Prediction of Precipitation in New York City and Boston Using Machine Learning Models

EAEE E4000: Machine Learning for Environmental Engineers
Fall 2022

Final Project Report

Joy Fan (jjf2179)

Abstract

This project evaluated the predictive accuracy of three machine learning models: Neural Network (NN), Transfer Learning, and Extreme Gradient Boosting (XGBoost). The predictands of the models were daily rainfall in New York City (NYC) (zip code 10027) for NN and XGBoost, and the predictand of the transfer learning model was daily rainfall in Boston (zip code 02138). Hyperparameters were tuned manually to optimize model performance, and resulting root-mean-squared-errors (RMSEs) were compared.

Introduction

Climate change has severely affected weather patterns, most noticeably so in the past few decades, as there have been changes to the intensity and frequency of precipitation [1]. This comes as a result of an increase in temperature of the oceans, which results in an increased amount of evaporation [1]. This increased evaporation and resultant condensation then may manifest as storm systems over land, which may cause a rise in precipitation. Potential impacts of this increase in precipitation can include serious soil erosion, mudslides, flood risks, and potential implications of these impacts may include injuries, adverse effects on health (i.e. impaired water quality), and even death [2]. As a result, it becomes imperative for precipitation to be better understood and predicted, so that preventative measures against these deleterious effects may be taken.

As various organizations improve the accessibility of accurate publicly accessible datasets, and with the development and advancement of machine learning (ML), there has been an increase in ML approaches to modeling and predicting weather-related phenomena in the past decade [3]. Specifically, this project sought to utilize NN, Transfer Learning, and XGBoost to predict daily rainfall in NYC (zip code 10027) and daily rainfall in Boston (zip code 02138), using predictors of temperature, humidity, cloud cover, and sea level pressure (pressure), for NYC and Boston, respectively. Given that there was a greater amount of data for the predictors and predictands for the NYC zip code, the NN model and XGBoost model were trained utilizing the NYC dataset, whereas the Transfer Learning model leveraged the pretrained NN model using the NYC dataset in conjunction with the limited data available for the Boston zip code. The results from the Transfer Learning model were also compared with the results from a NN model utilizing only the limited Boston dataset.

Data

The data was sourced using the Weather Query Builder available on the Visual Crossing database [4]. This dataset was selected, as it provided the greatest variety of predictors to choose from, and it also had a relatively user-friendly user interface.

Two separate CSV files were downloaded from the Weather Query Builder. Both datasets are available within the Github repository "EAEE-E4000-Final-Project" under jjf2179. The Data.csv file contains weather data from January 1, 2014 - January 1, 2023 for 10027 (NYC, New York zip code). The TransferData.csv contains weather data from January 1, 2022 - December 23, 2022 for 02138 (Boston, Massachusetts zip code).

The data for both files was preprocessed to isolate the selected predictors and predictands and to normalize the data. The columns "temp", "humidity", "cloudcover", "sealevelpressure", and "precip" were selected from the overall dataset and combined to form a new dataframe. Presence of null values were checked, and the following visualizations of the data can be found below in Figure 1 and Figure 2. The transparency of the points, alpha, is set to 0.1, so that areas with denser data can be visualized.

