Final Report

Introduction and Background

Flooding is a destructive force for communities. Because of abundant climate and weather data, it is difficult for humans to interpret this data to predict flooding. Machine learning (ML) provides new ways to utilize this data efficiently. Varying approaches including artificial neural networks, decision trees including random forests, long short-term memory (LSTMs), and support vector machines (SVMs) have been useful in flood prediction [1][2], and the accuracy of these is improved by optimizing the models and combining the models through algorithm ensemble [1].

Flood risk is often evaluated based on rainfall and water proximity. One of the datasets we will be using is the Georgia rainfall indicator that catalogs rainfall totals over monthly periods. We expect the anomaly feature will be utilized here (https://data.humdata.org/dataset/geo-rainfall-subnational). We will also utilize FEMA flood insurance information to help determine the location and extent of flooding. The latitude, longitude, and date of loss features will be used (https://www.fema.gov/openfema-data-page/fima-nfip-redacted-claims-v2). Note that these may change based on the status of the NCEI Archive which houses varying rainfall and flood disaster data sets and is currently inaccessible due to damage from Hurricane Helene.

Problem Definition

To address the need for flooding predictors, we will first compile different types of weather and climate data. We are looking at historical precipitation measurements and leveraging it as part of our dataset. Then, we will reduce the compiled data's dimensionality through principal component analysis (PCA) and finally, normalize the data to improve readability and clarity. We plan to analyze the data using LSTMs, SVMs, and random forest models. LSTMs will be useful to capture some longer term dependencies, making it robust for the time aspect of this study. Then, SVMs can be helpful for some nonlinear relationships. Finally, random forest models will delve through the complex relationships of the rainfall data and hopefully come up with a robust model. These each present their own unique advantages. Ultimately, the motivation for this project stems from wanting to predict floods before they occur, especially as soon as possible. This could prevent more future catastrophes and help increase evacuation times if needed.

Methods

To begin the preprocessing of our dataset, we removed unwanted features [2] and cleaned our data by filling in missing data points via interpolation. We then utilized principal component analysis (PCA), a dimensionality reduction technique effective in simplifying datasets by reducing the number of features while still maintaining and maximizing variance in the data to make patterns more discernible before applying a machine learning algorithm [4]. This also helps with preventing overfitting. Lastly, we normalized the data to ensure that no single feature dominates the model due to scale differences [5].

For a machine learning algorithm, we implemented long short-term memory (LSTM), a type of recurrent neural network effective for capturing sequential long-term dependencies in time series data[2]. Our dataset is a great example of this as it captures several predictors of flooding over a long period of time such as base flood elevation, lowest floor elevation, cause of damage, and precipitation. The LSTM model was tuned for different time steps and model depths to identify optimal configurations, aiming to capture temporal patterns effectively.

For a second machine learning algorithm, we implemented support vector machines (SVM), a perfect model for differentiating 2 classes with a hyperplane [9]. SVMs are designed to differentiate 2 classes, and since we have “flood” and “no flood”, it becomes the perfect model for our analysis. Here, we tried different kernel functions, tuned hyperparameters, and adjusted gamma all in an effort to make the best fit.

Finally, our last machine learning algorithm is random forest models. These models are all encompassing and great to research to round out the project because they have high accuracy, and are robust to overfitting and variance; in general, they are an evolution to decision trees that select a subset of feature splits [10]. Our model accounts for a lot of predictors involving flood elevation, precipitation, and lowest floor elevation, so a robust model is crucial to find. The robustness of random forest helps justify our other models by supporting their findings as well as just being well-rounded overall. For the random forest, we tuned depth and estimators to solve this classification problem and come to a final conclusion.

