Jack McMorrow

Final Project - Individual Report

Introduction

For our project, we decided to develop various machine learning models to predict the occurrence of a tsunami after an earthquake. Tsunamis can cause millions of dollars of damage and take thousands of lives, so preparing for their arrival is very important. After an earthquake, many people start to let their guard down as they believe the catastrophic event has already passed, therefore not being prepared for the potential arrival of a tsunami. Developing a robust system that can predict and alert people when a tsunami is imminent is crucial to saving lives. We have seen devastating tsunamis in the past couple decades, including the Boxing Day tsunami in 2004 and the tsunami in Japan in 2011. In both of these cases, people were not properly prepared for the devastation caused by the tsunami. Much research has been done in recent years to predict tsunamis with machine learning, however these models are based off of complicated seismic data. In this project, we aimed to develop a model that could accurately predict a tsunami using relatively simple data.

This project consisted of data collection, which we sourced from Kaggle, followed by some minimal data preprocessing since the dataset was already in good shape. Next, we did some exploratory data analysis along with some data visualization to give us a sense of where these earthquakes occur. Once our data was ready for modeling, we implemented various machine learning models. This required lots of experimenting with libraries like scikit-learn and keras, as well as hyperparameter tuning. We compared the results of the various models and picked the model that had the highest F1 score as the final predictive model.

Individual Work

My responsibilities in this group project were mostly with the implementation of the machine learning models. After we had finished preprocessing the data, I first used scikit-learn to create a K-nearest neighbor classifier, a multilayer perceptron, a random forest classifier, XGBoost, and a support vector machine. Alex also created many of these models on his own, and we compared our results and ended up using the hyperparameters that worked the best between our projects. Additionally, I used keras to implement a multilayer perceptron and compared the results to the MLP classifier that we used in the scikit-learn package. In addition to the actual code, I also contributed to the development of the final report and class presentation.

Results of Individual Work

KNN

For the k-nearest neighbors, I experimented with different k values to see how different neighboring values would affect the accuracy of the model. Most of the k values were between the 80-90% accuracy model. Ultimately, we decided to use a k value of 11, which had the best accuracy without causing any overfitting with some of the lower values of k.

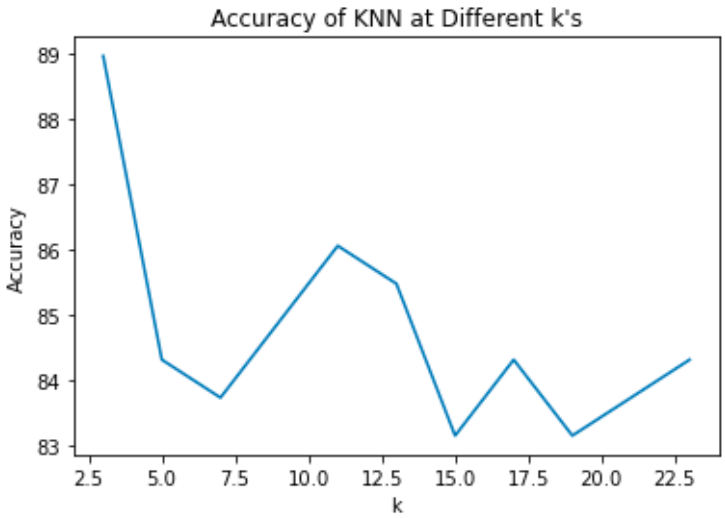

Figure 1. Accuracy of KNN model at different k.