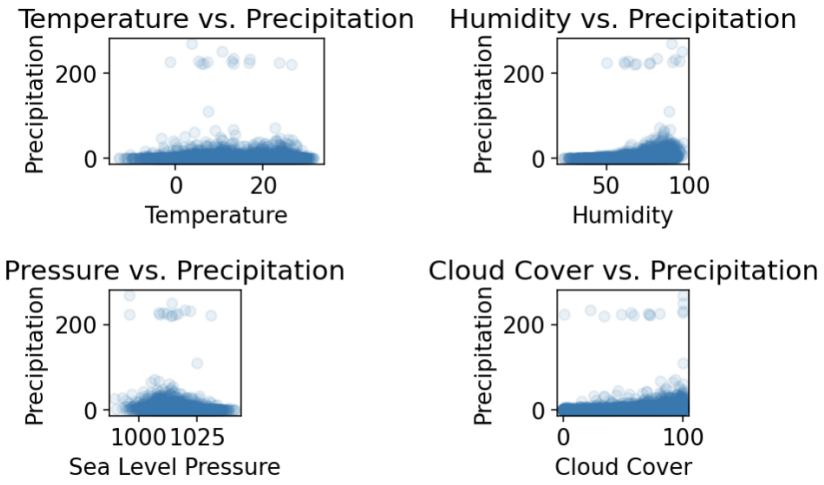

Figure 1. The relationship between the precipitation and individual predictors in NYC can be visualized in the 2x2 plot above.

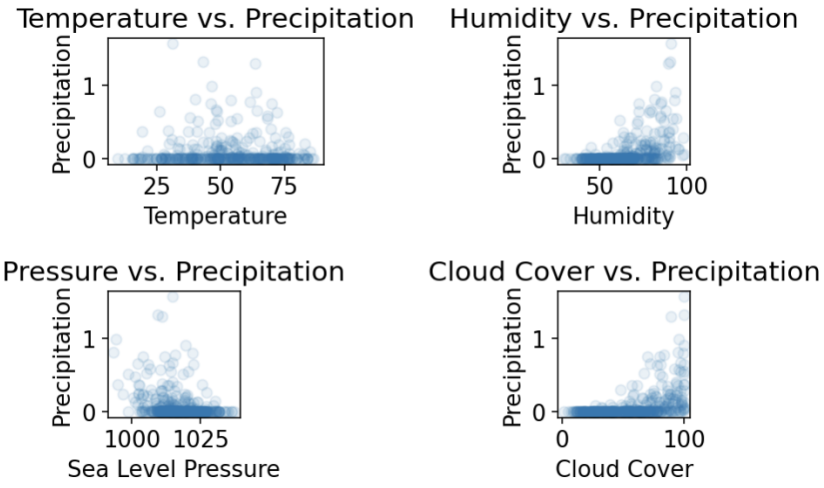

Figure 2. The relationship between the precipitation and individual predictors in Boston can be visualized in the 2x2 plot above.

As there were many large deviations from the most densely populated region of data, data was normalized by subtracting the mean and dividing by the standard deviation. The visualization of the normalized data is similar to the visualization in Figure 1 and Figure 2 above, with the main difference being that the scales of the x- and y-axis were adapted to reflect the normalization. This can be observed in Figure 3 and Figure 4 below.

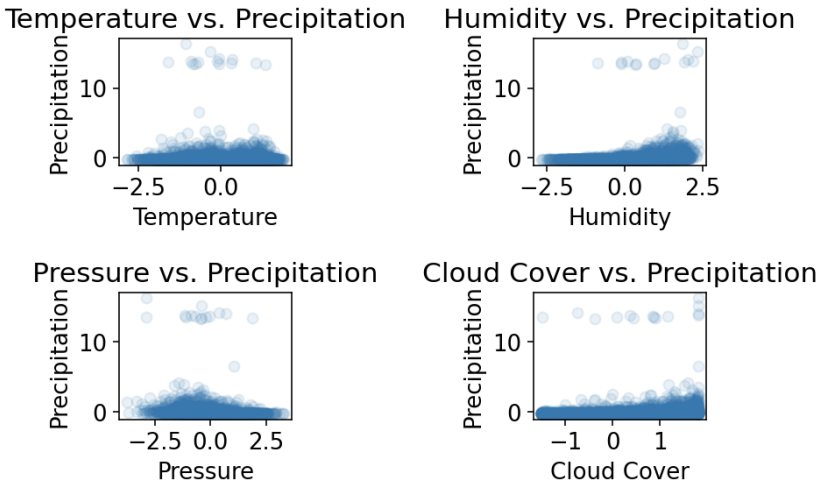

Figure 3. The normalized relationship between the precipitation and individual predictors in Boston can be visualized in the 2x2 plot above.

