day's attributes (i.e. temperature, pressure, wind, etc.). These models were a linear regression model, a Pytorch neural network, and a random forest classification model. I also fitted the random forest classification model to predict rainfall amounts.

Results:

The linear regression model, neural network, and random forest classification models tested at 66%, 84%, and 84%. I think the high accuracy is in large part due to guessing a question with a yes or no answer, where guessing no most of the time will get you close to 80% accuracy.

The rain amount prediction model in the second question got an accuracy score of 89.5% with a 30 mm margin of error. Within a margin of error of 30 millimeters is too loose, and the model needs some work. The best model for part one's predictions is tied between the random forest and neural network. The random forest was quicker, but the neural network would probably have a higher accuracy if the data set were to be expanded. This could give an edge to the neural network as the slight winner.

Project Plan

Data Set: Rain in Australia

No of applicable attributes: 17

Instance No: 145461

Data info:

The data covers most major cities and towns in mainland Australia as well as the islands of Tasmania and Norfolk. It has the following variables. Variables which have separately marked categories for time are marked as such.

Date: When the data was taken.

Location: What city the data was taken in.

MinTemp/MaxTemp: The minimum and maximum respective temperatures, both measured in Celsius.

Rainfall: Amount of rain that came down in a single day measured in millimeters.

Evaporation: "Class A" pan evaporation in the 24 hours to 9am (Not measured in all stations) unit millimeters

Sunshine: Measures how long the sun is out in hours, again a lot of stations do not have this variable

Wind Gust Direction(9am, 3pm, general): Where the wind blows.

WindGustSpeed(9am, 3pm, general): How fast the wind blows in km/hour

Humidity (9am, 3pm, general): The humidity measured in g/kg

Pressure(9am, 3pm, general): The air pressure measured in atm

Temp(9am, 3pm, general): The temperature measured in Celsius.

Cloud (9am, 3pm, general): Fraction of sky obscured by cloud. This is measured in "oktas", which are a unit of eigths.

Rain today: If it actually rained that day.

Rain tomorrow: If it actually rained tomorrow.

More about the data source:

When the Data was taken: 12/1/2008- 6/25/2017

Australian Government Bureau of Meteorology (BOM)

Founded: 1908

Employees:1500

Annual Budget: 270,979,800.00$ USD

Executive: Andrew Johnson, Director of Meteorology(2016-present)

Rob Vertessy, Director of Meteorology(2012-2016)

Greg Ayers, Director of Meteorology(2009-2012)

Neville Smith (Acting Director)(2008-2009)

Research Question 1: Which machine learning models have the highest rate of accuracy when it comes to correctly predicting whether or not it will rain tomorrow?

Method: The original aim of the dataset compiled by Joe Young and Adam Young is to build a machine learning model that predicts the binary rain variable. My objective somewhat differs as I will be expanding on this and seeing what ML models, I can have the most success with. I plan to try some supervised and unsupervised models. I am going to try Random Forest regression with sklearn, deep learning with a neural network setup using Pytorch, and K means clustering. I will probably add models as I start writing the code as I want a very large comparison pool. I hope to include plenty of mathematical explanations to the methods I am using and how they interact with my specific data, this will probably involve doing more research into weather science.

Goal: Predicting whether or not it will rain has obvious huge agricultural applications, but figuring out the best method for weather prediction can be difficult, I hope to make some discoveries on what works and how.

2. Can rainfall be made the predicted variable? This is somewhat an extension of the first question. By determining severity of rain, a lot more useful decision making can be made. For example, heavy rain damages power lines which lighter rain usually does not. If heavy rain can be predicted, power companies can install protective equipment the day before heavy rainfall occurs in order to prevent power supply damage.

Hypothesis: Factors such as temperature, pressure, windspeed, humidity and wind direction can be used to provide an accurate prediction of whether or not it will rain:

This will be tested by the creation of our ML models using the inputs of the said variables as test cases.

H2: Rainfall can also be predicted with similar machine learning models to the one described in hypothesis one: This will be tested by setting rainfall as our test case variable.

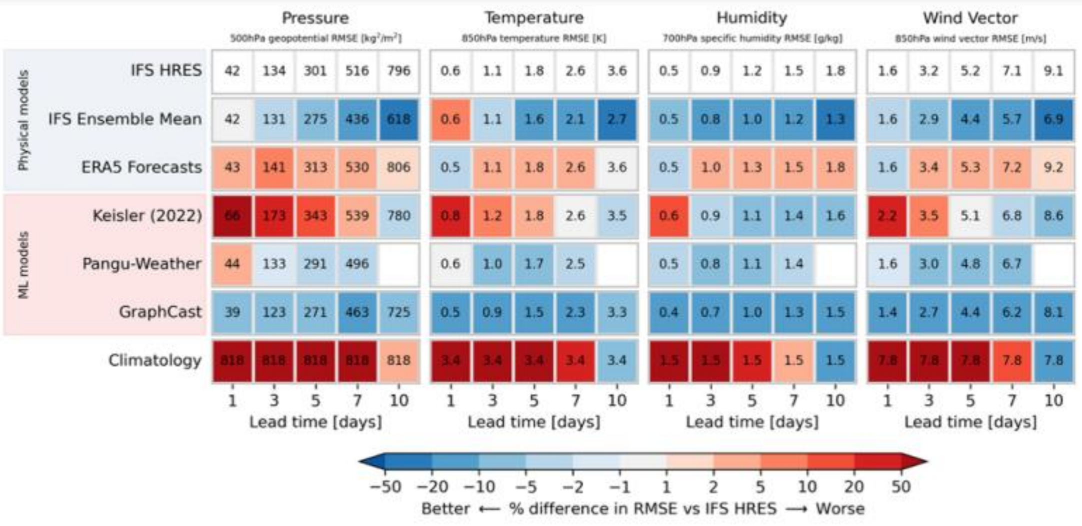
Methodology: I am going to create a Jupyter Notebook, then I will read the weather.csv file into that notebook, I am then going to remove all NAN values. With this modified file, I am going to make sure the data fits to my machine learning models, set my predicted variables, setup my test and training data sets, and then make various ml models. The logistical regression requires few other commands than the ones I specified. I'm going to have to figure out the specifics on how I want to layout the layers in the neural network, and   I will then repeat the same process, but using rainfall as the variable to determine. There will be some calculations as to number of false positives, false negatives to calculate the accuracy of each model, and I will have graphs for each model using Matplot.lib. I will then compare the models to each other and determine which model is best.

Computational Methods and Outputs:

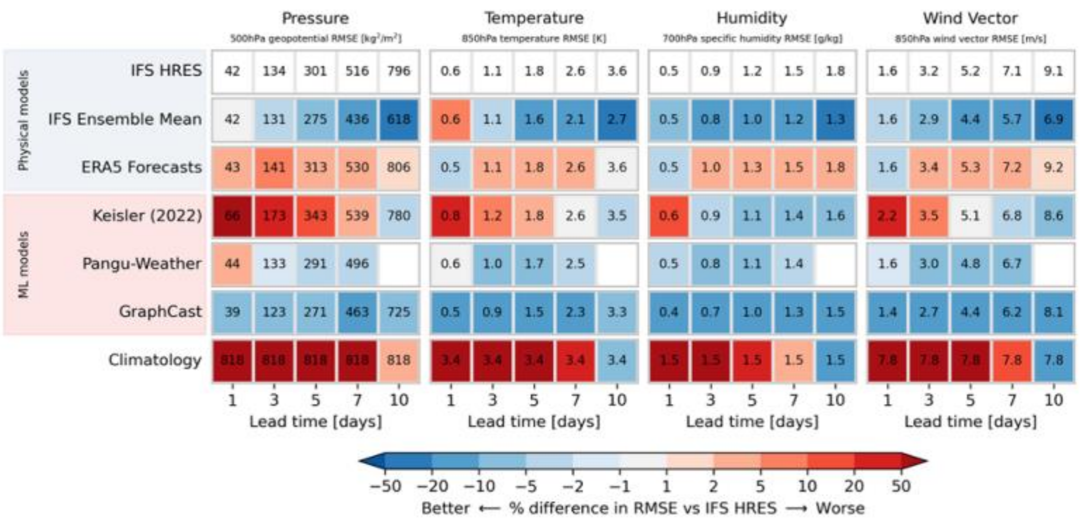
For question one and two, I think I will use AUC/ROC classification to determine which model predicts rain the best. I will also probably have to use cross fold validation to prevent model overfitting. The models I am currently going to try are random forest and a neural network of some type. I also hope to try a real developed weather based machine learning model such as GraphCast, but I am not sure if the data as prepared could be repurposed to fit GraphCast or any other pre developed model's necessary inputs as the data set I am using is missing spatial data. I may have to further text mine the sources of which the Kaggle dataset was developed from in order to get the necessary spatial variables if those are required for the predeveloped models I am looking at trying out in this project.

Output Summaries:

Which machine learning models have the highest rate of accuracy when it comes to correctly predicting whether or not it will rain tomorrow?



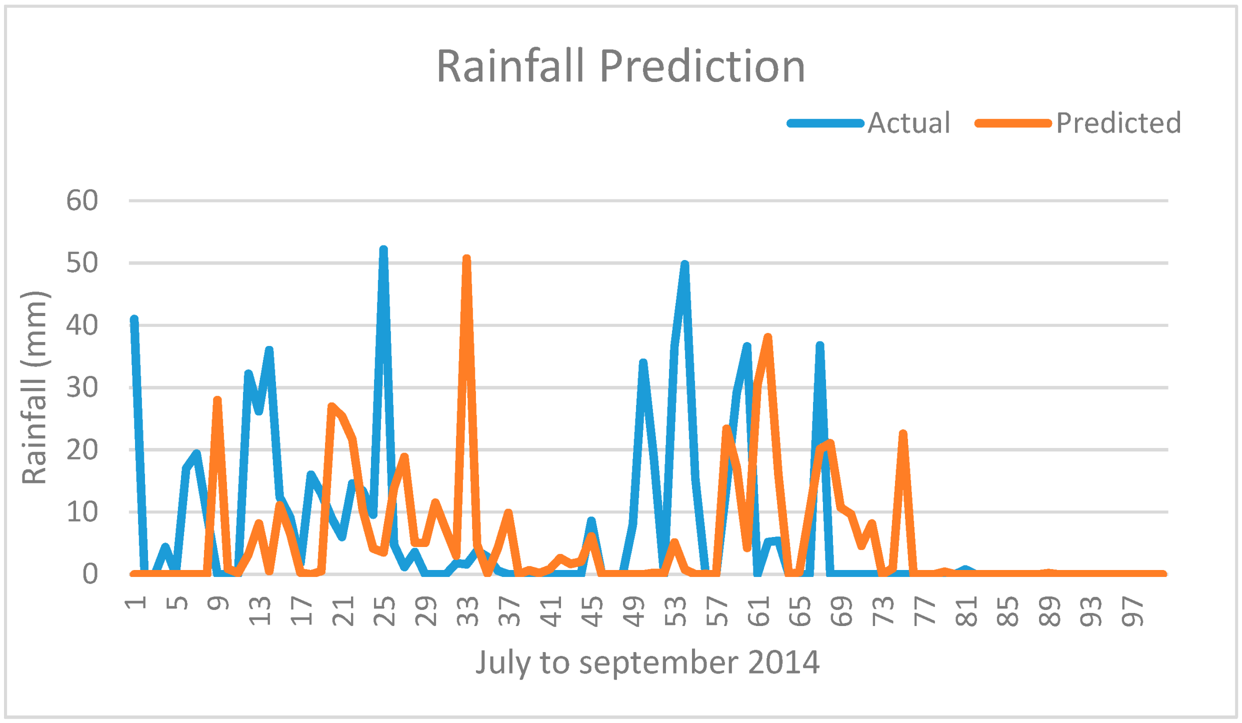
	Pressure					Temperature					Humidity					Wind Vector				
	500hPa geopotential RMSE $[kg^2/m^2]$					850hPa temperature RMSE [K]					700hPa specific humidity RMSE [g/kg]					850hPa wind vector RMSE [m/s]				
Lead time [days]	1	3	5	7	10	1	3	5	7	10	1	3	5	7	10	1	3	5	7	10
IFS HRES	42	134	301	516	796	0.6	1.1	1.8	2.6	3.6	0.5	0.9	1.2	1.5	1.8	1.6	3.2	5.2	7.1	9.1
IFS Ensemble Mean	42	131	275	436	618	0.6	1.1	1.6	2.1	2.7	0.5	0.8	1.0	1.2	1.3	1.6	2.9	4.4	5.7	6.9
ERA5 Forecasts	43	141	313	530	806	0.5	1.1	1.8	2.6	3.6	0.5	1.0	1.3	1.5	1.8	1.6	3.4	5.3	7.2	9.2
Keisler (2022)	66	173	343	539	780	0.8	1.2	1.8	2.6	3.5	0.6	0.9	1.1	1.4	1.6	2.2	3.5	5.1	6.8	8.6
Pangu-Weather	44	133	291	496		0.6	1.0	1.7	2.5		0.5	0.8	1.1	1.4		1.6	3.0	4.8	6.7	
GraphCast	39	123	271	463	725	0.5	0.9	1.5	2.3	3.3	0.4	0.7	1.0	1.3	1.5	1.4	2.7	4.4	6.2	8.1
Climatology	818	818	818	818	818	3.4	3.4	3.4	3.4	3.4	1.5	1.5	1.5	1.5	1.5	7.8	7.8	7.8	7.8	7.8


	Pressure					Temperature					Humidity					Wind Vector				
	500hPa geopotential RMSE $[kg^2/m^2]$					850hPa temperature RMSE [K]					700hPa specific humidity RMSE [g/kg]					850hPa wind vector RMSE [m/s]				
Lead time [days]	1	3	5	7	10	1	3	5	7	10	1	3	5	7	10	1	3	5	7	10
IFS HRES	42	134	301	516	796	0.6	1.1	1.8	2.6	3.6	0.5	0.9	1.2	1.5	1.8	1.6	3.2	5.2	7.1	9.1
IFS Ensemble Mean	42	131	275	436	618	0.6	1.1	1.6	2.1	2.7	0.5	0.8	1.0	1.2	1.3	1.6	2.9	4.4	5.7	6.9
ERA5 Forecasts	43	141	313	530	806	0.5	1.1	1.8	2.6	3.6	0.5	1.0	1.3	1.5	1.8	1.6	3.4	5.3	7.2	9.2
Keisler (2022)	66	173	343	539	780	0.8	1.2	1.8	2.6	3.5	0.6	0.9	1.1	1.4	1.6	2.2	3.5	5.1	6.8	8.6
Pangu-Weather	44	133	291	496		0.6	1.0	1.7	2.5		0.5	0.8	1.1	1.4		1.6	3.0	4.8	6.7	
GraphCast	39	123	271	463	725	0.5	0.9	1.5	2.3	3.3	0.4	0.7	1.0	1.3	1.5	1.4	2.7	4.4	6.2	8.1
Climatology	818	818	818	818	818	3.4	3.4	3.4	3.4	3.4	1.5	1.5	1.5	1.5	1.5	7.8	7.8	7.8	7.8	7.8

Retrieved from: https://arxiv.org/abs/2308.15560

The outputs I am looking at for my first research question involve a heatmap similar to the one found in the weather-based ml paper I found in my lit review. I want to compare each of the models based on the variables inputted into them in a similar way to that photo. I would also like to include a table of the various predicted results against their actual results.