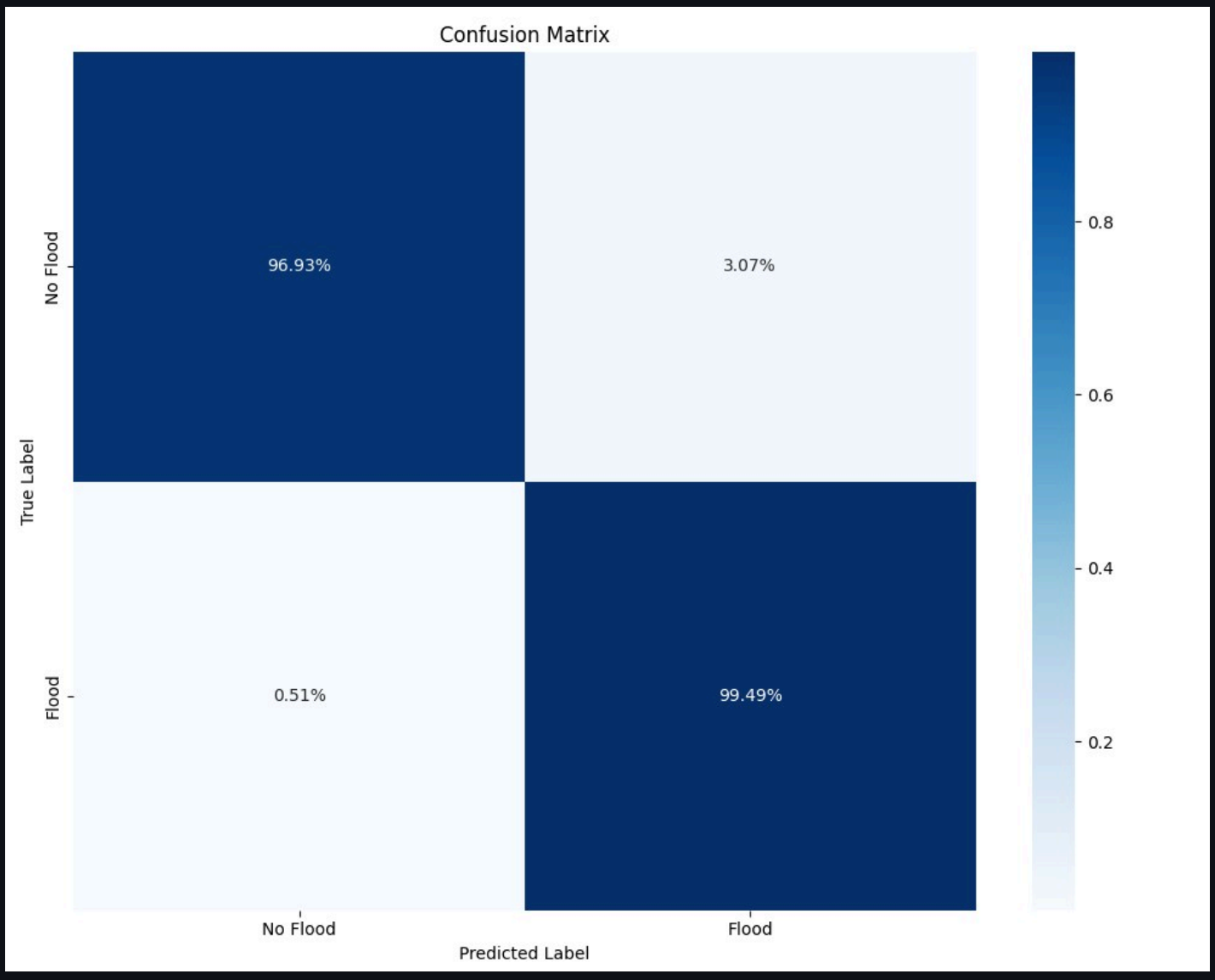
Results and Discussion

After running LSTM on our dataset, we achieved very high accuracy. In the classification report below, class 0 represents a “No Flood” prediction and class 1 represents a “Flood” prediction. As can be seen, the model predicted these events with 97% and 99% accuracy, respectively.