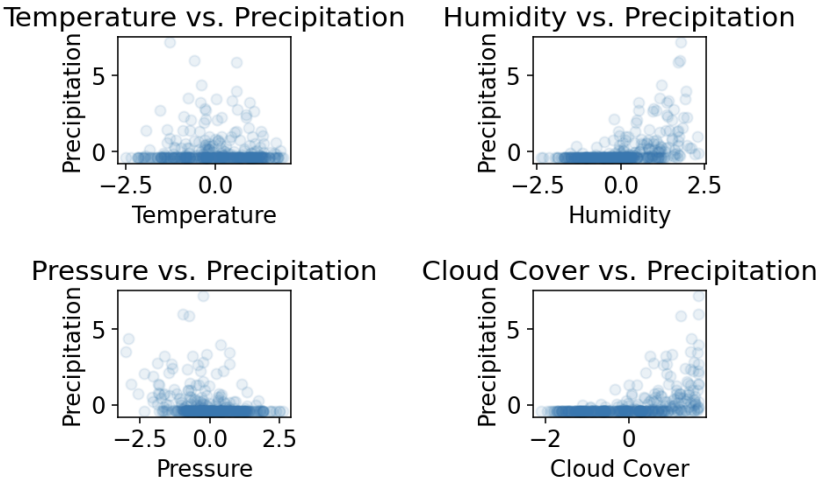

Figure 4. The normalized relationship between the precipitation and individual predictors in Boston can be visualized in the 2x2 plot above.

The data was then split for both dataframes into training and testing datasets. To ensure that there was a representative distribution of data between the two datasets, this split was also visualized and can be observed in Figure 5 and Figure 6 below, with the blue dots representing the training dataset and the orange dots representing the testing dataset. As can be seen when comparing Figure 5 and Figure 6, there is a significant difference in the amount of datapoints available between NYC and Boston. This is because there is nine times more data in the NYC dataset than the Boston dataset. As a result, Figure 6 appears to be more sparse than Figure 5, but both show a relatively even and representative distribution of data. The data between training (consisting, at this stage, of training data and validation data) and testing data was split in a 3:1 ratio.

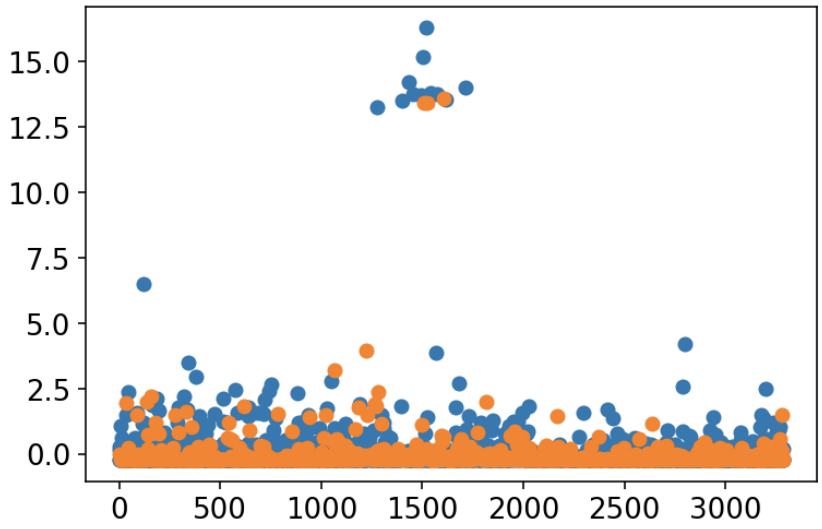

Figure 5. Training and Testing datapoint split for the NYC dataset, with blue representing training data and orange representing testing data.