Can rainfall be made the predicted variable?

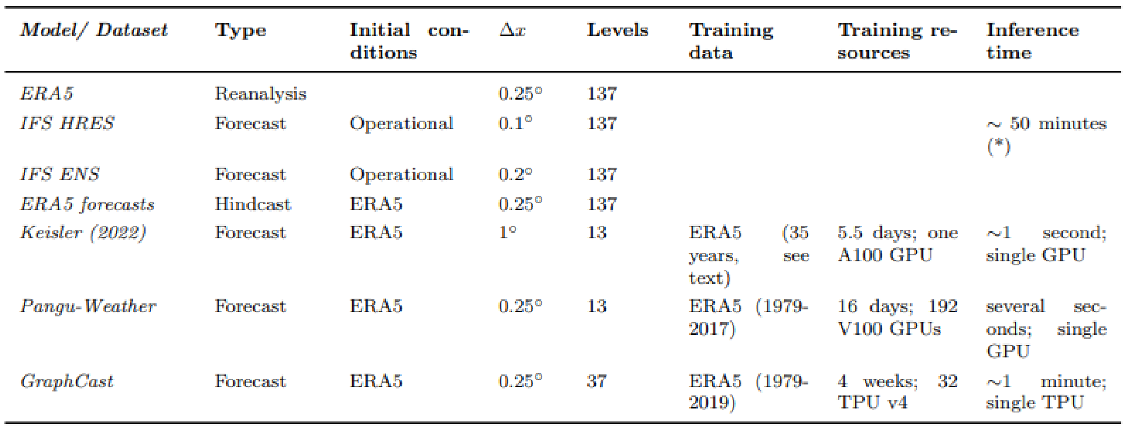


Retrieved from: https://www.mdpi.com/2073-4433/10/11/668 (Prediction of Rainfall Using Intensified LSTM Based Recurrent Neural Network with Weighted Linear Units)

I hope to have a comparative graph like this with actual rainfall in blue and my predicted rainfall in orange plotted against my time values. As I am using multiple models, I could theoretically put all the predicted model results against the actual, but I will have to see how crowded that graph actually looks before deciding on whether to include it or not.

Lit Review

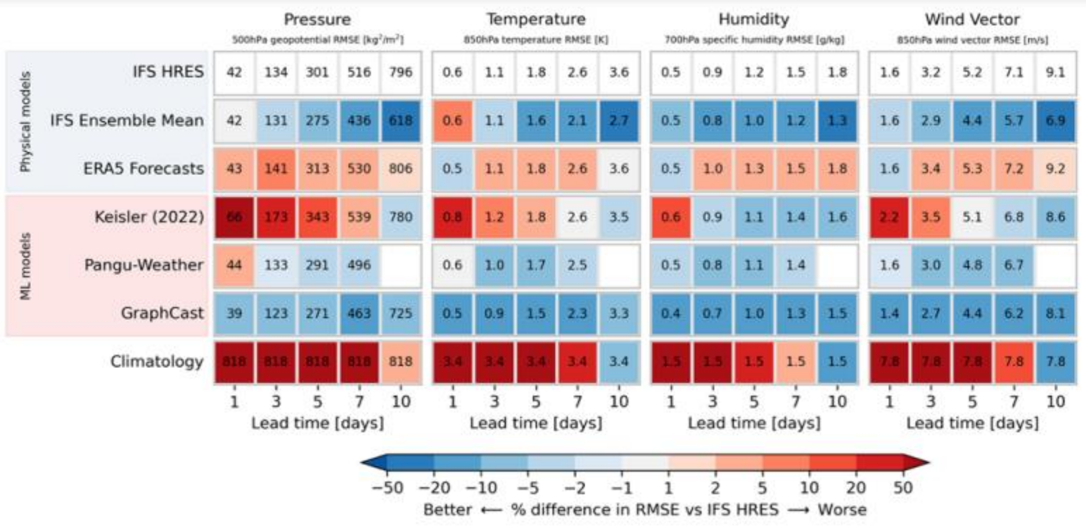
Using machine learning for rain prediction has been around for almost as long as machine learning has. There is no shortage of literature to review which can make it a challenge to find what to write about in this section. I thought I would first start off with Australian uses for rain prediction. One of the biggest users of weather-based ML is Aus Net an Australian power company. An Aus Mag article by Renny Vandewege in Sep 2023 goes into how the 2020 bush fires and flooding forced Aus Net to adapt their current methods of weather-based grid protection using ML. Ultimately, they have switched to using DTN Storm Impact Analytics. I looked up DTN's website, but couldn't really find anything about the AI specifics of the algorithms they use, other than it can factor in grid stability, weather, topography etc. I looked at some of the 550 projects of the dataset I am using on Kagle. Not all of them used ML, so I started with the most upvoted ones. One of them I found, a Kaggle notebook by Prashant Banree on Aug 3 rd 2021, Prashant was able to make a logistic regression model with 85% accuracy, and 77% null accuracy and stated in his conclusions that a more complex model could probably get better accuracy. I'm planning on confirming whether this statement is accurate by fitting a neural network with the same information. I came across a paper, WeatherBench 2: A Benchmark for the Next Generation of Data-Driven Global Weather by Stephan Rasp, Research Scientist, and Carla Bromberg, Program Lead, Google Research which is about Google recently releasing Benchmark 2 which compares various ML models against each other for the specific purpose of weather prediction. There is publicly available free training and ml model code, so I might use this in my project.

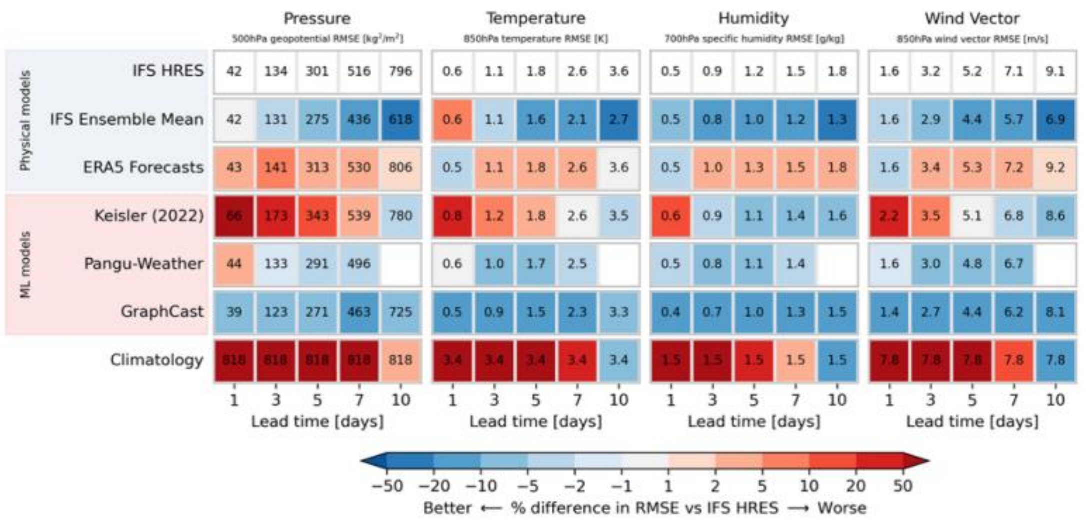


Model/ Dataset	**Type**	**Initial conditions**	Δx	**Levels**	**Training data**	**Training resources**	**Inference time**
ERA5	Reanalysis		0.25°	137			
IFS HRES	Forecast	Operational	0.1°	137			~ 50 minutes (*)
IFS ENS	Forecast	Operational	0.2°	137			
ERA5 forecasts	Hindcast	ERA5	0.25°	137			
Keisler (2022)	Forecast	ERA5	1°	13	ERA5 (35 years, see text)	5.5 days; one A100 GPU	~1 second; single GPU
Pangu-Weather	Forecast	ERA5	0.25°	13	ERA5 (1979-2017)	16 days; 192 V100 GPUs	several seconds; single GPU
GraphCast	Forecast	ERA5	0.25°	37	ERA5 (1979-2019)	4 weeks; 32 TPU v4	~1 minute; single TPU

Table 1: Table of datasets and models; Δx refers to the horizontal resolution and "levels" to the number of vertical model levels used in the input and output. (*) See details in the text for IFS inference time.

This paper may be one of my most significant finds, but a lot of the terminology is above my level, and most of the models seem to be using high spec computers or supercomputers to run.





These heat maps are basically similar to how I want to display my project results, but the machine learning models they are using are more advanced than the ones I know. I hadn't heard of Keisler, Pangu, or GraphCast until reading this paper.

"One feature of current AI methods is that they produce unrealistically smooth forecasts, a trend that often becomes more pronounced with lead time. This is a direct consequence of these models minimizing a deterministic mean error (such as MSE). Because the evolution of the atmosphere is chaotic and forecast uncertainty grows with time, this will lead models to "hedge their bets" by predicting the mean of the potential forecast distribution. In a non-linear, high-dimensional system such as the atmosphere, the mean of many states is not a realistic state itself. The result are blurry forecasts that perform well on some skill metrics (like RMSE or ACC) but do not represent a possible weather state. This is evident in several analyses presented here: spectra show excessive blurring of AI models for longer lead times; the SEEPS score shows how blurry models (IFS ENS mean and GraphCast) fail to predict the right precipitation category; wind speed biases are evidence that current AI models have difficulties learning correlations between variables; and the case studies show that the intensity of local features such as wind speed, precipitation and cyclone intensity aren't represented. For some applications, having good forecasts of average weather is sufficient but for others AI models are not yet appropriate."

Reading this on page 24 was interesting. This passage states a couple of things relevant to my paper. The biggest one is that machine learning models often have trouble learning how different weather variables are related, which is a big challenge as my model is trying to get predictions based on correlations of weather variables. The second is that the mean of all weather states is not a possible weather state. Thirdly, some weather models can work with just average weather which means, the ones that I am testing that can work with average weather states will probably be the most accurate. I also looked into what it meant that Graphcast cannot predict the right precipitation category. From what I could tell, this means Graphcast can't tell the difference between sleet, rain, and drizzle, along with other precipitation types.

I had time to read some of GraphCast: Learning Skillful Medium-Range Global Weather Forecasting a paper written in August, 2022 by a team of professors working for Google DeepMind a research branch of Google and figured some things about Graphcast.   Page 2 gives the best general overview of the model.

“Here we introduce a new MLWP approach for global medium-range weather forecasting called “GraphCast”, which produces an accurate 10-day forecast in under a minute on a single Google Cloud TPU v4 device, and supports applications including predicting tropical cyclone tracks, atmospheric rivers, and extreme temperatures. GraphCast takes as input the two most recent states of Earth’s weather—the current time and six hours earlier—and predicts the next state of the weather six hours ahead. A single weather state is represented by a 0.25° latitude/longitude grid (721 × 1440), which corresponds to roughly 28 × 28 kilometer resolution at the equator (Figure 1a), where each grid point represents a set of surface and atmospheric variables (listed in Table 1). Like traditional NWP systems, GraphCast is autoregressive: it can be “rolled out” by feeding its own predictions back in as input, to generate an arbitrarily long trajectory of weather states (Figure 1b–c).”

I understood that GraphCast despite only taking 2 measurements that are 6 hours apart, can predict a theoretically infinitely long weather state by just feeding in the 6 hour ahead output prediction you got as an input. GraphCast’s innerworkings seem to be that of a highly complex neural network. I am not sure how I would go about fixing its precipitation based blurry state problem, nor am I sure if I could. In my project I hope to be able to find some solution to properly defining precipitation states.

Exploratory Data Analysis

For this project I chose a Kagle dataset called Rain in Australia:

https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package. The data is in a csv format and was scraped from the Australian Bureau of Meteorology.

The data has 145461 observations and 25 variable fields. because these variables were selectively. My goal is to find what factors influence rain happening as well as make a model that predicts rainfall amounts, so I would rather not delete too many variables if necessary.

MinTemp/MaxTemp: The minimum and maximum respective temperatures, both measured in Celsius.

Rainfall: Amount of rain that came down in a single day measured in millimeters.

Wind Gust Direction(9am, 3pm, general): Where the wind blows.