MLP

For the multilayer perceptron in scikit-learn, I experimented with different hyperparameters, such as activation functions and solvers. Overall, we ended up with an F1 score of 0.8156. After performing the MLP classifier in sk-learn, I also implemented it in keras. It was a challenge to learn a new machine learning library while also trying to implement an accurate neural network. I initially built a three layer model, with the first two layers with 100 neurons and using the relu activation function. The final layer was one neuron that used the sigmoid function. This first run at the keras model had an accuracy of 84.3 percent. Next, I did a couple Grid Search CV to try to find the optimal hyperparameters and structure of the multilayer perceptron. This ended up with the final hyperparameters for the model:

```
Best parameters: {'activation': 'tanh', 'batch_size': 48, 'epochs': 200, 'neurons': 150, 'optimizer': 'adam'}
Best score: 0.947349
```

The best score on the cross validation for these hyperparameters was 94.73%. However, when this was applied to the test set the accuracy was about 86%. This is a minimal improvement from the original scikit-learn MLP. Although this work didn't result in a much better model, it was

definitely a good experience to understand the structure of the MLP as well as learning a new machine learning model. A downside of this approach was how long it took to run the Grid Search CV, which was about 45 mins for the two rounds that I performed.

Random Forest

I implemented a Random Forest Classifier using scikit-learn. With this model, I experimented with different cut off values using the predict_proba() function built into the function. I plotted how these cutoff values changed the accuracy and the recall rate of the model, as shown below:

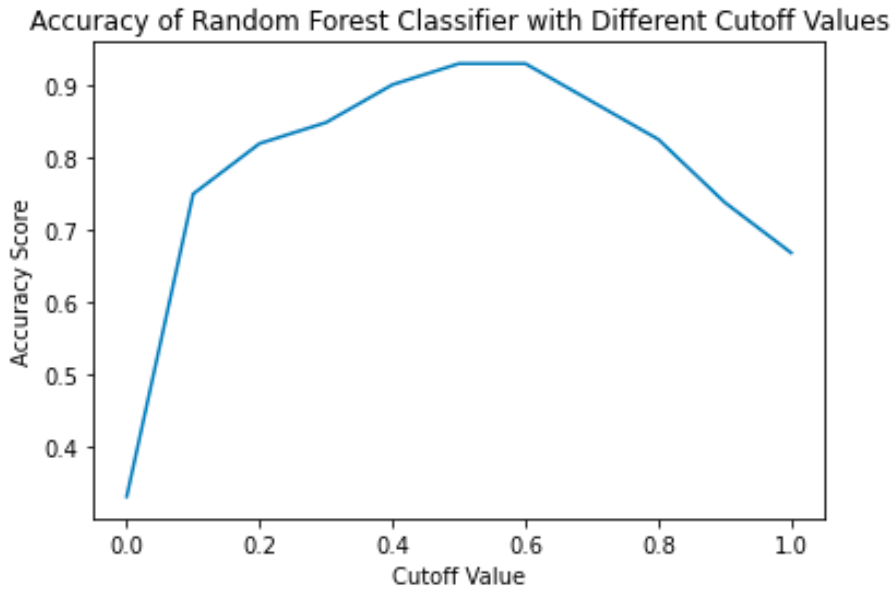

Figure 2. Accuracy of Random Forest Classifier at different Cutoff Values

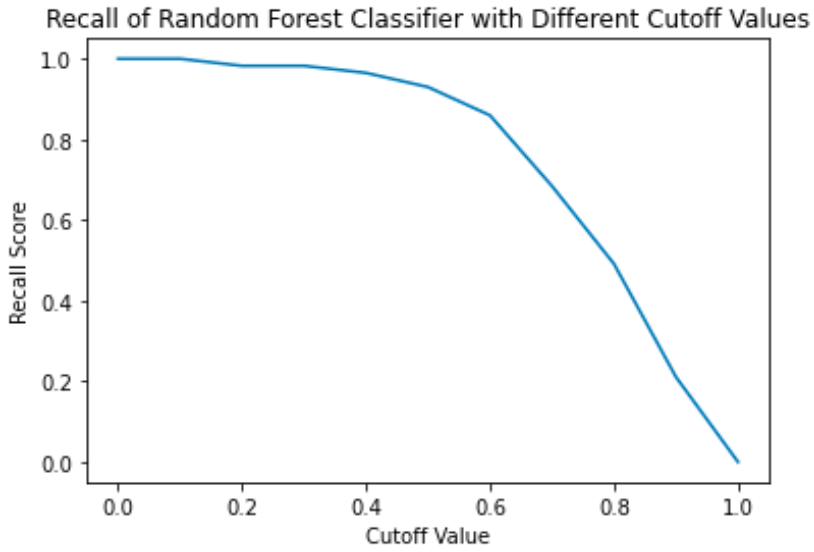

Figure 3. Recall of Random Forest Classifier at Different Cutoff Values

As shown above, the accuracy levels out at the 0.5 cut off value, while the recall rate remains high before falling off at around 0.6. Because of this, the default 0.5 cutoff value was what we decided to go with. However, I felt that this was worth investigation given the unbalanced nature of our data.

XGBoost and SVM

I also created an XGBoost model, which performed well with an accuracy of 0.91. The support vector machine performed lower at 0.84. We ultimately went with the models that Alex used for these two as that had included more extensive hyperparameter tuning. This was still a great way to initially compare the performance of different models.

Summary and Conclusions

Through our work as a group we were successfully able to build an effective model to predict the development of a tsunami after the event of an earthquake. Using simple seismic data,

these models could provide quick and accurate predictions for when a tsunami might occur, potentially saving lives.

There are some limitations to our study. First, we only have data on whether or not a tsunami occurred, not the location of the tsunami or the intensity of it. Sometimes earthquakes can trigger tsunamis that are only a couple of inches high and cause minimal damage. Being able to predict the severity of a tsunami could be even more beneficial. Secondly, while we have data for the location of the earthquake, the tsunami can occur in other places that were not close to the epicenter of the earthquake, potentially on different continents. Adding this information could improve the accuracy of our model.

Code Percentage: I used Chat GPT to help with the Grid Search on the MLP for my keras model. This ended up being about 4% of my total code.

References

Chamola, V., Hassija, V., & Gupta, S. (2020). *Disaster and Pandemic Management Using Machine Learning: A Survey*. NCBI. Retrieved April 29, 2023, from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8768997/

Chollet, F., & others. (2015). Keras. GitHub. Retrieved from https://github.com/fchollet/keras

Guha, S., K. Jana, R., & K. Sanyal, M. (2022, July 29). *https://www.sciencedirect.com/science/article/abs/pii/S2212420922004952*. Science Direct. Retrieved April 29, 2023, from https://www.sciencedirect.com/science/article/abs/pii/S2212420922004952

Kainthura, P., & Sharma, N. (2022, July 29). *Hybrid machine learning approach for landslide prediction, Uttarakhand, India*. https://www.nature.com/articles/s41598-022-22814-9. Retrieved April 29, 2023, from https://www.nature.com/articles/s41598-022-22814-9

Leventis, D. (2022, January 2). *XGBoost mathematics explained*. Medium. Retrieved April 30, 2023, from https://dimleve.medium.com/xgboost-mathematics-explained-58262530904a

Linardos, V., Drakaki, M., Tzionas, P., & Karnavas, Y. L. (2022). *Machine Learning in Disaster Management: Recent Developments in Methods and Applications*. MDPI. Retrieved April 29, 2023, from https://www.mdpi.com/2504-4990/4/2/20

NWS. (n.d.). *NWS JetStream - Tsunami Generation: Earthquakes*. National Weather Service. Retrieved April 29, 2023, from https://www.weather.gov/jetstream/gen_earth

scikit learn. (n.d.). *scikit.learn manual*. scikit-learn: machine learning in Python — scikit-learn 1.2.2 documentation. Retrieved April 29, 2023, from https://scikit-learn.org/stable/index.html

Pupale, R. (2019, February 11). *Support vector machines(svm) - an overview*. Medium. Retrieved April 30, 2023, from https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

Yousefi, S., & Reza Pourghasemi, H. (2022, July 29). *A machine learning framework for multi-hazards modeling and mapping in a mountainous area*. https://www.nature.com/articles/s41598-020-69233-2. Retrieved April 29, 2023, from https://www.nature.com/articles/s41598-020-69233-2