Machine Learning Weather Prediction System Development Report

Developed by Jakub Jaroslaw Zimoch

Table of Contents

List of figures

Introduction

This report is a shortened version of a document produced for a dissertation. The goal of the project is to explore how modern Machine Learning compares to state-of-the-art weather prediction systems which utilize super computers and incredible amounts of data. While the project is not yet finalized, this report describes an overview of the problem, strategies, key milestones achieved so far, difficulties and considerations encountered at these milestones as well as future improvements/next steps.
Note, a longer version of the report is available on request. This is a shortened version to demonstrate key parts of the process in developing this project as the goal is to explore both the problem and potential solutions. If a solution is deemed to be effective, it is expanded and researched into further, the dissertation report itself is very long and in-depth, so in the interest of saving the readers time, only the milestones reached are presented. An extended version is available on request along with the source code exploring all the different models and solutions (20+ models/approaches).

Problem Introduction

Modern day weather forecasting is a highly complex task. Meteorological organizations around the world attempt to predict the chaotic nature of rain. While the details of strict models and/or techniques are not publicly available, the general state of weather predictions is only reliable for a few days at most. Anything beyond 4 days becomes a shot in the dark and often predicting the mean is just about as accurate as world class weather prediction systems using most modern technology. These systems often involve direct modelling of highly complex weather systems. The goal of this project is to use Machine Learning techniques along with data processing and feature engineering to attempt to predict weather through indirect modelling by capturing patterns hidden within temporal data.

Approach strategy

The approach to this problem is to firstly, analyse and process the data. Engineer any features that are not strictly within the dataset but might be useful based on knowledge of weather predictions. Build a model architecture and train the model. Analyse the performance metrics, note potential improvements and escalate the program complexity as necessary. If the solution shows no promise at all, it is discarded since the project is researched based. On the other hand, even if the solution is not perfect but shows potential it is expanded upon. Once the basic solutions are exhausted, based on ML knowledge, topic research as well as metrics recorded, the best solutions are then explored further.