WindGustSpeed(9am, 3pm, general): How fast the wind blows in km/hour

Humidity (9am, 3pm, general): The humidity measured in g/kg

Pressure(9am, 3pm, general): The air pressure measured in atm

Evaporation: "Class A" pan evaporation in the 24 hours to 9am (Not measured in all stations) unit millimeters

Sunshine: Measures how long the sun is out in hours, again a lot of stations do not have this variable

Temp(9am, 3pm, general): The temperature measured in Celsius.

Cloud (9am, 3pm, general): Fraction of sky obscured by cloud. This is measured in "oktas", which are a unit of 1/8.

Rain today: If it actually rained that day.

Rain tomorrow: If it actually rained tomorrow.

Methods of Analysis:

I will be using Excel and some Python as my tools for analysis in this section. There are a couple of advantages and disadvantages to both. Python has a lot more flexibility and ultimately is what I will be coding the actual project in. I find Excel better for some aspects of looking at and fixing the data.

The Excel portion:

So to start this of I wanted to figure out if wind direction is a factor in whether or not it would rain. Generally rainy days occur less than non-rainy days, so I am expecting ratios of less than 40% in all of the directions.

Count of RainToday														
	SE	**W**	**S**	**WSW**	**SW**	**SSE**	**N**	**WNW**	**NW**	**ESE**	**E**	**NE**	**SSW**	**NNW**
No	8328	7005	6962	6563	6567	6932	7363	6310	6572	6887	6918	7070	5553	6345
Yes	2340	2911	2623	2750	2616	2208	1326	2363	1892	1482	1426	1095	2454	1373
Grand Total	**10668**	**9916**	**9585**	**9313**	**9183**	**9140**	**8689**	**8673**	**8464**	**8369**	**8344**	**8165**	**8007**	**7718**
Rain ratio	21.93	29.36	27.37	29.53	28.49	24.16	15.26	27.25	22.35	17.71	17.09	13.41	30.65	17.79

We found the highest likelihood of rain being on days when South Southwest winds were blowing at 30.65%. This coincides with the Australia Bureau of Meteorology's findings "In eastern Australia, the southward movement of the westerly winds means that more moist onshore flow from the Tasman and Coral seas is drawn inland, and thus increases rainfall for eastern Australia." Most of our data points are in eastern Australia, so it makes sense that South Southwest winds would rank the highest here.

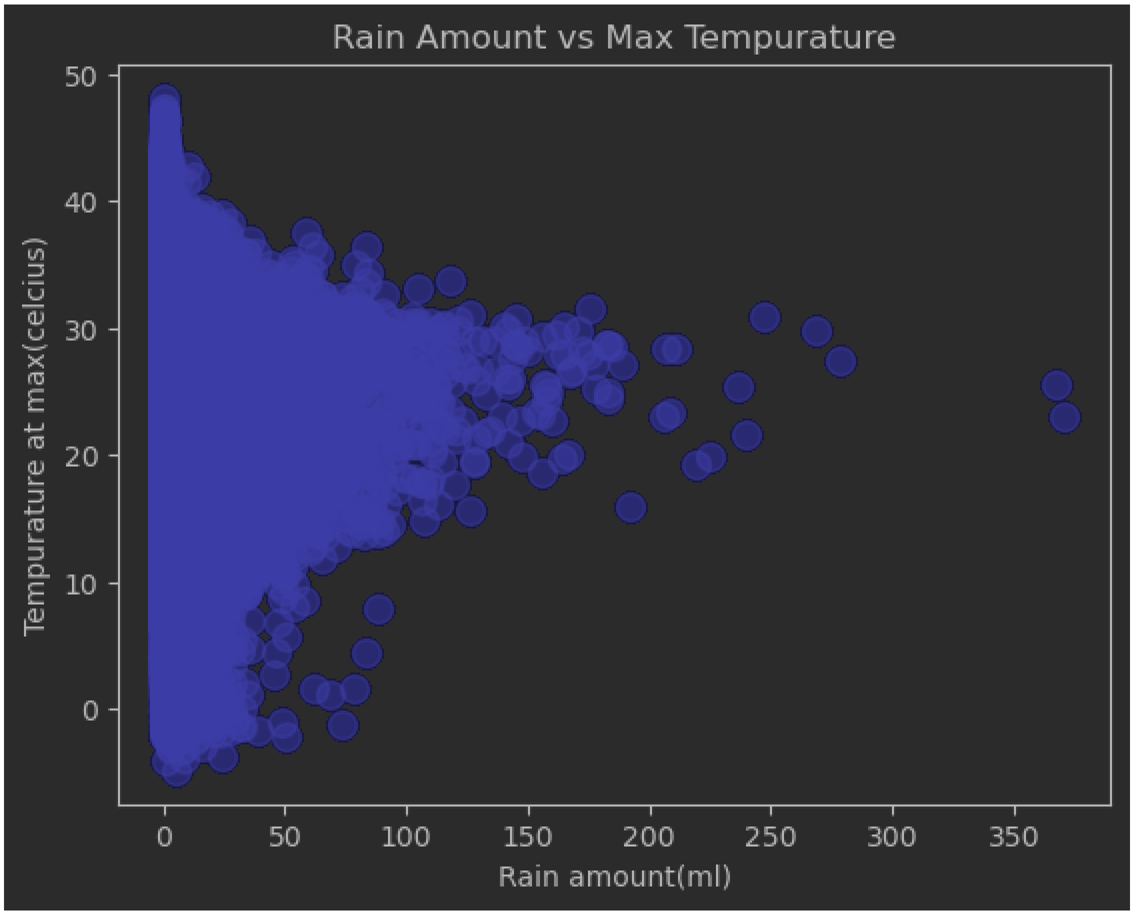
It’s not stated here, but I found some NAS in the whether it will rain today and tomorrow columns and was wondering how easy it would be to fix as whether or not it rained on a certain day in a part of Australia should be somewhat readily available information.

RainTomorrow	Count of RainTomorrow
No	110316
Yes	31877
NA	3267
Grand Total	**145460**

Not very easy without the extensive use of programming it would seem. The NA values line up with eachother, if raintoday is NA, than the previous raintomorrow will also be NA. Rainfall is also NA on days when raintoday is as well, but for some reason Rainfall has 6 fewer NA values.

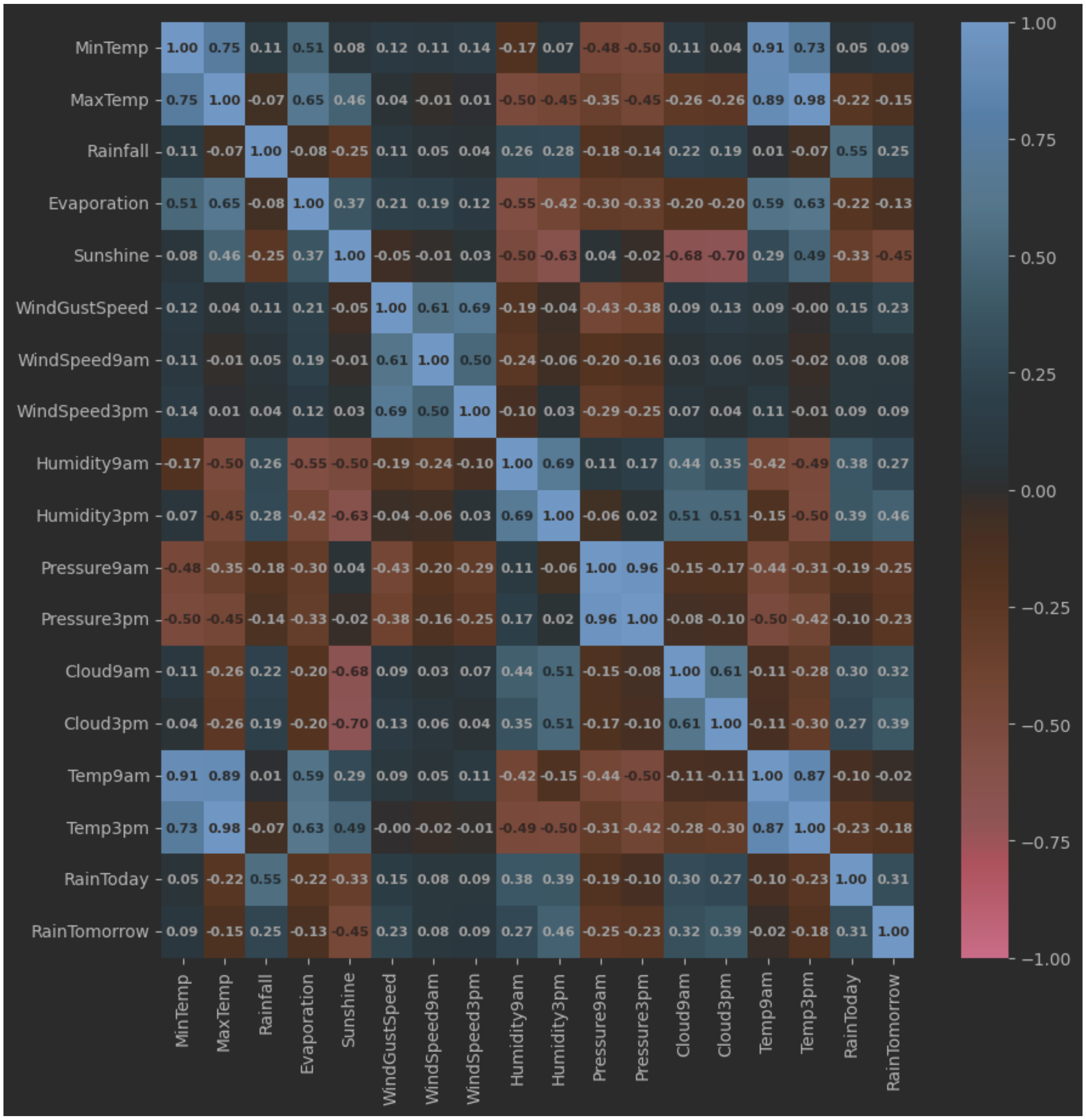
I tried some graphing in Excel, but hated most of what I got, it’s time to switch to Python.

The Python Portion:



Here I have a scatter of the max temperature column compared to rain fall amount. It's apparent from the graph that the highest amount of rain tends to result at temperatures between 22-30 Celcius. This may be useful in my second research question, but I don't think it's the most pertinent info right now. Let's see what else is there.

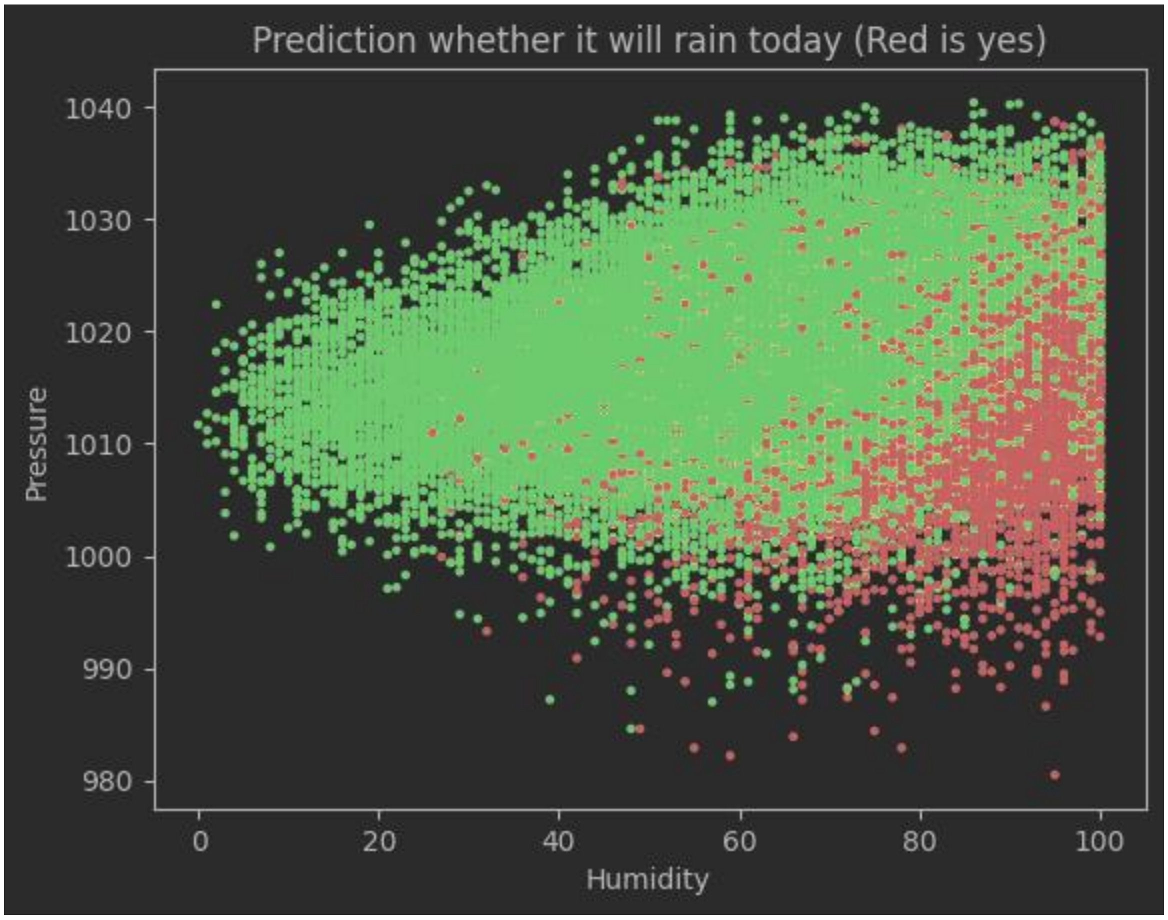
Part of the reason I used Python for parts of this project is because unlike Excel, it can handle very large volumes of data when graphing. Let's see an example of this.