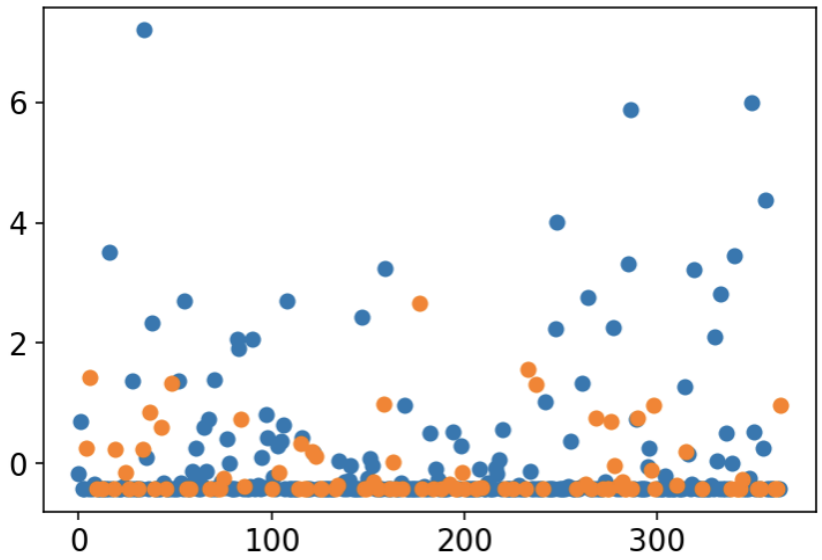

Figure 6. Training and Testing datapoint split for the Boston dataset, with blue representing training data and orange representing testing data.

Methodology and Implementation

Neural Network – NYC and Boston

An NN model is a subset of ML, with the structure of an NN mirroring that of the human brain and its neurons. NNs can have multiple layers, consisting of an input layer, one or more hidden layers, and an output layer, with each layer containing a number of nodes [5]. A depiction of the aforementioned structure is shown below in Figure 7.

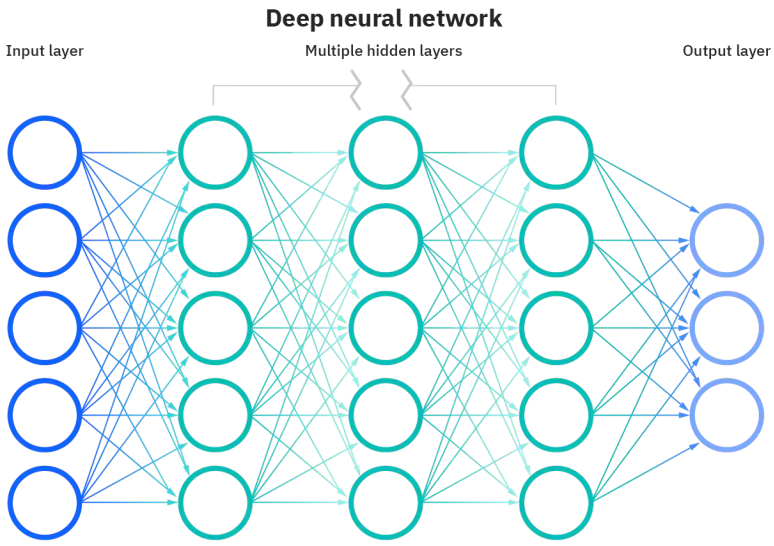

Figure 7. Schematic of the structure of a neural network model [5].

For both NN models, there were four layers, consisting of an input layer, two hidden layers, and an output layer. Each of the first three layers consisted of 64 neurons.

Each of the neurons in the hidden and output layers receive some weighted sum of inputs, and each of the neurons process the input utilizing an activation function. This output is then passed on to the next layer [5]. In the first layer, the weights and biases of the functions are initialized randomly. Then, input data is introduced via this first input layer to the model. This information is forward propagated, with each line in Figure 7 also representing a weight applied to the output of the neuron. At each neuron, an activation function is applied to the weighted sum of inputs. In both the NYC and Boston NN models, a Rectified Linear Unit (ReLU) activation function was chosen for all but the output layer to promote quicker and more reliable convergence and mitigate the issue of the vanishing gradient. The output layer utilizes a linear activation function,

as normalized precipitation values can be negative. The prediction loss is calculated and back-propagated utilizing the concept of chain rule so that the partial derivatives of the loss function can be found with respect to the weights and biases present in the model. The model then updates these values by taking a step opposite that of the gradient, modulated by a proportion called the learning rate. The learning rate for both models was 0.00001.