Findings

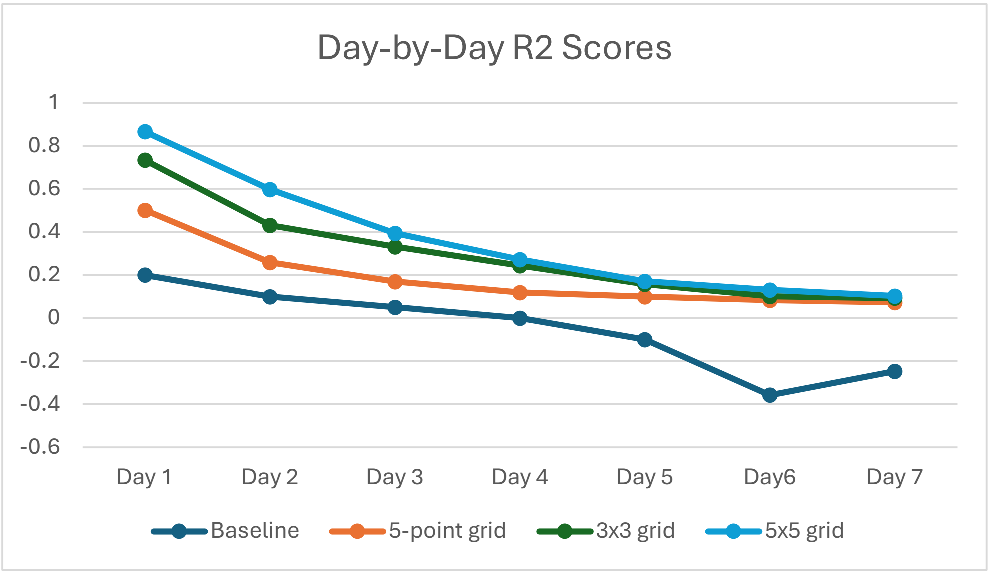

Figure 1

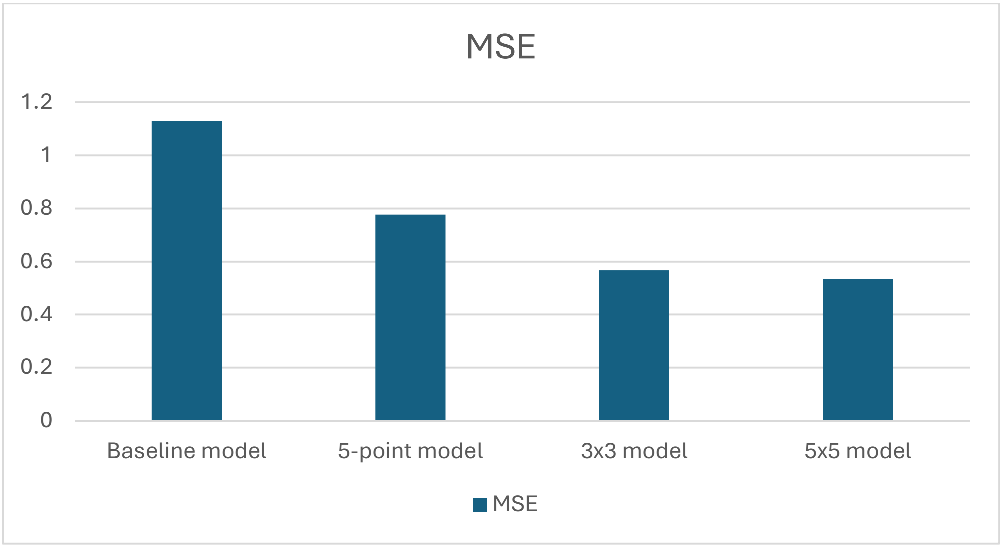

Figure 2

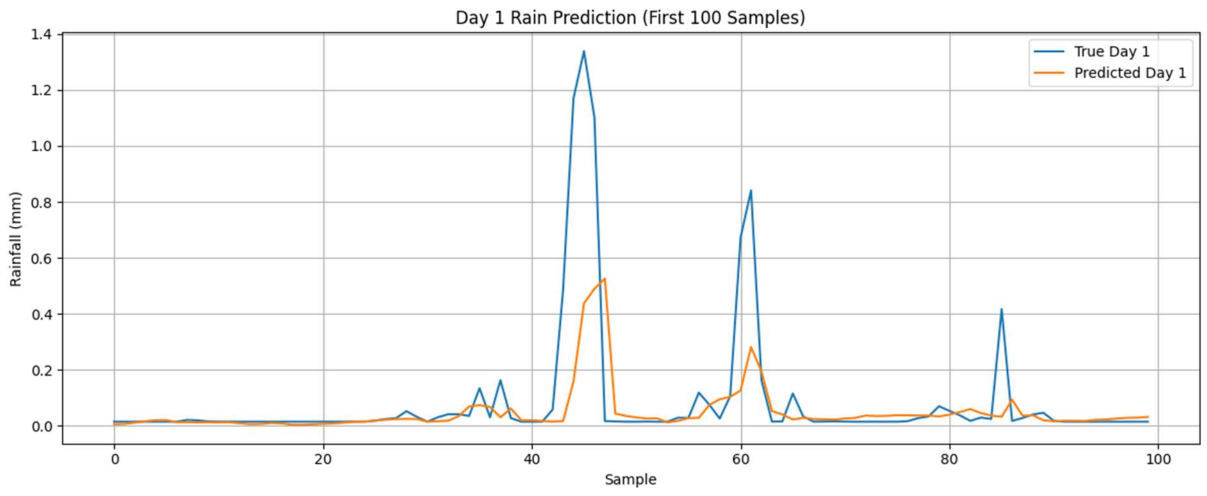

Figure 3

As seen in Figure 1, the 3 model iterations presented in this report are discussed below. These aren't the full progression in the interest of time but are the meaningful upgrades. To put simply, for the basic model a transformer was compared against other models, since it was the best option, it was then improved and expanded until the best performing version was found, the 5-point grid. That was then explored and exhausted and 3x3 grid was created. To push the bar further, the 5x5 grid is the most recent version of the model.