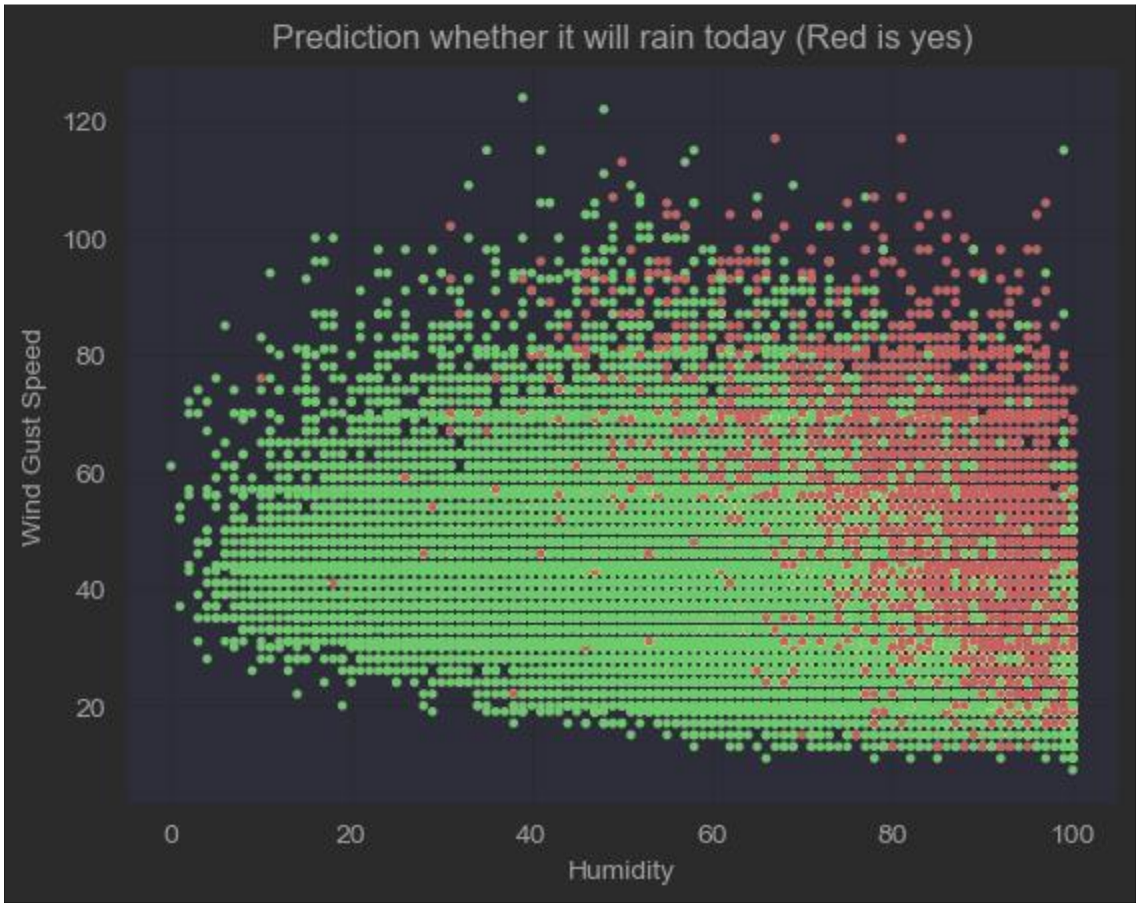
	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
MinTemp	1.00	0.75	0.11	0.51	0.08	0.12	0.11	0.14	-0.17	0.07	-0.48	-0.50	0.11	0.04	0.91	0.73	0.05	0.09
MaxTemp	0.75	1.00	-0.07	0.65	0.46	0.04	-0.01	0.01	-0.50	-0.45	-0.35	-0.45	-0.26	-0.26	0.89	0.98	-0.22	-0.15
Rainfall	0.11	-0.07	1.00	-0.08	-0.25	0.11	0.05	0.04	0.26	0.28	-0.18	-0.14	0.22	0.19	0.01	-0.07	0.55	0.25
Evaporation	0.51	0.65	-0.08	1.00	0.37	0.21	0.19	0.12	-0.55	-0.42	-0.30	-0.33	-0.20	-0.20	0.59	0.63	-0.22	-0.13
Sunshine	0.08	0.46	-0.25	0.37	1.00	-0.05	-0.01	0.03	-0.50	-0.63	0.04	-0.02	-0.68	-0.70	0.29	0.49	-0.33	-0.45
WindGustSpeed	0.12	0.04	0.11	0.21	-0.05	1.00	0.61	0.69	-0.19	-0.04	-0.43	-0.38	0.09	0.13	0.09	-0.00	0.15	0.23
WindSpeed9am	0.11	-0.01	0.05	0.19	-0.01	0.61	1.00	0.50	-0.24	-0.06	-0.20	-0.16	0.03	0.06	0.05	-0.02	0.08	0.08
WindSpeed3pm	0.14	0.01	0.04	0.12	0.03	0.69	0.50	1.00	-0.10	0.03	-0.29	-0.25	0.07	0.04	0.11	-0.01	0.09	0.09
Humidity9am	-0.17	-0.50	0.26	-0.55	-0.50	-0.19	-0.24	-0.10	1.00	0.69	0.11	0.17	0.44	0.35	-0.42	-0.49	0.38	0.27
Humidity3pm	0.07	-0.45	0.28	-0.42	-0.63	-0.04	-0.06	0.03	0.69	1.00	-0.06	0.02	0.51	0.51	-0.15	-0.50	0.39	0.46
Pressure9am	-0.48	-0.35	-0.18	-0.30	0.04	-0.43	-0.20	-0.29	0.11	-0.06	1.00	0.96	-0.15	-0.17	-0.44	-0.31	-0.19	-0.25
Pressure3pm	-0.50	-0.45	-0.14	-0.33	-0.02	-0.38	-0.16	-0.25	0.17	0.02	0.96	1.00	-0.08	-0.10	-0.50	-0.42	-0.10	-0.23
Cloud9am	0.11	-0.26	0.22	-0.20	-0.68	0.09	0.03	0.07	0.44	0.51	-0.15	-0.08	1.00	0.61	-0.11	-0.28	0.30	0.32
Cloud3pm	0.04	-0.26	0.19	-0.20	-0.70	0.13	0.06	0.04	0.35	0.51	-0.17	-0.10	0.61	1.00	-0.11	-0.30	0.27	0.39
Temp9am	0.91	0.89	0.01	0.59	0.29	0.09	0.05	0.11	-0.42	-0.15	-0.44	-0.50	-0.11	-0.11	1.00	0.87	-0.10	-0.02
Temp3pm	0.73	0.98	-0.07	0.63	0.49	-0.00	-0.02	-0.01	-0.49	-0.50	-0.31	-0.42	-0.28	-0.30	0.87	1.00	-0.23	-0.18
RainToday	0.05	-0.22	0.55	-0.22	-0.33	0.15	0.08	0.09	0.38	0.39	-0.19	-0.10	0.30	0.27	-0.10	-0.23	1.00	0.31
RainTomorrow	0.09	-0.15	0.25	-0.13	-0.45	0.23	0.08	0.09	0.27	0.46	-0.25	-0.23	0.32	0.39	-0.02	-0.18	0.31	1.00

Here I have a table of how each variable correlates to one another whether it is a positive or negative correlation. My target prediction values are Rainfall and Rain tomorrow. Rainfall has relatively high correlation with evaporation, and sunshine, both of which are unfortunately rather high in NA values, and also has a .26 and .28 correlation with Humidity at 9 am and Humidity at 3pm respectively. Rain tomorrow has a surprisingly higher correlation with Humidity at 3pm at .46. Pressure also has a relatively high negative correlation with Rain tomorrow at -0.25 for 9am and -0.23 at 3pm. Temperature only gets as low as -0.18 in its 9am column in terms of negative correlation with Rain Tomorrow. Time to graph some of this data.

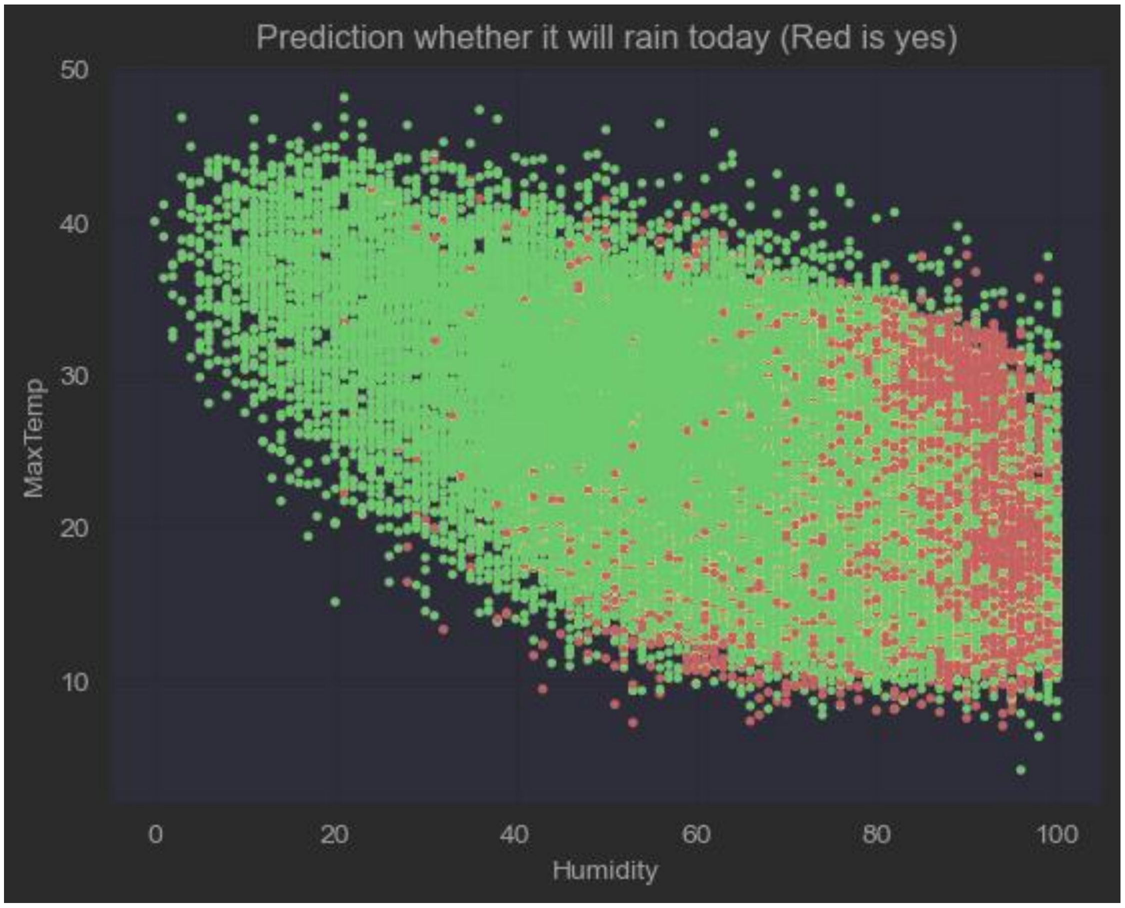
Note: (Changing ‘yes’ and ‘no’ to a binary 0 and one makes RainToday and RainTomorrow actually graphable)

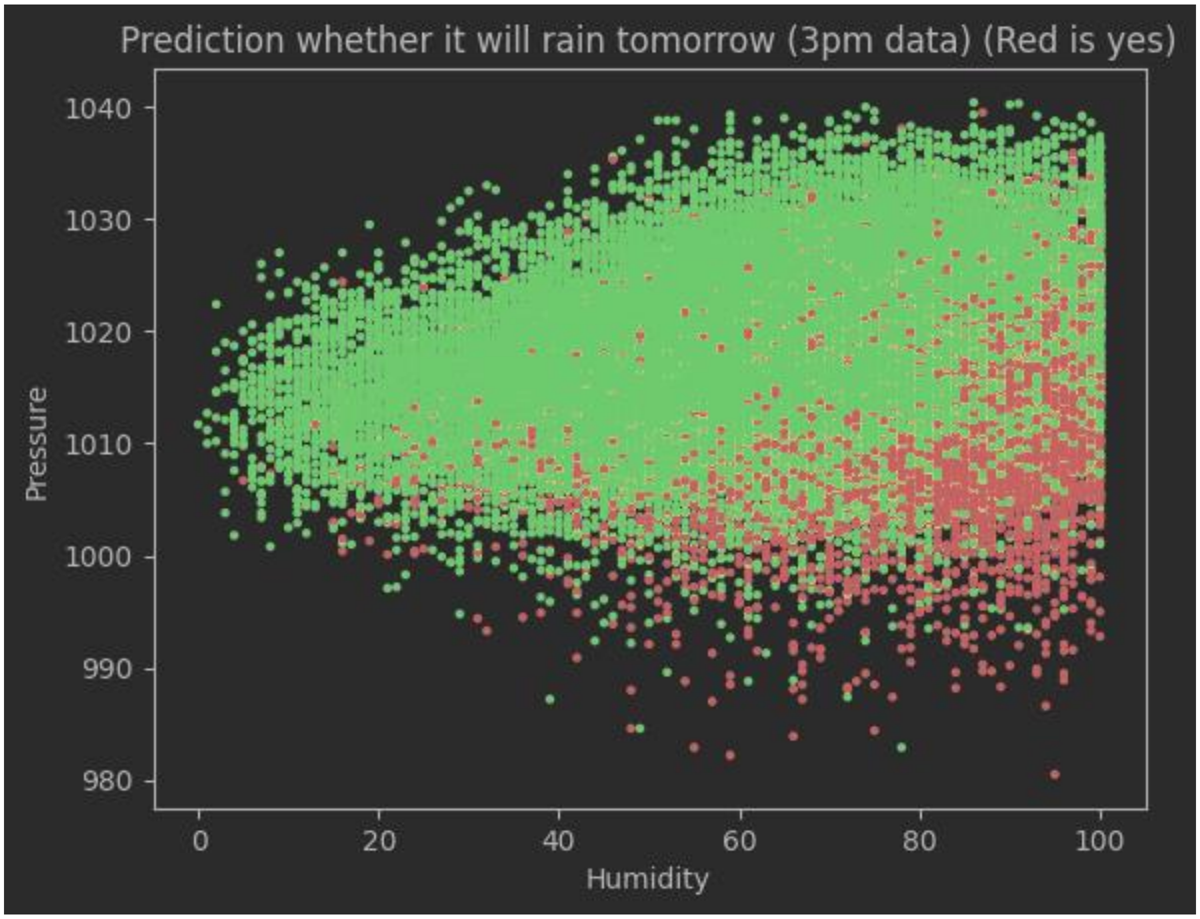


This is a graph between humidity and pressure using 9am data. I am also using the Rain Tomorrow variable as a binary colormap. Higher humidity seems to be making more of a difference to whether or not it will rain tomorrow although, high pressure and high humidity seems to result in fewer rainy days.