The model summaries for both the NYC and Boston NN models can be found below in Figure 8 and Figure 9.

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_32 (Dense)            (None, 64)                320

 dense_33 (Dense)            (None, 64)                4160

 dense_34 (Dense)            (None, 64)                4160

 dense_35 (Dense)            (None, 1)                 65

=================================================================
Total params: 8,705
Trainable params: 8,705
Non-trainable params: 0
_________________________________________________________________
```

Figure 8. NN model summary for the NYC NN model.

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_36 (Dense)            (None, 64)                320

 dense_37 (Dense)            (None, 64)                4160

 dense_38 (Dense)            (None, 64)                4160

 dense_39 (Dense)            (None, 1)                 65

=================================================================
Total params: 8,705
Trainable params: 8,705
Non-trainable params: 0
_________________________________________________________________
```

Figure 9. NN model summary for the Boston NN model.

Both models utilize the Adam (adaptive moment) Optimization Algorithm, as Adam combines the benefits of AdaGrad and RMSProp algorithms and is able to achieve good results quickly. Both models ran for 100 epochs, with a minibatch size of 32, and monitored by an EarlyStopping function to monitor validation loss and avoid overfitting. If the validation loss did not improve in 15 steps for the NYC model or 25 steps for the Boston model, the model would end training

prior to the 100 epochs were complete. Additionally, the training set from above was split so that one-third of the training set was portioned out to function as the validation set in training the model.
The purpose of the NN model for NYC was for comparison against the XGBoost model for NYC described later in this paper. The purpose of the NN model for Boston was for comparison against the Transfer Learning model for Boston described in the next section.

Transfer Learning – Boston

Transfer Learning in this application mirrors that of the above explained neural network model; however, Transfer Learning allows for the leveraging of a pre-trained model on a larger dataset to make predictions on a smaller dataset. This pre-trained model allows for a more robust prediction, assuming that there are similar underlying characteristics between the two datasets. In this case, it was assumed that there were similar underlying characteristics resulting in precipitation from the two datasets that could be captured in a model that was pre-trained on the larger NYC dataset. Fine tuning of this pre-trained NN model was performed my unfreezing the top two layers of the pre-trained dataset to train on the Boston-specific dataset, allowing for adaptation of the model to the Boston dataset.

Similar to the NN models described above, the Transfer Learning model ran for 100 epochs, with a minibatch size of 32, and monitored by an EarlyStopping function to monitor validation loss and avoid overfitting. If the validation loss did not improve in 25 steps, the model would end training prior to the 100 epochs were complete. Additionally, the training set from above was split so that one-third of the training set was portioned out to function as the validation set in training the model. The model also utilizes the Adam Optimization Algorithm.

A visualization of Transfer Learning can be found in Figure 10.