Model progression milestones:

Baseline overview:

A simple model approach. A single point on the map containing timestep, coordinates, u10, v10v d2m, t2m, msl, sp, tp. Tp is converted into mm from meters for readability and to avoid tiny numbers (8.583069e-06), and set as the y label, the other temporal features (not timestep, coordinates) are set as features. Baseline performance is measured in the form of R2 and MSE. A transformer is utilised due to its capabilities when dealing with temporal data.

5-point grid:

Geo-grid- Jumping couple of versions ahead; a geographical grid is implemented to attempt to capture spatial-temporal data interactions. Precipitation patterns are based around weather fronts. The implementation attempts to capture some of the front movements by giving nearby coordinates to predict rain within the centre. The points are to the north, south, east and west, (difference of 0.25 on the lat/long position). Figure 4 demonstrates the layout logic of each location that the model receives.

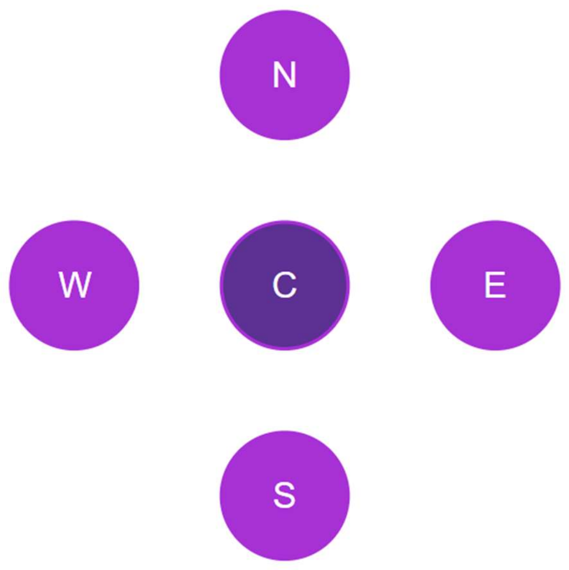

Figure 4

The hybrid approach- To further boost the performance of the model, a classifier is implemented. While the main task is to predict total tp amount, it is a much easier task to try and predict whether it will rain or not, than to predict the amount of rain. The classifier performs well against rain but poorly against no rain. The imbalance is handled with binary cross entropy however it still leads to the rain missing a lot of rain in favour of no rain. While the classifier performance at this stage is somewhat weak, see figure 5, it is important to note that it does improve the regressor performance metrics. Day 1 r2 improves by roughly 0.08-0.1 and is consistent across the first 4 days (weak supervision or auxiliary learning) showing that the hybrid model is successful.
Overall this results in an increase in performance metrics, however, upon looking at figure 3, it Is clear that the model still misses the magnitude of the spikes. It can identify IF it rains somewhat reliably but not how much. This is relevant as it shows a clear area of improvement for future hybrid iterations.