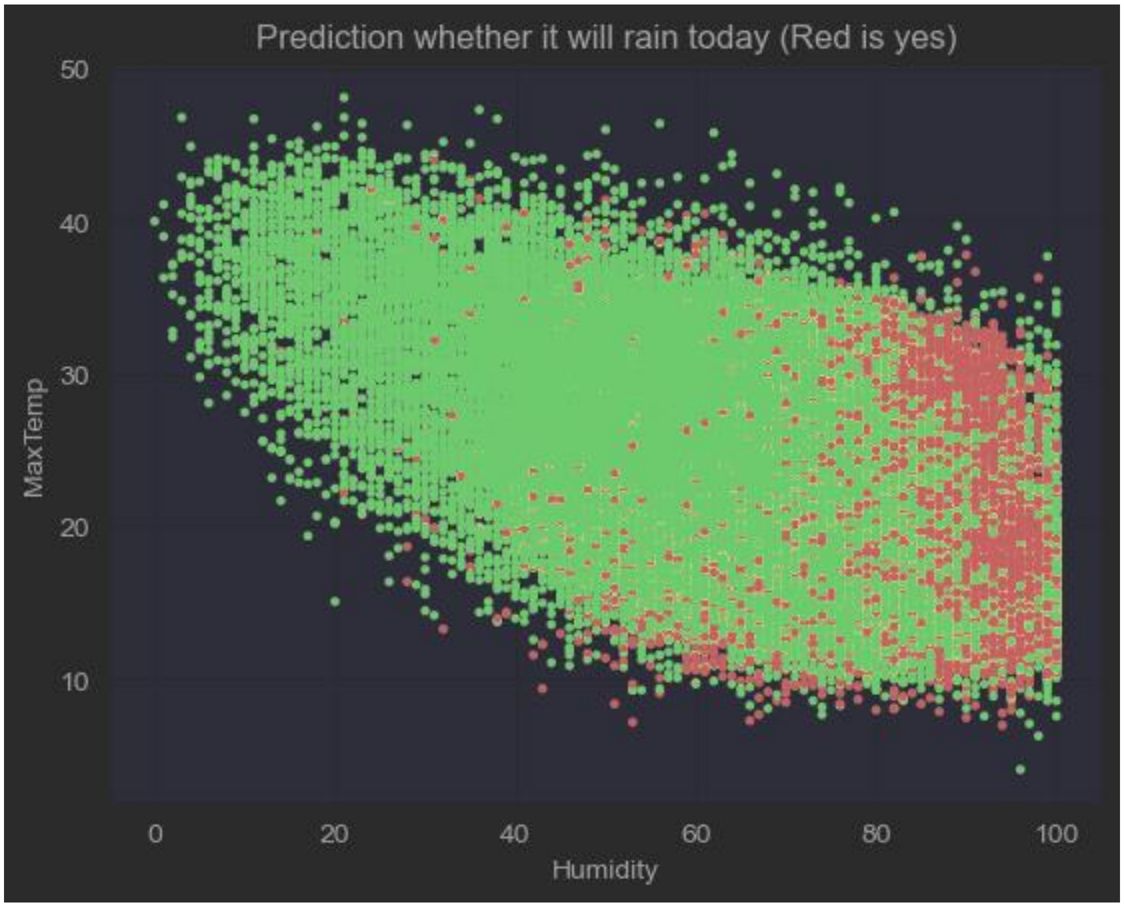


Now we are predicting wind gust speed against humidity and it seems both high humidity and high wind gust speeds are needed for a high chance of rain.

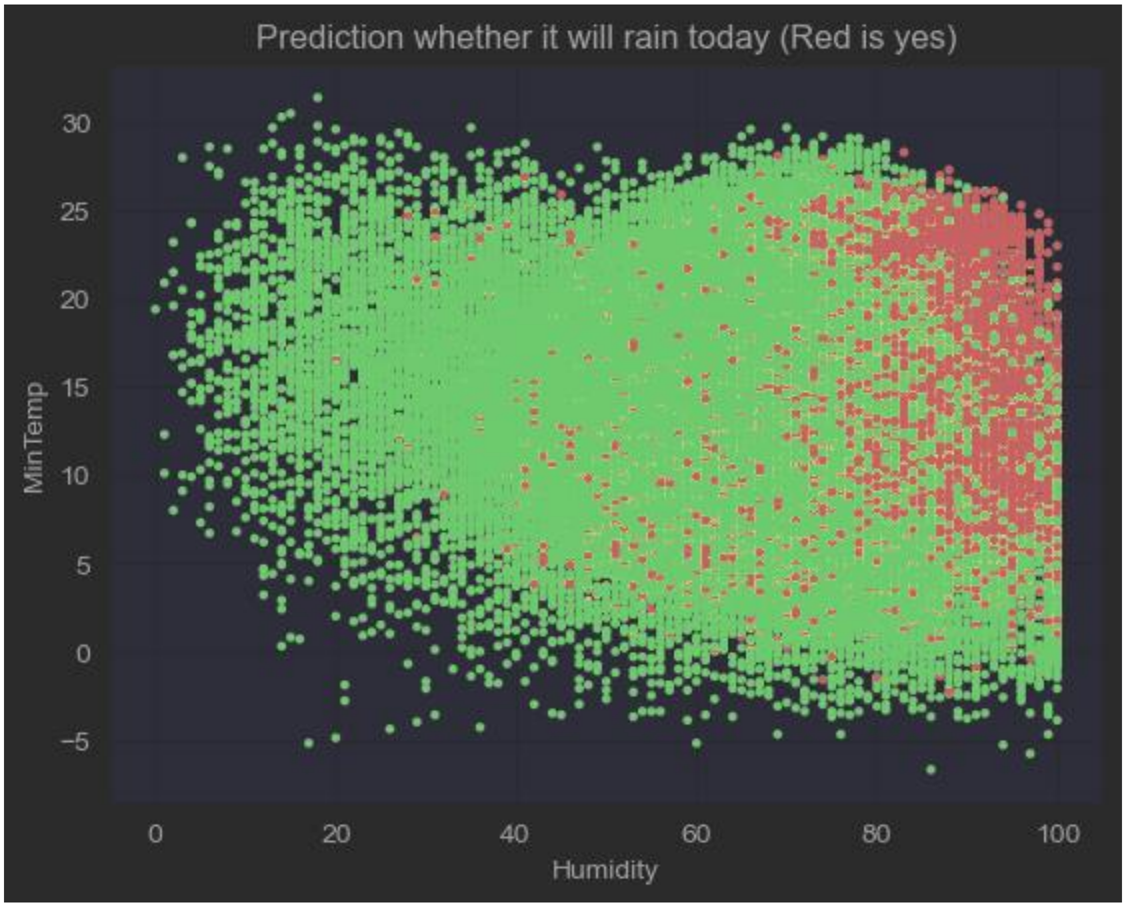

Prediction whether it will rain today (Red is yes)
MaxTemp
Humidity
50
40
30
20
10
0
20
40
60
80
100



The 3pm data looks slightly different, but the same conclusions can be drawn.

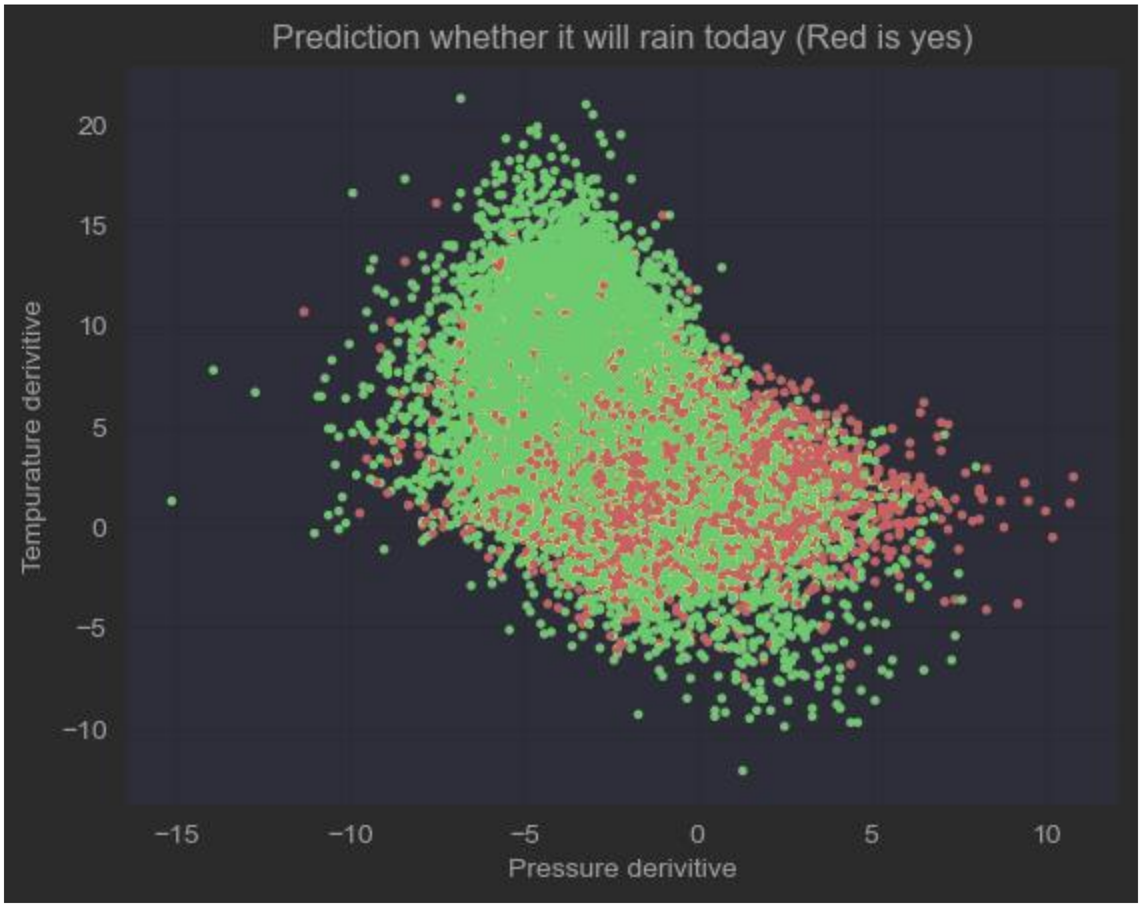


The high temperatures don’t seem to correlate as much as humidity does here.



Min temperature tells a similar story only that it seems to rain less at temperatures below 5 degrees Celsius.

For the last part of my analysis, I would like to involve a bit of math. I didn't exactly know what to do with temperature recorded at 3pm and 9am as well as pressure at 3pm and 9am, but then I thought about derivatives. Sudden drops or gains in air pressure and temperatures increase the chances of rain, so instead of measuring the variables separately what if you subtract 3pm's count by 9 a.m.'s count and do a prediction graph of that?



This graph shows pressure increases to increase the chance of rain, and a little that pressure decreases also cause the same. It also seems temperature raises between 1 and 5 degrees as well as 1 and -4 also increase the chances of rain.

Methodology Report Owen Bartels

RQ1 –What machine learning model works the best for predicting whether it will rain tomorrow?

I am going to use the Jupyter Notebook I made for the EDA that I have sufficiently cleaned already. With this modified file, I am going to make sure the data fits to my machine learning models, set my predicted variables, set up my test and training data sets, and then make various ml models. I'm going to set up a neural network using Pytorch's deep learning library. It's going to be a 5 layer network with an input dimension of 100 each using a lognet fit that as epochs set to 80 and a batch size of 250. I will adjust this neural network several times depending on outputs given. I plan on using lognet to also classify the neural network's accuracy. There will be some calculations as to number of false positives, false negatives to calculate the accuracy of each model and ROC AUC will also be calculated at this time. There will be several graphs of each model used to better display what is at work.

Methods Planned for use:

Regression analysis: Use coefficients of independent variables (pressure, wind, temp, etc) to determine the log odds of rain occurring tomorrow.

Random forest: Supervised learning model type where we are giving the model the target of predicting rainfall with pressure, temperature and other independent variables as the predictors.

RQ2 -Can Rainfall amounts be predicted with a machine learning model?

This will just repeat the methodology and programming of the first question, but use rainfall as the variable to predict instead of whether or not it will rain tomorrow. I will also only use the Random Forest model. Through my EDA, I have determined that rainfall has a slightly different set of correlations to whether or not it rains tomorrow, so it makes some sense to weigh the predictive variables differently in each of these model trials. Pressure in particular should be weighted higher in rainfall amounts as opposed to how it is weighted in predicting rainfall tomorrow.