Classification Report:

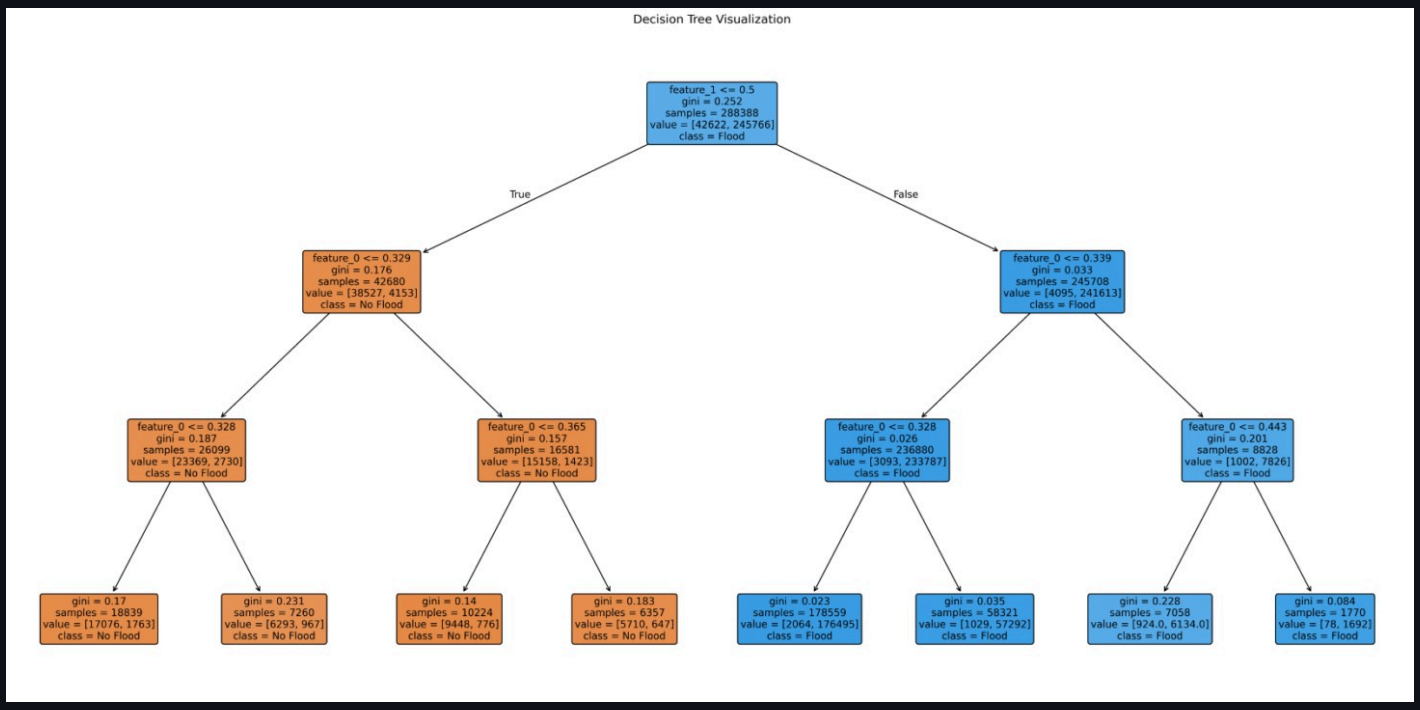
	precision	recall	f1-score	support
0	0.97	0.97	0.97	10666
1	0.99	0.99	0.99	61431
accuracy			0.99	72097
macro avg	0.98	0.98	0.98	72097
weighted avg	0.99	0.99	0.99	72097

We also produced a confusion matrix which shows that the model performs very well in predicting both events, successfully identifying “No Flood” events 96.93% of the time and “Flood” events 99.49% of the time. It only reported “Flood” events incorrectly as “No Flood” events 0.51% of the time and “No Flood” events as “Flood” events incorrectly 3.07% of the time. Overall, this model is very reliable in classifying between “No Flood” and “Flood” events.