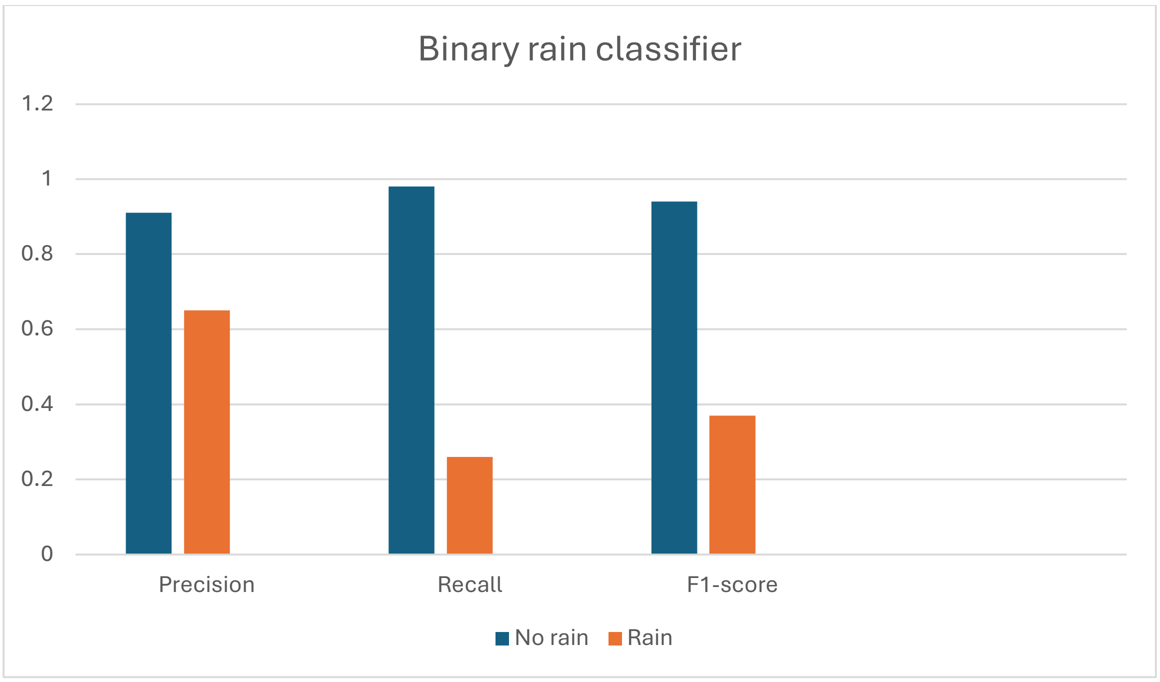

Figure 5

3x3 grid:

Next presented version is a much more complex variant. Rather than being a centre prediction point surrounded by 4 points at ‘nsew’ locations, the model becomes a full-on grid, attempting to capture weather fronts coming in from further away areas.

The classifier has also been upgraded to categorical instead of binary. It now predicts either 0.1, 0.5, 2.0, 10.0. Once again, the performance leaves a lot to be desired as seen in figure 6, and critically misses the heavy rain class which is usually a very bad sign, however once again, its purpose is to serve as auxillary learning and helps the model by reducing the mse floor and improving the r2 scores across the board. The weightings during training allow the biasing in loss_weights. There is one for tp_amount (the regressor) and rain_class (classifier). Once again the best performance coming in at 1:1 ratio, showing that the hybrid approach does in fact work regardless of the classifiers performance. It helps reduce the smaller errors at the very least however next point of improvement is to work on the classifier and strengthening it.