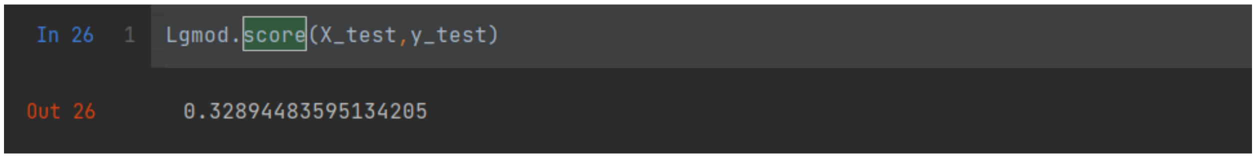
Data Visualizations:

In general:

The plots in this section are similar to that in my EDA, but the key difference is that these graphs are using correct and incorrect predictions by the machine learning models to determine the color map as opposed to whether or not it will rain today or tomorrow which were the predictors in my EDA.

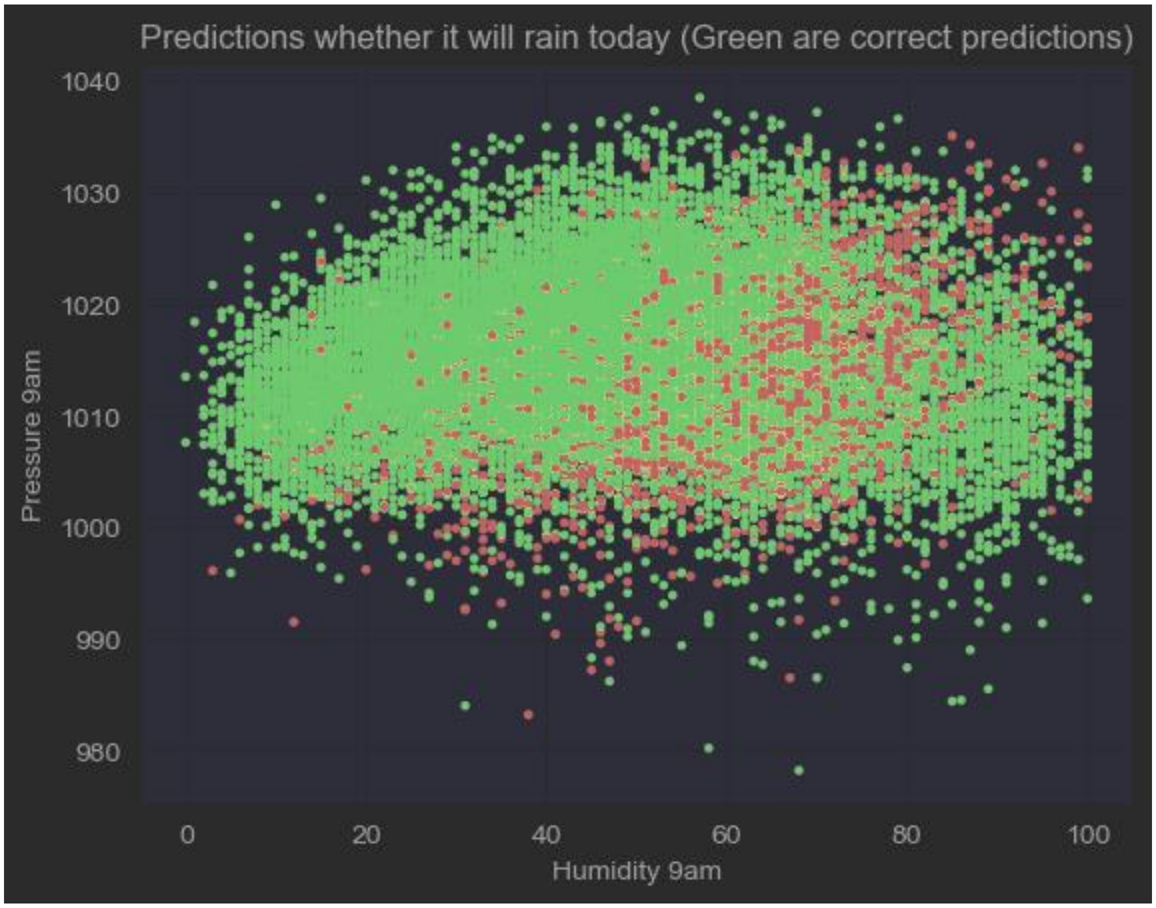
Linear Regression:

The first thing I tried doing was running a basic linear regression on the data. The accuracy ran at 32% but I think this was flipped, because the graph displayed the opposite at 66%. After investigation, I am not sure what is causing this error, Dataspell can be weird sometimes. I think Linear Regression is still the worst preforming of the three models as most of the variables involved in predicting have non-linear relationships. This is what an output of the score looks like:



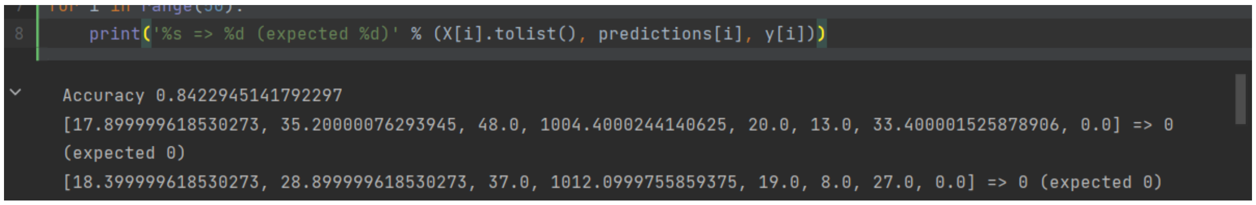
```
In 26    Lgmod.score(X_test,y_test)

Out 26     0.32894483595134205
```



Neural Network:

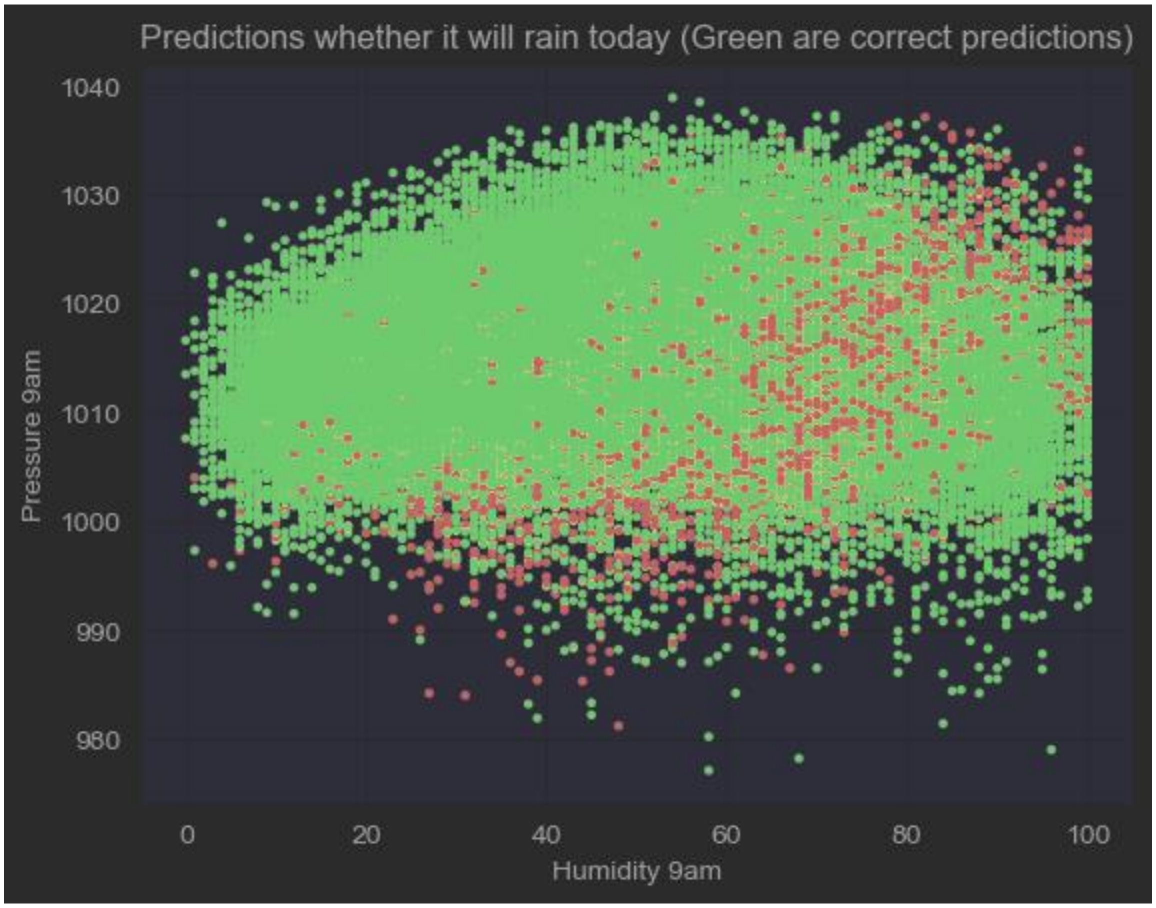
Running 100 epochs at a batch size of 10 took a while on my computer, I think if I specifically devoted more GPUs, this process might be shorter, but 12 mins every time I wanted to recalculate a model is a while. That being said, it gave a significantly high accuracy score.



```
for i in range(50):
    print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))
```

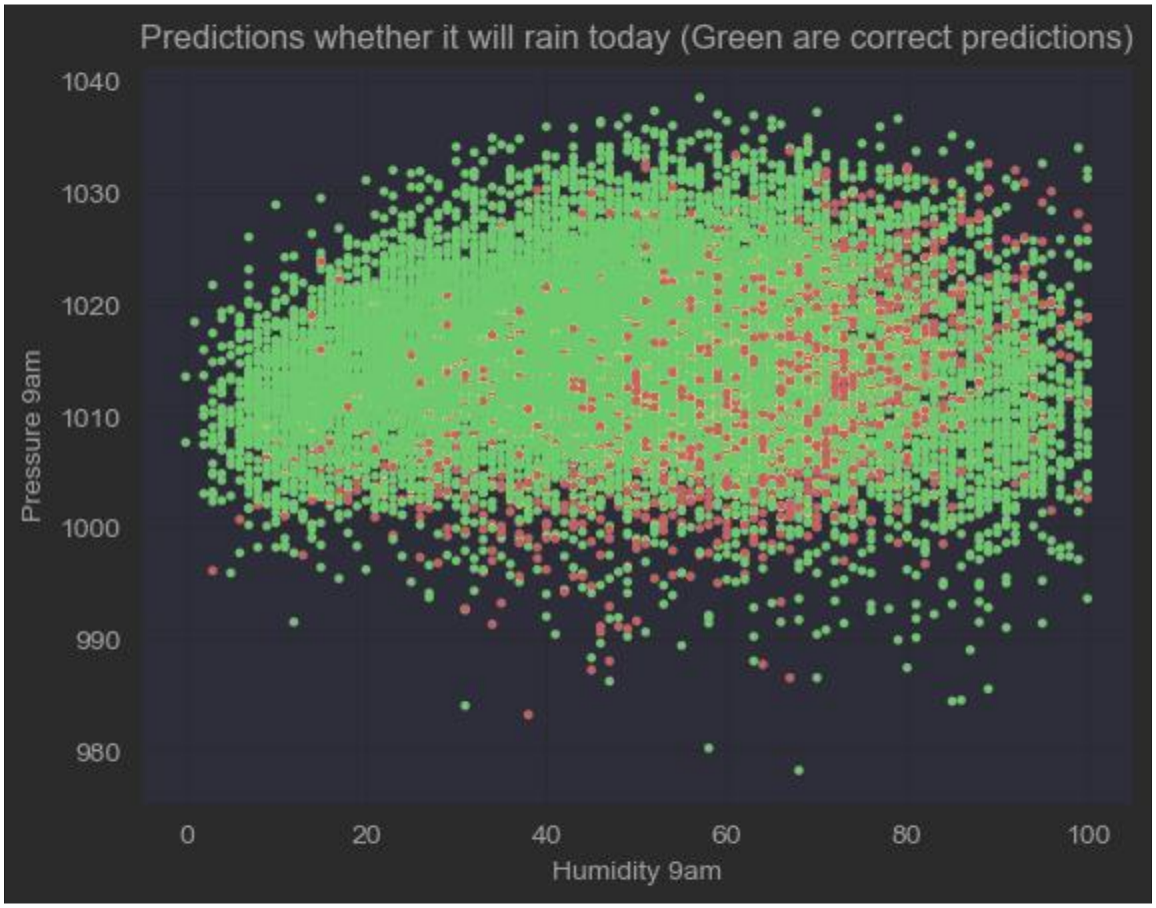
```
Accuracy 0.8422945141792297
[17.899999618530273, 35.20000076293945, 48.0, 1004.4000244140625, 20.0, 13.0, 33.400001525878906, 0.0] => 0
(expected 0)
[18.399999618530273, 28.899999618530273, 37.0, 1012.0999755859375, 19.0, 8.0, 27.0, 0.0] => 0 (expected 0)
```

84.22% is very high, but again you have to remember that the chance of rain is low and this is only guessing if it will rain. Here is a graph that somewhat illustrates the models accuracy using a variation of a graph I made in the analysis section of this paper. ‘



The model seems to be making the most incorrect guesses at the junction where raining and not raining are uncertain. If you remember the similar graph where red was representing when it does rain, most of the red occurs further to the right. So its incorrect guesses are basically where it is hard to determine the weather. This somewhat decreases my confidence that these predictions are useful.