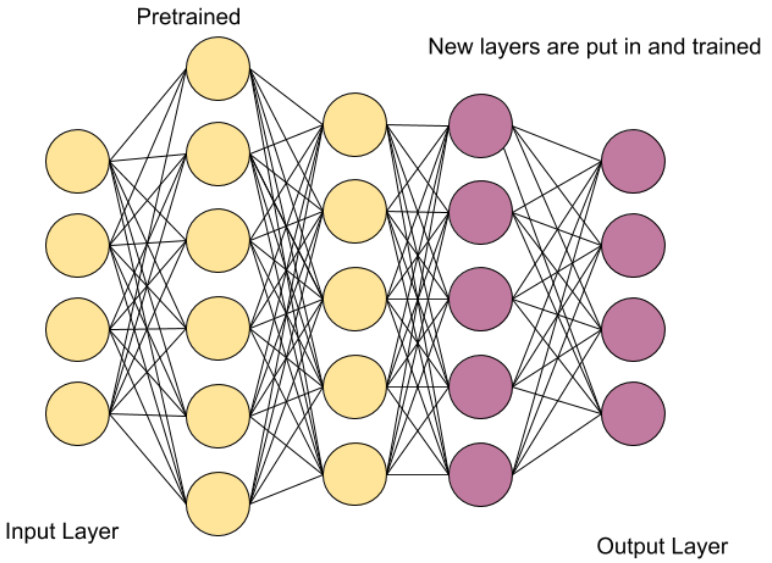

Figure 10. Transfer Learning model sketch [6].

Extreme Gradient Boosting – NYC

XGBoost is a gradient boosted decision tree ML library. XGBoost leverages the concept of the gradient boosting algorithm, wherein each following decision tree is able to correct and optimize the tree that came before it. This training occurs in an iterative manner, in which new trees are added that predict the errors of the prior trees; the newly added regression trees are then combined with previous trees in an ensemble for the final precipitation prediction using a weighted average that is dependent on the aforementioned error [7]. XGBoost also typically offers a more efficient execution speed as compared to other boosting algorithms, and it also is more effective in both classification and regression predictive modeling problems [7].

In this implementation, the XGBRegressor function was utilized.