The following decision tree model explains how the classification between "No Flood" and "Flood" events is made. At each decision point, the model splits the data based on specific feature values to separate the two classes, with each group classified according to whether it contains more "No Flood" or "Flood"

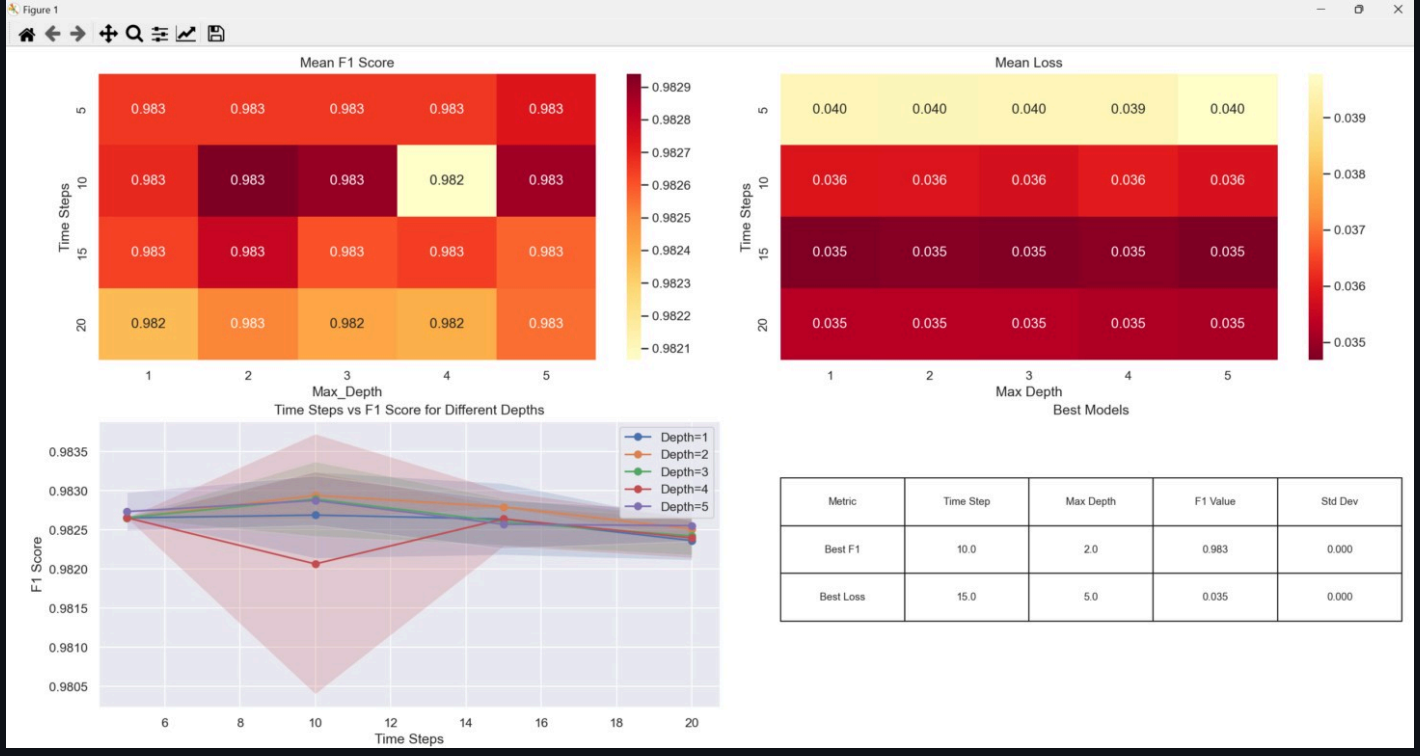
events. This process continues until the final classification at the leaf nodes, which represents the maximum depth that we set to prevent overfitting.



We also performed tuning analysis to determine the ideal number of time steps and depth to produce the best results. We measured mean loss, mean accuracy, and mean F1 score along with standard deviations of these metrics and found that somewhere between 10-15 time steps produced the best results. This, combined with good data preprocessing helped us achieve great results.