Random Forest:



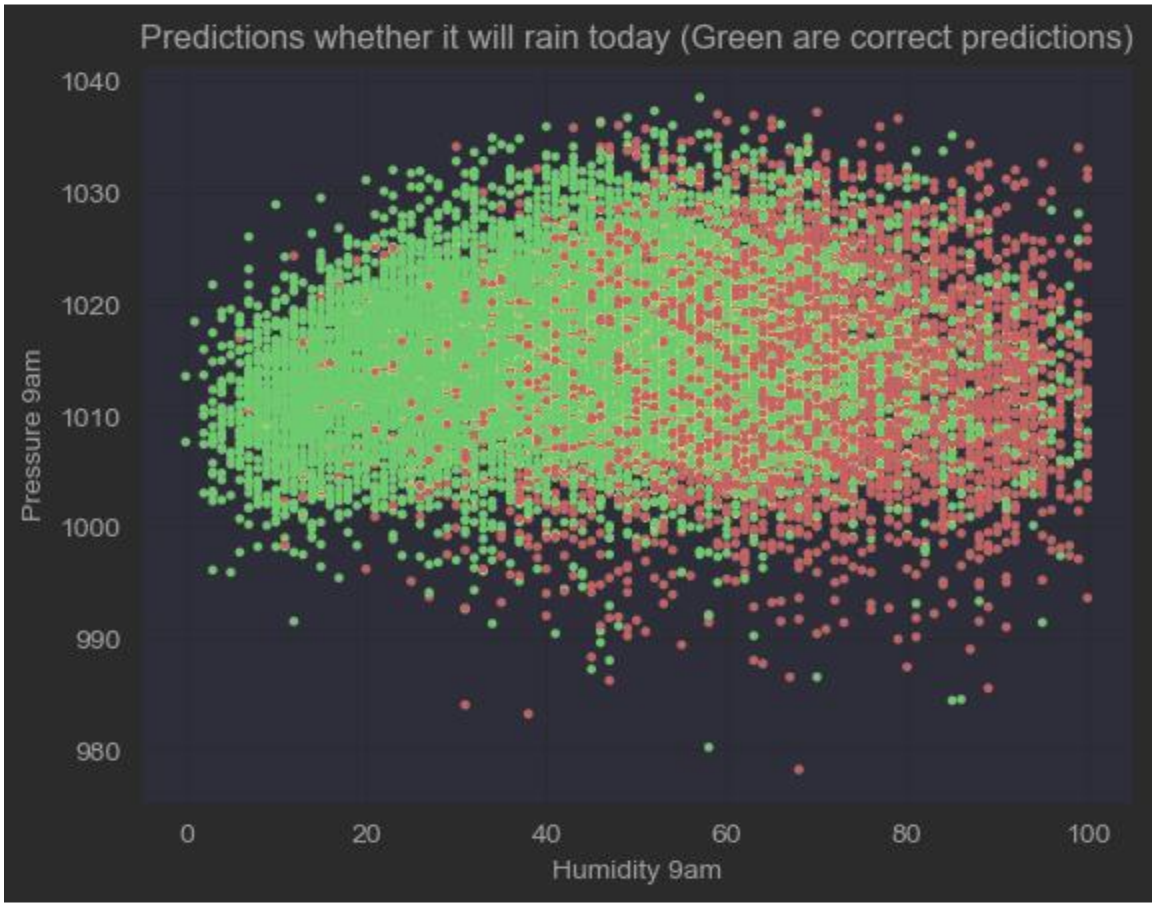
The Random Forest has a lot of predictions similar to the Neural network and has a similar level of accuracy. It also takes 5 seconds as opposed to 12 mins. I think for data that's in the 500k range Neural Networks would be able to handle better than Random Forest. 100k is still in the amount range where Forest can perform similarly at a fraction of the speed.

Conclusion:

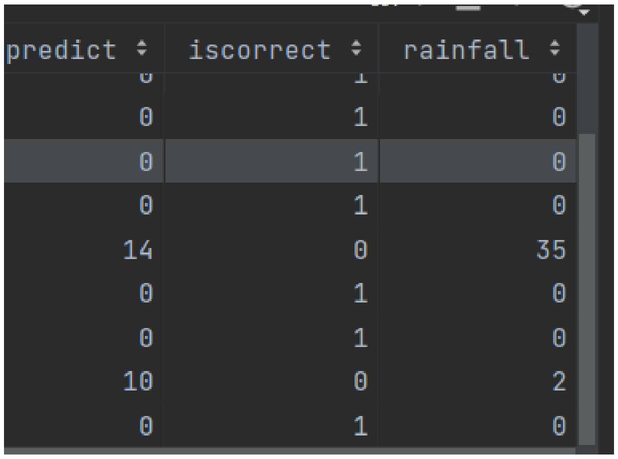
All the models except the linear regression in my opinion performed generally well. This may somewhat have to do with what's being predicted being a yes no answer that heavily favors no. The Random Forest and Neural Network both fared around 85% accuracy, but the Random Forest model was 2 seconds to what the Neural Network took quite a few mins to achieve. I think the neural network is slightly better than the Random Forest as it can handle more data better and I have plans on expanding the data set on this program in the future.

Part II question

What if we wanted to predict rainfall? I decided in the end only the Random Forest was needed to address this part.(I could not get the neural network to give me epochs losses that were not just 100 values.)A unique problem with predicting rainfall is that unlike predicting "Yes" or "No" rainfall is in amounts and there will probably never be an exactly correct prediction, this means there needs to be some level of acceptable error, I decided for this +/- 5 millimeters of rain will be our margin of error.

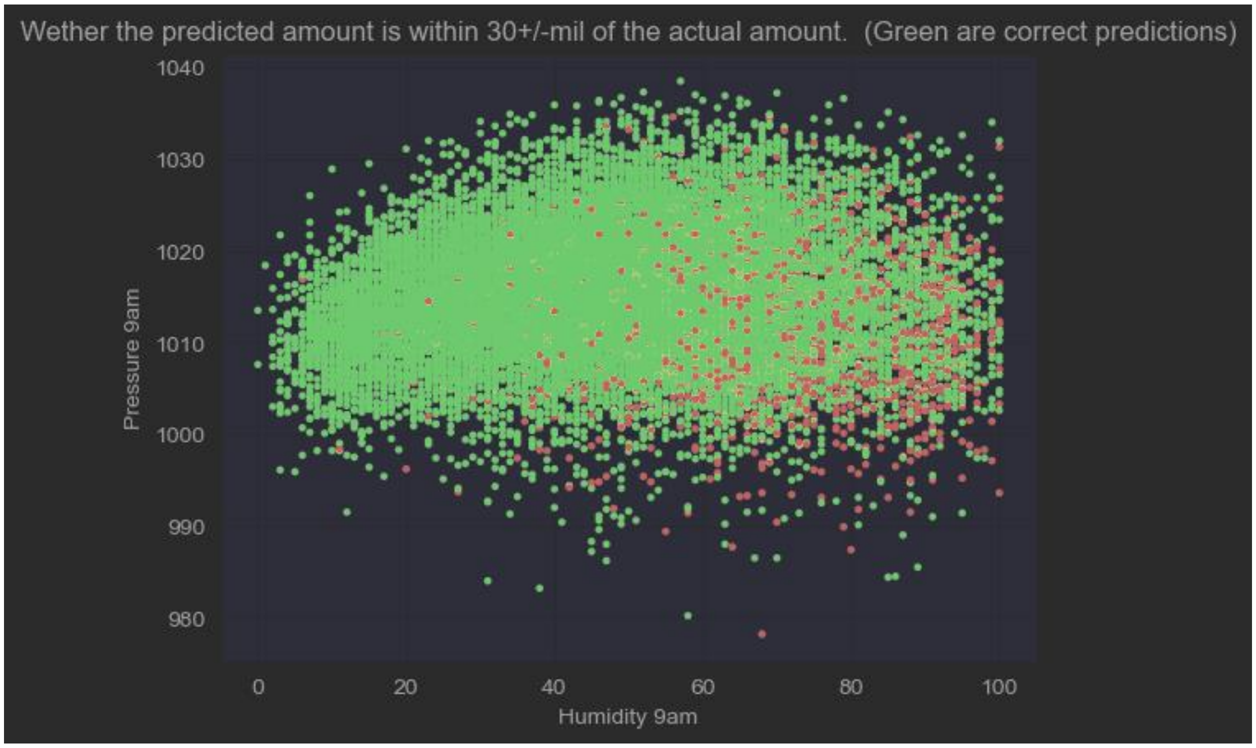


This seems like every time it wants to predict an exact amount of rain that isn't zero, it's wrong. Let's look at a table to see what is going on.

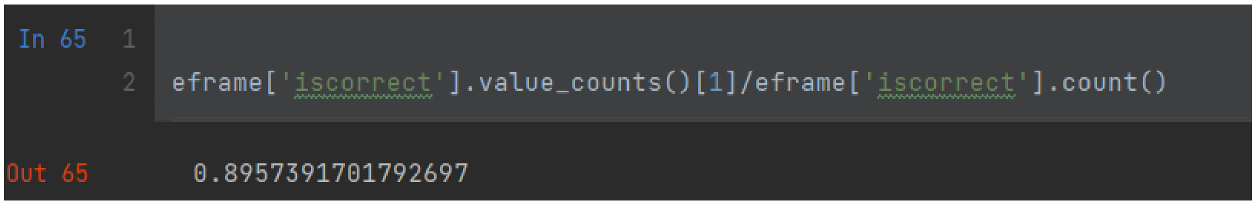


predict	iscorrect	rainfall
0	1	0
0	1	0
0	1	0
14	0	35
0	1	0
0	1	0
10	0	2
0	1	0

I don't think +/- 5 mil is a high enough margin. It looks like +/- 30 will probably give better results.



It gives a better graph, but I don't know if knowing the amount of rain within +/- 30 milliliters of certainty is all that useful for anything.



```
In 65
eframe['iscorrect'].value_counts()[1]/eframe['iscorrect'].count()

Out 65    0.8957391701792697
```

The model gives an accuracy score of 89.5%

Challenges:

Picking a research question: This is a lot harder than it seems, I wanted a data set that was 100k items, interesting to me, was easy to handle, and had relatively few data problems. It's a big ask of a dataset and I did not get exactly what I wanted. My first idea was using a Grib File, and while that may have outputted more concrete, geographically useful data, it was not something I could do with my current skill set at the time.

Reading: A lot of terminology in the weather model prediction sector of machine learning is pretty complex, and I still don't understand it completely. It's also using concepts from physics that I have not studied the math behind.

The actual code: There was a lot I wanted to do, but couldn't due to time limitations and lack of coding knowledge which wasn't remedied by research. For the stuff I managed to get done, there were a fair share of challenges as well. For example, the neural network took a couple of tries to get running without there being epoch losses of 100 points. Jupyter Notebooks is kind of weird in that if you don't order the cells right, it can often result in issues, especially with like 30 lines of ML code to keep track of in 8 cells. I didn't know how to set a +30/-30mm acceptable margin of error, so that took some Googling in order to setup and for the second part I forgot about label setting dat

Ethical recommendations:

It's hard to go about examining what the ethical commendations for a rain-based machine learning prediction program would look like. If the program becomes depended on by meteorologists or other relevant authorities and has significant flaws, this could result in property and infrastructure damage, and people being outside unnecessarily during dangerous weather conditions.

Assuming however the algorithm generally works well, it's a bit harder to narrow down potential ethical concerns. As the program is in Australia, water rights are under somewhat of a different legal purview than those of the US. 11.2% percent of Australian Water Rights are owned by foreign interests. 69.3% used for agricultural purposes, another 19% used for mining. Giving said companies a tool to tell where the better parts of Australia to purchase water rights are, could be deemed controversial and interpreted by Australian inhabitants as giving corporate interests water while a lot of them cannot make basic needs.

Aboriginal Australians have been historically subjected to diminishing water rights of the Murray Basin. They currently hold only 12.1 gigalitres of water in the basin which is 0.2% of total volume. The Aboriginals had 3 waves of said stripping of water rights, one at the end of the 19th century, the early 80s, and 2008-present. The earliest stripping of water rights, was when the Australian government tied water rights to owning land. The Aboriginals legally owned almost no land ergo no water rights. There was an attempted rectification with the NSW Aboriginal Land Rights Act (1983), but this act specifically gave Aboriginals land that was agriculturally poor and often had no water rights attached to it. Coming to the present day, the Murray Basin is in the process of drying out. It shrunk 17.2% from 2009 to 2018 at a rate of 2 gigalitres a year. The Aboriginal community haven't acquired any significant rights outside the basin, meaning the only water they do have significant rights could be gone. Until laws change to give Aboriginals back the water rights they lost, introducing a new ML tool for finding water could just exacerbate these problems.

Recommendations/Future Plans:

Further iron out code: Frankly I think the code I have is a bit choppy and some optimizing can be done to each of the models. I can also make a neural network, and linear regression for the rain amount prediction in question 2. If I could actually get a neural network with different layers and more epochs working, it would be interesting to see if I could get a high accuracy model that only requires a 5 mm buffer zone.

Better data set: I think there were quite a few variables my data set would produce better predictions with if it had. Specifically I was thinking about some kind of actual geographic identifying data, so I could actually produce more accurate predictions of where the rain could be. This would probably involve learning how to manipulate GRIB files which has proven to be difficult so far.

More runs: I've tested the sets with only on one random seed, using no seed(or multiple seeds) and taking the average of multiple runs will probably give a better idea of accuracy.

Learn More: I came into this project as a machine learning weather model novice, and there is still a lot I don't understand about the field. One specific thing that would help me is learning how to work pre-existing models like GraphCast. I'd also like to be able to code the Monte Carlo Simmulation I had planned, but couldn't carry out due to lack of understanding how the math applied with my variables or how to really set it up in Python.