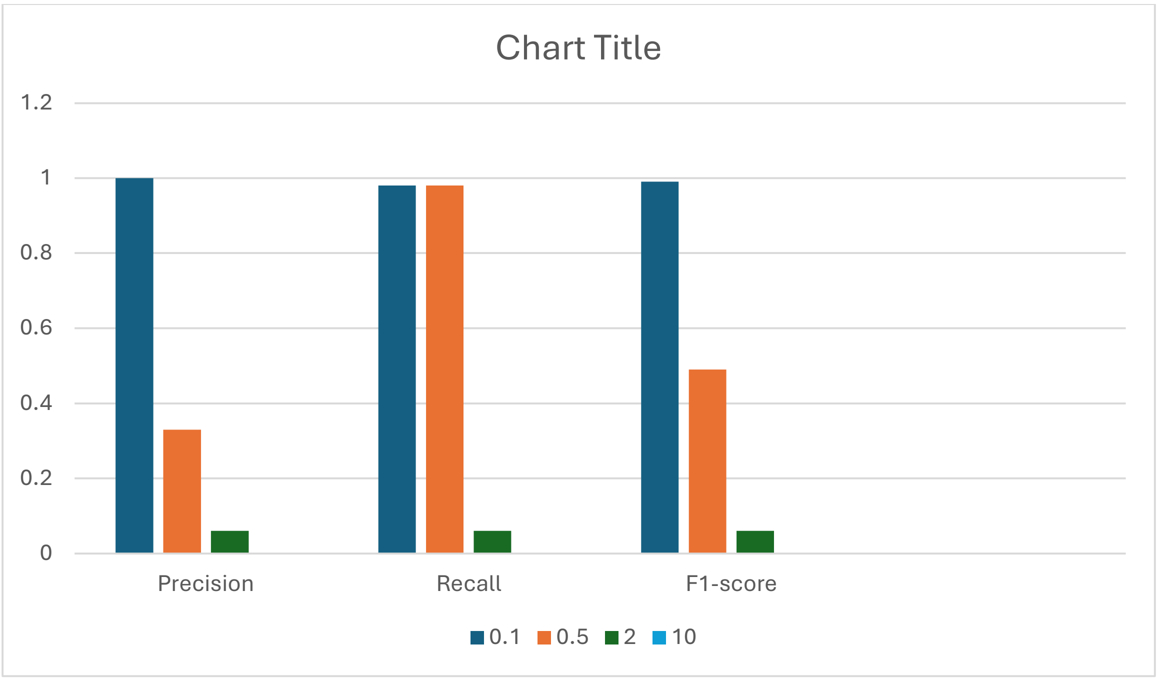

Figure 6

5x5 grid:

The most recent iteration and currently a work in progress. The grid has been expanded to 5x5 with the area covered being roughly 19354km2. Additional pipeline upgrades have been implemented. With 5x5 regions, it becomes an issue to code in regional names, joining the dfs and/or performing other operations that involve naming the regions. Instead a loop processes and names the dfs to cleanly join them together and make a clean dataset for the model to work with. This requires consistent naming conventions. The performance currently is not that much better than the 3x3 grid. The project will continue to analyse the issue and test the following most likely issues:

Due to the sequence length being too short, i.e. 7 hour-input window is not long enough to capture the front moving in from the further regions.
The model complexity is not high enough, the 3x3 grid having exhausted most of its capabilities.
The increased datapoints need more processing, perhaps frontal feature engineering should be revised.

Challenges and solutions

With a complex topic such as this there were many issues encountered along the way. Here are some of the most ML and CS specific issues encountered relevant to the milestones mentioned as well as the solutions with justifications.

Issue 1: memory errors

Understanding the error: The most obvious issue with a task as complex as this as well as the highly sophisticated architecture of the most recent model is memory. A key aspect of the project is to see what modern ML techniques can achieve as opposed to super-computers. The key difference is a consumer-level hardware vs a state-of-the-art machine. While this approach is limited in scope, that does not mean that the amount of data required is small by any means. The dataset is comprised of 7 years of hourly measurements of temporal atmospheric data. Just based on that alone a single point on the map contains 61392 rows and 10 columns. Features are engineered on top of that. Now, the basic iterations can handle it, but as soon as more points are introduced, memory errors become a reality very quickly. For example, this might involve, 25 points each containing over 61000 rows containing multiple features (10 starting columns plus lagged features, wind interactions, seasonal data etc). This quickly becomes an issue. Even through heatmaps and finding the best predictors, with a task as complex as rainfall in a chaotic environment such as a rainforest, there is no clear feature that can be singled out as the goal is to explore the possibilities of a model that can be handled on regular hardware.

Solution: Memory errors aren't solved by doing one thing, they are an on-going battle to maximise performance while minimizing the hardware demands. Some of the solutions and their justifications are-

- The sequence fed into the model (i.e. loaded into memory) quickly becomes Gbs. Sequence length was optimized for each model to strike a balance between metrics, memory demands and processing times.
- The sequence contains float numbers, the default type in python is float64, simply specifying float32 cuts the memory cost in half while still keeping enough accuracy. Float16 could also be specified however that would risk losing some accuracy.
- Heatmaps and feature importance was explored. Engineered features that showed no correlation were discarded. Key point, just because a specific iteration did not benefit from additional features, does not mean that the more complex iteration also would not. For this, it is important to consider the value ranges to assign them correct data types. For example, wind vector, does not need float32 as most points are within 3 decimal places.