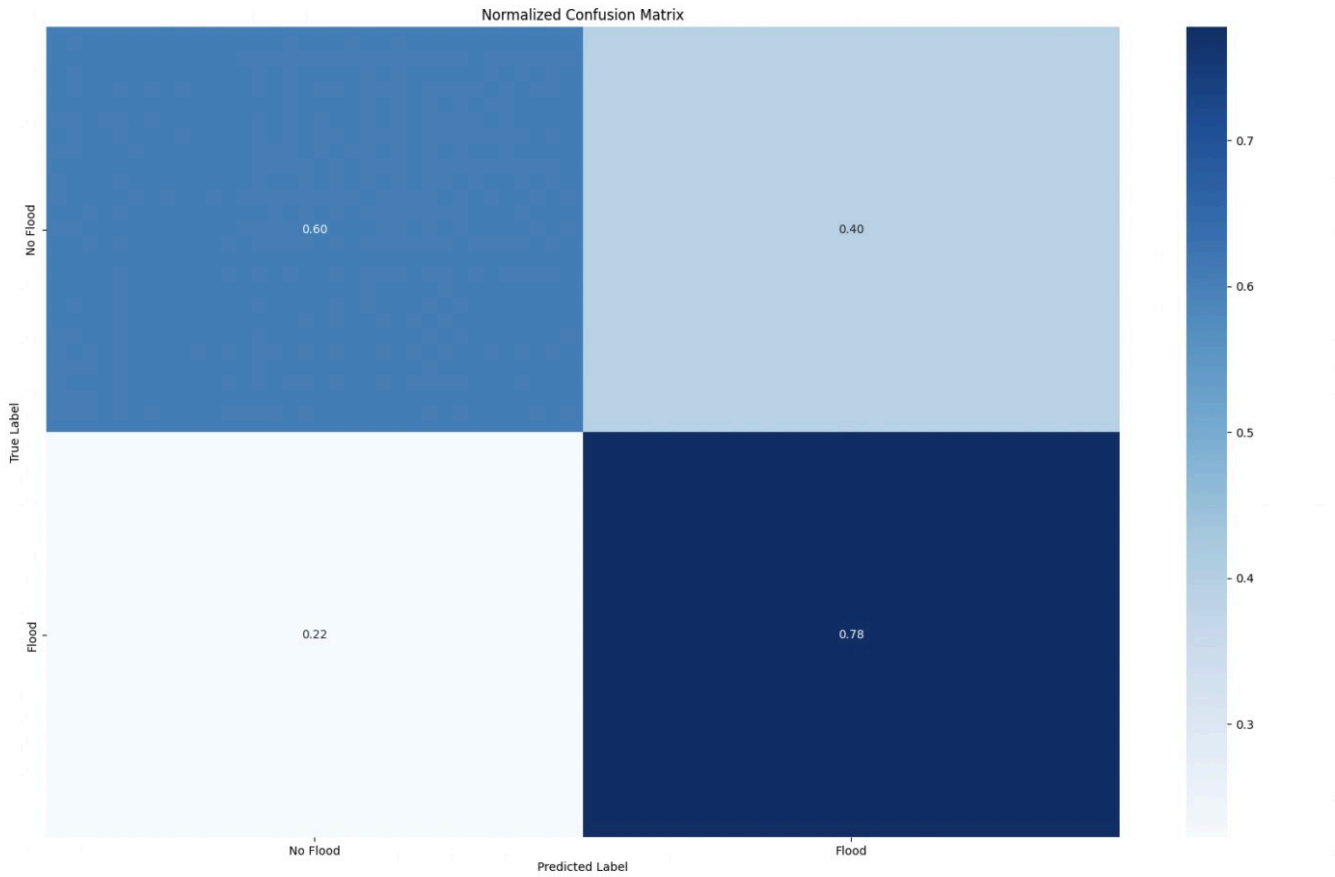
Detailed Results:

time_step	max_depth	mean_loss	mean_acc	mean_f1	std_loss	std_acc	std_f1
5	1	0.039517	0.991226	0.982653	0.000063	0.000007	0.000004
5	2	0.039524	0.991226	0.982653	0.000108	0.000007	0.000004
5	3	0.039515	0.991224	0.982656	0.000089	0.000009	0.000009
5	4	0.039497	0.991227	0.982654	0.000124	0.000007	0.000004
5	5	0.039783	0.991076	0.982734	0.000704	0.000442	0.000245
10	1	0.035869	0.991155	0.982689	0.000146	0.000133	0.000550
10	2	0.035854	0.991240	0.982940	0.000138	0.000071	0.000303
10	3	0.035766	0.991148	0.982894	0.000201	0.000240	0.000477
10	4	0.035984	0.991059	0.982065	0.000302	0.000298	0.001662
10	5	0.035798	0.991206	0.982876	0.000160	0.000063	0.000318
15	1	0.034705	0.991475	0.982638	0.000167	0.000088	0.000456
15	2	0.034769	0.991517	0.982794	0.000164	0.000009	0.000018
15	3	0.034740	0.991497	0.982602	0.000140	0.000042	0.000299
15	4	0.034840	0.991479	0.982640	0.000234	0.000063	0.000353
15	5	0.034685	0.991496	0.982571	0.000171	0.000044	0.000310
20	1	0.035295	0.991096	0.982363	0.000261	0.000035	0.000248
20	2	0.035304	0.991099	0.982512	0.000180	0.000031	0.000127
20	3	0.035294	0.991077	0.982423	0.000181	0.000115	0.000245
20	4	0.035196	0.991090	0.982397	0.000124	0.000029	0.000255
20	5	0.035146	0.991069	0.982554	0.000155	0.000035	0.000198



Overall, the LSTM model proved to be a highly effective solution for flood prediction in this context. In the coming weeks, we plan to explore the viability of support vector machines and random forest models in predicting flood events. In addition to these models that we'll explore, we'll also analyze how we can produce the best model for predicting flood events.

Next, for SVMs, the results for this model are unfortunately not the best. Whilst the physical results are not the best, it is still interesting to delve in why the results are this way and what we did to explore this aspect. First of all, for results:



In our produced confusion matrix, we identified “No Flood” events 60% of the time and “Flood” events 78% of the time. While these numbers are still quite high, for an accurate model, which we are trying to achieve, this is pretty low.

For some more of the numbers, our predictive performance f1-score is quite low, both below 90% where class 0 is a “No Flood” classification and class is a “Flood” classification. Now to analyze how we got here. Overall, we tried a ton of various methods to try and fit the model, but none of them produced much better results than these. First, the hyperparameters were tuned: the regularization value was tried at 0.001, 0.01, 0.1, 0.5, 0.9, 1, and 10, with the best at 0.9. The gamma value was also tuned at 0.01 and 0.1 where 0.1 produced the best results. Additionally, we tested different kernel functions to see if different kernels could fit the hyperplane better. We tested linear, rbf, and poly, and eventually found the best results with rbf. Finally, our last effort at fitting the model came from class balancing where we balanced the class distribution by oversampling the minority class. At the end, after optimizing everything we could, we ended up with these results. To conclude, the reason why our results are not the most optimal is most likely because the feature values between “No Flood” and “Flood” events were way too similar, making it much harder to fit the model accurately. This helps to expose a vulnerability with our results, but that is why we will look at this upcoming third model to finalize our results.