References

In general:

My Data: Rain in Australia by Adam and Joe Young 2020:

https://www.kaggle.com/datasets/jsphyg/weather-dataset-rattle-package

Where the data set is sourced from:

Australian Government Bureau of Meteorology: About page:

http://www.bom.gov.au/inside/index.shtml?ref=hdr

Data page: http://www.bom.gov.au/climate/dwo/IDCJDW2801.latest.shtml

Prediction of Rainfall Using Intensified LSTM Based Recurrent Neural Network with Weighted Linear Units by M. Pushpalatha and S Poornima: https://www.mdpi.com/2073-4433/10/11/668

Rasp, Stephan, et al. "WeatherBench: a benchmark data set for data-driven weather forecasting." *Journal of Advances in Modeling Earth Systems* 12.11 (2020): e2020MS002203.

Lit Review:

Lam, Remi, et al. "GraphCast: Learning skillful medium-range global weather forecasting." *arXiv preprint arXiv:2212.12794* (2022).

[Prashant Banerjee]. ([2021 & Aug]). [Rain in Australia], [21 of 21]. Retrieved [9/3/2023] from [https://www.kaggle.com/code/prashant111/extensive-analysis-eda-fe-modelling].

Rasp, Stephan, et al. "WeatherBench: a benchmark data set for data-driven weather forecasting." *Journal of Advances in Modeling Earth Systems* 12.11 (2020): e2020MS002203.

Vandewege, Renny. “AusNet Embraces Data-Driven Storm Impact Modeling.” *POWER Magazine*, 1 Sept. 2023, www.powermag.com/ausnet-embraces-data-driven-storm-impact-modeling/.

Ethical Recommendations:

Davies, Anne. "Foreign Ownership of Australia’S Water Rights on the Rise." The Guardian, 30 Aug. 2023. https://www.theguardian.com/australia-news/2023/aug/31/foreign-ownership-of-australias-water-rights-on-the-rise

Hartwig, L. D. (2020, December 21). *Australia’s legacy of denying water rights to Aboriginal people*. Open Rivers Journal. https://openrivers.lib.umn.edu/article/australias-legacy-of-denying-water-rights-to-aboriginal-people/

EDA:

AUS BoM:
http://www.bom.gov.au/climate/sam/#:~:text=for%20western%20Tasmania.-,In%20eastern%20Australia%2C%20the%20southward%20movement%20of%20the%20westerly%20winds,increases%20rainfall%20for%20eastern%20Australia.

Source code: Written in Jupyter Notebooks using Data Spell: Cells indicated with line breaks.

```
#Cell 1
import os
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
import torch.optim as optim
# TensorFlow and tf.keras
import tensorflow as tf
import seaborn as sns

#Some scikitlearn stuff
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn import metrics
import plotly.express as px
# And the usual suspects
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn import preprocessing
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

```
aus_rain= pd.read_csv("weatherAUS.csv")
aus_rain.head()
```

```
aus_rain.shape[0]
```

```
aus_rain_clean = aus_rain.dropna()
aus_rain_clean.head()
```

```
aus_rain_clean.drop(columns=['Date',
'Location','WindGustDir','WindDir9am','WindDir3pm'], inplace=True)
```

```
aus_rain_clean['RainToday'] = aus_rain_clean['RainToday'].apply(lambda x: 0
if x == 'No' else 1)
aus_rain_clean['RainTomorrow'] = aus_rain_clean['RainTomorrow'].apply(lambda
x: 0 if x == 'No' else 1)
aus_rain_clean.head(150)
```

```
from matplotlib.colors import LinearSegmentedColormap

c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)
```

```
plt.scatter(aus_rain_clean['Humidity9am'],aus_rain_clean['Pressure9am'],s=5,
c=aus_rain_clean['RainTomorrow'],cmap=GnRd)
plt.xlabel('Humidity 9am')
plt.ylabel('Pressure 9am')
plt.title('Prediction whether it will rain tomorrow (3pm data) (Red is yes)')
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['Humidity9am'],aus_rain_clean['Pressure9am'],s=5,
c=aus_rain_clean['RainTomorrow'],cmap=GnRd)
plt.xlabel('Humidity 9am')
plt.ylabel('Pressure 9am')
plt.title('Prediction whether it will rain tomorrow (3pm data) (Red is yes)')
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['Humidity3pm'],aus_rain_clean['Pressure3pm'],s=5,
c=aus_rain_clean['RainTomorrow'],cmap=GnRd)
plt.xlabel('Humidity')
plt.ylabel('Pressure')
plt.title('Prediction whether it will rain tomorrow (3pm data) (Red is yes)')
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['Pressure3pm'],aus_rain_clean['Cloud3pm'],s=5,
c=aus_rain_clean['RainTomorrow'],cmap=GnRd)
plt.xlabel('Humidity')
plt.ylabel('Pressure')
plt.title('Prediction whether it will rain tomorrow (3pm data) (Red is yes)')
```

```
rom matplotlib.colors import LinearSegmentedColormap

c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clmod['Humidity9am'],aus_rain_clmod['Pressure9am'],s=5,
c=aus_rain_clmod['RainToday'],cmap=GnRd)
plt.xlabel('Humidity')
plt.ylabel('Pressure')
plt.title('Prediction whether it will rain today (Red is yes)')
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['Humidity9am'],aus_rain_clean['Pressure9am'],s=5,
c=aus_rain_clean['RainToday'],cmap=GnRd)
plt.xlabel('Humidity 9am')
plt.ylabel('Pressure 9am')
plt.title('Prediction whether it will rain today (Red is yes)')
```

```
plt.scatter(aus_rain_clean.Rainfall, aus_rain_clean.MaxTemp, s=100, c='blue',
alpha=0.5)

plt.xlabel("Rain amount(ml)")
plt.ylabel("Tempurature at max(Celsius)")
plt.title("Rain Amount vs Max Tempurature")
```

```
aus_rain_clmod = aus_rain.drop(columns=['Cloud9am', 'Cloud3pm'])
# aus_rain_clean.drop(columns=['Date',
'Location','WindGustDir','WindDir9am','WindDir3pm'], inplace=True)
aus_rain_clmod
```

```
aus_rain_clmod.drop(columns=['Date',
'Location','WindGustDir','WindDir9am','WindDir3pm','WindSpeed9am','Humidity9a
m','Pressure9am','Temp9am'], inplace=True)
aus_rain_clmod
```

```
aus_rain_clmod = aus_rain_clmod.dropna()
aus_rain_clmod.head()
aus_rain_clmod['RainToday'] = aus_rain_clmod['RainToday'].apply(lambda x: 0
if x == 'No' else 1)
aus_rain_clmod['RainTomorrow'] = aus_rain_clmod['RainTomorrow'].apply(lambda
x: 0 if x == 'No' else 1)
aus_rain_clmod
plt.figure(figsize = (10, 10))
sns.heatmap(aus_rain_clmod.corr(), cmap='RdBu', vmin=-1, vmax=1, annot=True,
            annot_kws={'fontsize':8, 'fontweight':'bold'}, fmt=".2f")
```

```

c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['Humidity9am'],aus_rain_clean['WindGustSpeed'],s=5
, c=aus_rain_clean['RainToday'],cmap=GnRd)
plt.xlabel('Humidity')
plt.ylabel('Wind Gust Speed')
plt.title('Prediction whether it will rain today (Red is yes)')
#%%
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['Humidity9am'],aus_rain_clean['MaxTemp'],s=5,
c=aus_rain_clean['RainToday'],cmap=GnRd)
plt.xlabel('Humidity')
plt.ylabel('MaxTemp')
plt.title('Prediction whether it will rain today (Red is yes)')
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['Humidity9am'],aus_rain_clean['MinTemp'],s=5,
c=aus_rain_clean['RainToday'],cmap=GnRd)
plt.xlabel('Humidity')
plt.ylabel('MinTemp')
plt.title('Prediction whether it will rain today (Red is yes)')
```

```
aus_rain_clean['pressured']=aus_rain_clean['Pressure3pm']-
aus_rain_clean['Pressure9am']
aus_rain_clean['Tempd']=aus_rain_clean['Temp3pm']-aus_rain_clean['Temp9am']
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(aus_rain_clean['pressured'],aus_rain_clean['Tempd'],s=5,
c=aus_rain_clean['RainToday'],cmap=GnRd)
plt.xlabel('Pressure derivitive')
plt.ylabel('Tempurature derivitive')
plt.title('Prediction whether it will rain today (Red is yes)')
```

```
class PimaClassifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.hidden1 = nn.Linear(8, 12)
        self.act1 = nn.ReLU()
        self.hidden2 = nn.Linear(12, 8)
        self.act2 = nn.ReLU()
        self.output = nn.Linear(8, 1)
        self.act_output = nn.Sigmoid()

    def forward(self, x):
        x = self.act1(self.hidden1(x))
        x = self.act2(self.hidden2(x))
        x = self.act_output(self.output(x))
        return x

model = PimaClassifier()
print(model)
```

```
loss_fn = nn.BCELoss()  # binary cross entropy
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

```
aus_rain_clmod
```

```
X_df=aus_rain_clmod[['MinTemp','MaxTemp','WindGustSpeed','Pressure3pm','WindS
peed3pm','Humidity3pm','Temp3pm','RainToday']]
y_df=aus_rain_clmod['RainTomorrow']
# X= torch.tensor(X.values)
# Y= torch.tensor(Y.values)
X = X_df.to_numpy()
y = y_df.to_numpy()
```

```

X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.float32).reshape(-1, 1)
X.shape
y.shape
```

```
n_epochs = 100
batch_size = 10

for epoch in range(n_epochs):
    for i in range(0, len(X), batch_size):
        Xbatch = X[i:i+batch_size]
        y_pred = model(Xbatch)
        ybatch = y[i:i+batch_size]
        loss = loss_fn(y_pred, ybatch)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
    print(f'Finished epoch {epoch}, latest loss {loss}')
```

```
y_pred = model(X)
accuracy = (y_pred.round() == y).float().mean()
print(f"Accuracy {accuracy}")

# make class predictions with the model
predictions = (model(X) > 0.5).int()
for i in range(50):
    print('%s => %d (expected %d)' % (X[i].tolist(), predictions[i], y[i]))
```

```
column_names=['MinTemp','MaxTemp','WindGustSpeed','Pressure3pm','WindSpeed3pm
','Humidity3pm','Temp3pm','RainToday']
X_back=pd.DataFrame(X.numpy(),columns=column_names)
X_back['predictions']=predictions.numpy()
X_back['actual']=y.numpy()
X_back['iscorrect']= X_back['predictions'] == X_back['actual']
X_back['iscorrect'] = X_back['iscorrect'].apply(lambda x: 0 if x == False
else 1)
X_back
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
c.reverse()
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(X_back['Humidity3pm'],X_back['Pressure3pm'],s=5,
c=X_back['iscorrect'],cmap=GnRd)
plt.xlabel('Humidity 9am')
plt.ylabel('Pressure 9am')
plt.title('Predictions whether it will rain today (Green are correct

predictions)')
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_df, y_df,
test_size=0.3, random_state=44)
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=50, max_features="auto",
random_state=44)
rf_model.fit(X_train, y_train)
```

```
eframe=X_test
eframe["rain_tomorrow"]=y_test
eframe['predict']=predict
eframe['predict'] = eframe['predict'].apply(lambda x: 0 if x < 0.3 else 1)
eframe['iscorrect']= eframe['predict'] == eframe['rain_tomorrow']
eframe['iscorrect'] = eframe['iscorrect'].apply(lambda x: 0 if x == False
else 1)
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
c.reverse()
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(eframe['Humidity3pm'],eframe['Pressure3pm'],s=5,
c=eframe['iscorrect'],cmap=GnRd)
plt.xlabel('Humidity 9am')
plt.ylabel('Pressure 9am')
plt.title('Predictions whether it will rain today (Green are correct
predictions)')
```

```
Lgmod=LinearRegression()
Lgmod.fit(X_train,y_train)
predict=Lgmod.predict(X_test)
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
c.reverse()
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(eframe['Humidity3pm'],eframe['Pressure3pm'],s=5,
c=eframe['iscorrect'],cmap=GnRd)
plt.xlabel('Humidity 9am')
plt.ylabel('Pressure 9am')
plt.title('Predictions whether it will rain today (Green are correct

predictions)')
```

#Question 2

```
X_df=aus_rain_clmod[['MinTemp','MaxTemp','WindGustSpeed','Pressure3pm','WindS
peed3pm','Humidity3pm','Temp3pm','RainToday']]
y_df2=aus_rain_clmod['Rainfall']
lab_enc = preprocessing.LabelEncoder()
y_df = lab_enc.fit_transform(y_df2)
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_df, y_df,
test_size=0.3, random_state=44)
from sklearn.ensemble import RandomForestClassifier
rf_model = RandomForestClassifier(n_estimators=50, max_features="auto",
random_state=44)
rf_model.fit(X_train, y_train)
```

```
predictions = rf_model.predict(X_test)
predictions
rf_model.score(X_test,y_test)
```

```
eframe=X_test
eframe["rainfall"]=y_test
eframe['predict']=predictions
eframe["iscorrect"]=eframe['predict'].between(eframe['rainfall']-30,
eframe['rainfall']+30,inclusive = False)
#eframe['iscorrect']= eframe['predict'] == eframe['rainfall']
eframe['iscorrect'] = eframe['iscorrect'].apply(lambda x: 0 if x == False
else 1)
```

```
c =
["darkgreen","forestgreen","green","limegreen","palegreen","salmon","lightcor
al","red","darkred","maroon"]
c.reverse()
GnRd=LinearSegmentedColormap.from_list('gr',c, N=256)

plt.scatter(eframe['Humidity3pm'],eframe['Pressure3pm'],s=5,
c=eframe['iscorrect'],cmap=GnRd)
plt.xlabel('Humidity 9am')
plt.ylabel('Pressure 9am')
plt.title('Wether the predicted amount is within 30+/-mil of the actual
amount.  (Green are correct predictions)')
```