Issue 2: Hybrid model challenges

Understanding the error: Tensorflow and Keras are both fantastic libraries but they come with many limitations and quirks. One such issue becomes apparent as soon as a hybrid approach is utilized. The dataset is highly imbalanced for the classification part of the hybrid. The clear solution is to weight the classes based on the disparity and keras allows that during training with a class weights argument. However, that is not applicable to hybrid models, as it cannot apply class weights to a regressor head.

Solution: There are many ways of fixing this, sampling would be an alternative method, however the class weights were implemented instead in a custom cross entropy loss function which is then given to the loss functionality when training the model. That way the model is penalized more for missing the minority classes. This approach is more robust as the dataset we have is plentiful and sufficient, it is not a lack of data in the smallest classes, but rather the high imbalanced ratio to other classes. The weights are dynamically calculated and used in the function based on the imbalance ratio between the classes.

Issue 3: Shape errors and modularity

Understanding the error: With a highly complex pipeline such as that in the 5x5 grid model, changing the pipeline will change the shape fed into the model. A hybrid architecture with multiple sequence lengths and inputs very quickly leads to errors down the line since different parts of the model expect different shapes and inputs. After establishing the structure and shapes, any changes to the pipeline will also change what the model gets vs what it expects to get.

Solution: Now, these errors are part of the development process however proactive steps can be taken to minimize them and/or make dealing with the easier. The 5x5 grid has multiple assertions at various phases of the pipeline to ensure that the shapes, scaling and data splitting are all done correctly. When creating the architecture, it is important to not hardcode any shapes so that they can be modular and should additional features be changed or removed, the program can dynamically adjust them within the code without having to manually search the pipeline and model architecture for the differences. Lastly strictly for testing purposes, the dataset can be quickly checked and sliced based off the shape mismatches and any duplicates and/or missing values, but it is important that those are dealt with properly once testing is complete.

Issue 4: Datasets acquisition and managements

Understanding the error: As mentioned in the report previously, the grid has been expanded over time. The era5 datasets can be obtained freely online upon request. With that being said, manually acquiring a single dataset takes a couple of minutes. It entails requesting the dataset on a website with features, location and a timestamp, followed by waiting for the processing, then downloading the data, unzipping, renaming and placing in a correct location. While not a difficult task, when downloading 25 files, this would take an extensive amount of time.

Solution: Not included in any of the model iteration files, instead in dataset_managment.ipynb, is a code of block that uses the cds api to cleanly, request data, extract the files and rename them correctly.

Next steps

The next steps that will be carried out before the investigation is completed and the project is finalized:

1. Explore the 5x5 grid further, including the feature engineering process, hyperparameter tuning as well as the architecture of the model. Explore the sequence length to see if we can collect interaction features further from the centre.
2. Improve on the classifier, by looking at the architecture as well as the sequence that is fed into it, this is one of the biggest areas of opportunities for improvement however, since classification serves as an auxiliary to the regressor, it is not a direct improvement.
3. Evalute the metrics thoroughly, report on them and conclude the best absolute model as well as chose and justify the best model based on performance compared to hardware demand.

Conclusion

Overall, so far the investigation has been successful in finding ways to improve the evaluation metrics of our rain prediction tasks. Various strategies were employed to escalate model complexity and capabilities however, it is becoming increasingly apparent that the principal of diminishing returns is starting to occur. While the complexity of the models has steadily increased and subsequently so has the data pipeline, steps were taken as necessary to make the data processing more streamlined, automated and modular. The project will continue to explore further possibilities.

Final note, source files as well as extra documents comparing various models and strategies are available on request, this is a large project with Gbs of data and so has been shortened down to present key findings so far.