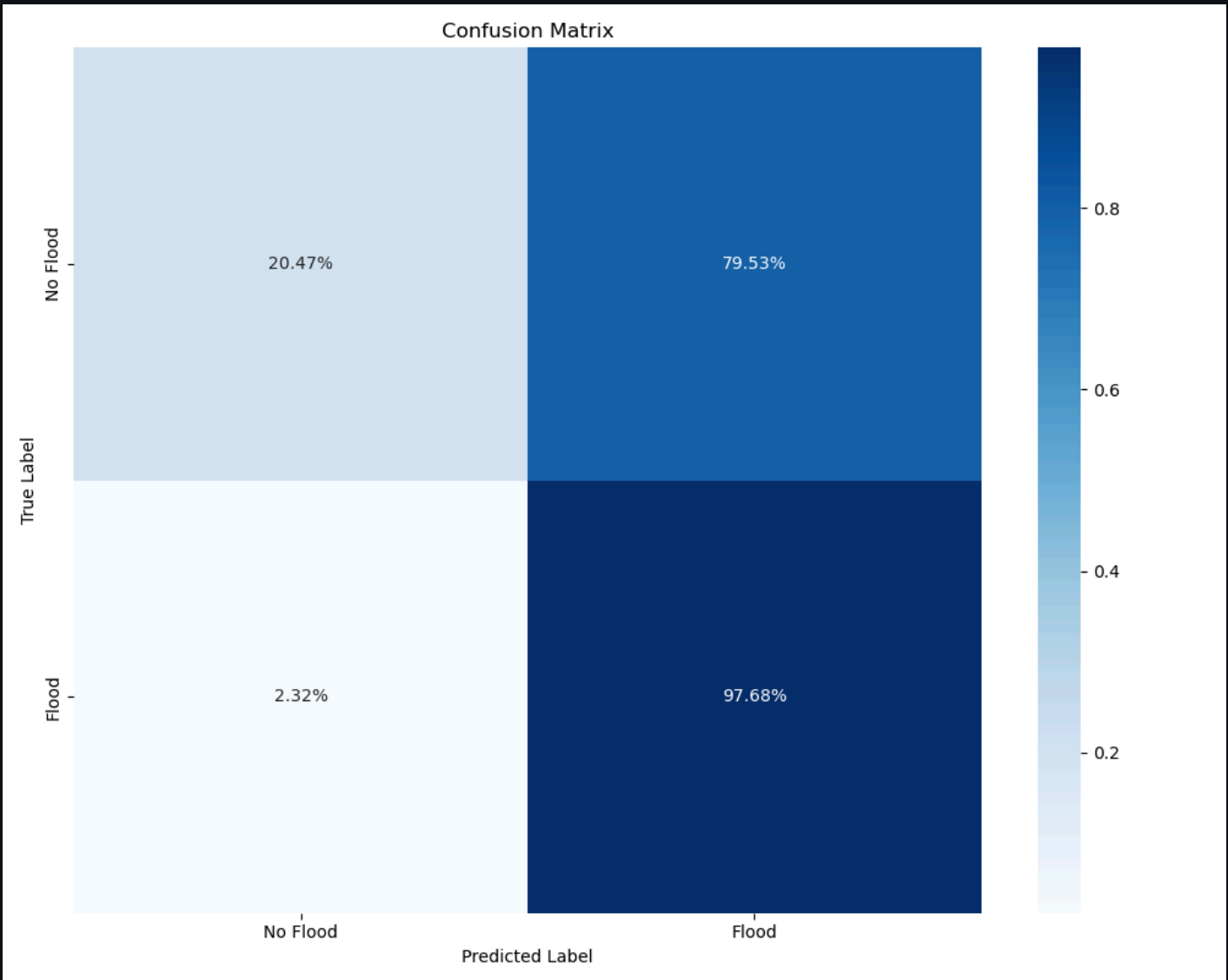
For our last model, we looked into random forest models. We got good results here with reasonably high precision and decent f1-scores, especially for class 1, our identification of “Flood”. These are high classification values and showcase the power of random forests as they fit the model pretty well.

Classification...

	precision	recall	f1-score	support
0.0	0.59	0.21	0.31	10666
1.0	0.88	0.98	0.92	61433
accuracy			0.86	72099
macro avg	0.74	0.59	0.62	72099
weighted avg	0.83	0.86	0.83	72099

Additionally, with an 86% accuracy, while not as strong as the LSTM, these are still very good results. Here, we tuned depth and estimators to get the best results. Ultimately, it was crucial to harness the random forest and scale the features to test the robustness of the model and showcase a model without overfitting.

These results are further justified by the confusion matrix where we identified “No Flood” events 20.47% of the time and “Flood” events 97.68% of the time.



The random forest showed how it can handle complex feature interactions and adjust to the nonlinear nature of the model as we discovered in SVMs.

Overall, LSTMs are our best performing model. This makes sense as it can capture some more of the temporal dependencies, which is crucial in flood scenarios. Random forest still did a good job as it handled the complex interactions from the various features passed in. Finally, it is crucial to note that SVMs were crucial in identifying the complex relationship between the classes as linear and polynomial classifications were not good enough to find the hyperplane. In general, it is crucial to note that while comparing these models, all three do come together to showcase the vulnerabilities as well as the successes of our study. In the end, LSTMs are indeed the best model in this situation.

Moving forward, the next steps of this project would be to leverage the successes we have and build on them. To see this in action, we would combine LSTMs and random forest into the same model. We can take advantage of the temporal aspect of LSTMs as well as its high accuracy, and then also leverage more of the robust nature of random forest and build an even more robust model than just LSTMs by themselves. This can help address the limitations of individual models and overall build a more comprehensive model to help predict floods.

References

[1] A. Mosavi, P. Ozturk, and K. Chau, "Flood Prediction Using Machine Learning Models: Literature Review," Water, vol. 10, no. 11, p. 1536, Oct. 2018, doi: https://doi.org/10.3390/w10111536.

[2] X.-H. Le, H. V. Ho, G. Lee, and S. Jung, “Application of Long Short-Term Memory (LSTM) Neural Network for Flood Forecasting,” Water, vol. 11, no. 7, p. 1387, Jul. 2019, doi: https://doi.org/10.3390/w11071387.

[3] The Click Reader, “Data Preprocessing in Python — Handling Missing Data,” Medium, Sep. 21, 2021. https://medium.com/@theclickreader/data-preprocessing-in-python-handling-missing-data-b717bcd4a264 (accessed Oct. 04, 2024).

[4] J. Murel, "What is Dimensionality Reduction? | IBM," www.ibm.com, Jan. 05, 2024. https://www.ibm.com/topics/dimensionality-reduction

[5] A. Bhandari, “Feature Scaling | Standardization Vs Normalization,” Analytics Vidhya, Apr. 03, 2020. https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

[6] S. Nevo et al., “Flood forecasting with machine learning models in an operational framework,” Hydrology and Earth System Sciences, vol. 26, no. 15, pp. 4013-4032, Aug. 2022, doi: https://doi.org/10.5194/hess-26-4013-2022.

[7] M. Khan, Afed Ullah Khan, B. Ullah, and S. Khan, “Developing a machine learning-based flood risk prediction model for the Indus Basin in Pakistan,” Water Practice & Technology, vol. 19, no. 6, pp. 2213-

2225, Jun. 2024, doi: https://doi.org/10.2166/wpt.2024.151.

[8] A. Bajaj, “Performance Metrics in Machine Learning [Complete Guide],” neptune.ai, May 02, 2021. https://neptune.ai/blog/performance-metrics-in-machine-learning-complete-guide

[9] Ibm. “What Is Support Vector Machine?” IBM, 13 Aug. 2024, www.ibm.com/topics/support-vector-machine#:~:text=27%20December%202023-,What%20are%20SVMs%3F,in%20an%20N-dimensional%20space.

[10] Ibm. “What Is Random Forest?” IBM, 25 Oct. 2024, www.ibm.com/topics/random-forest.

Gantt Chart

https://docs.google.com/spreadsheets/d/14JV9ATnk5HLtbtUmybd7S8qw2GmBIz_9/edit?usp=sharing&ouid=108468648048564250512&rtpof=true&sd=true

Contribution Table

https://docs.google.com/spreadsheets/d/1dApQIcvUl-MthNjfa1V3ELUzFv5_YK5NvMCyhfdKis0/edit?gid=810636487#gid=810636487