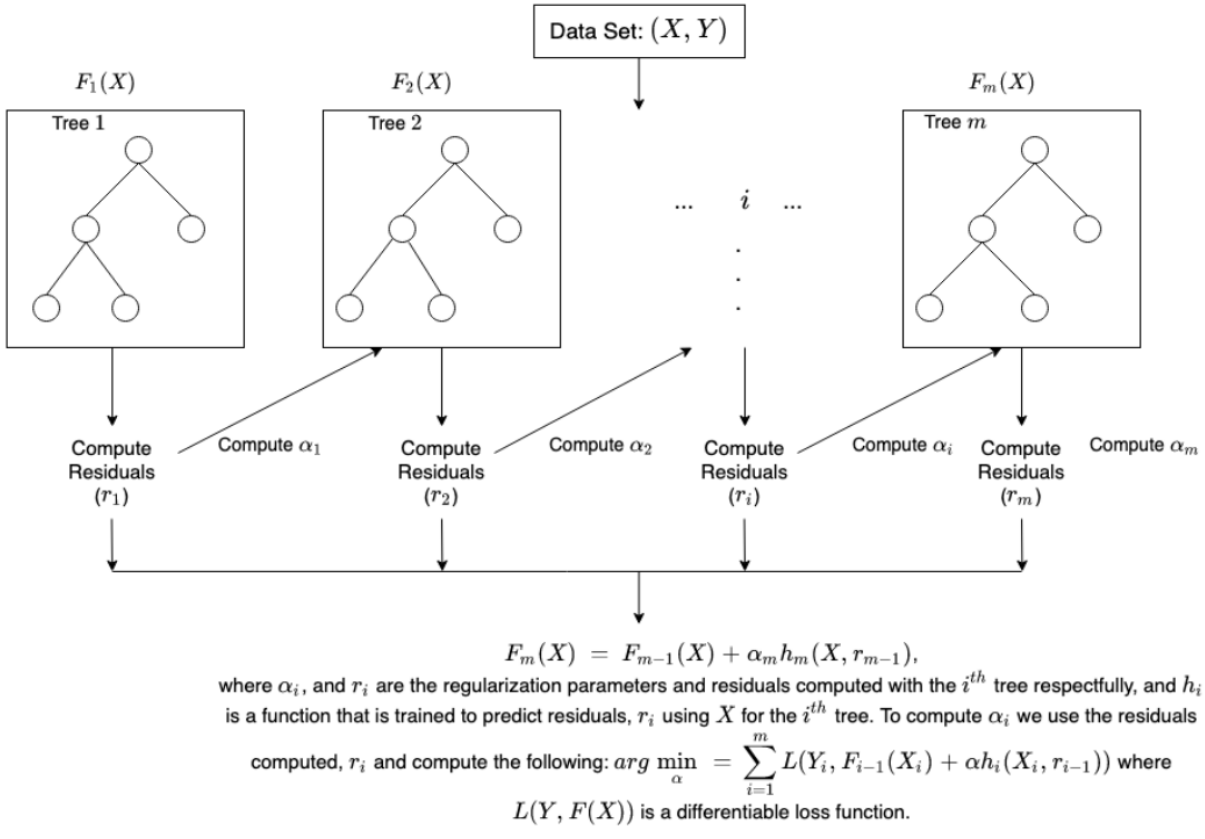

Figure 11. XGBoost model sketch [7].

The following parameters were specified: learning objective (objective), how many trees are built into the ensemble model (n_estimators), learning rate (learning_rate), and maximum depth of a tree (max_depth). The values for the aforementioned hyperparameters after many rounds of trial-and-error style tuning were: regression with squared error, 200, 0.006, and 1, respectively.

Tuning Hyperparameters

All hyperparameters were tuned manually, balancing over- and underfitting of the different models. Hyperparameters were originally set at values used in class examples. Then, hyperparameters were tuned by changing one hyperparameter while keeping the other hyperparameters constant. For example, in the beginning, training loss was not decreasing significantly, so the learning rate was increased to make greater jumps. After a certain point, the learning rate became so large that smaller achievable losses were not being reached. At which point, the learning rate was then decreased. Steps like the aforementioned manual tuning of hyperparameters were completed for all hyperparameters.

Results

Evaluation Metrics

All models were evaluated using RMSE. The equation for RMSE can be found in Equation 1 below. RMSE values were calculated on the predictions using normalized values, so that RMSE values can be thought of as standard deviations from the real test value.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}}$$

Equation 1. Calculation for RMSE.

Neural Network – NYC and Boston

For the NN using NYC data, the figure for the real and predicted values for precipitation using the testing dataset on the trained model can be found in Figure 12 below, where the blue dots signify data from the testing dataset, and the red dots represent predictions from the trained model. It can be seen that the predictions are localized to the regions of greatest density, and outliers are almost wholly not represented in the prediction.

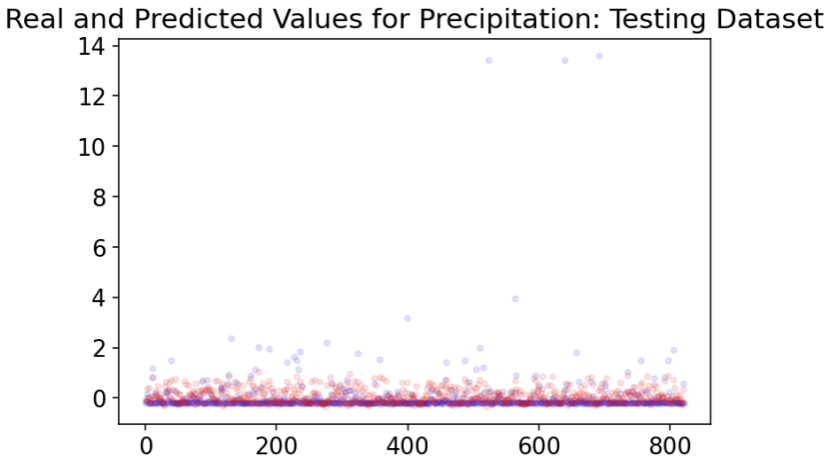

Figure 12. Real and predicted values for precipitation from the testing dataset using the NN trained on the NYC dataset.

The losses between the predicted and real values of the NYC data are depicted below in Figure 13, and the squared losses for the NYC data can be seen in Figure 14. The overall RMSE for the NYC NN model was calculated to be 0.90.

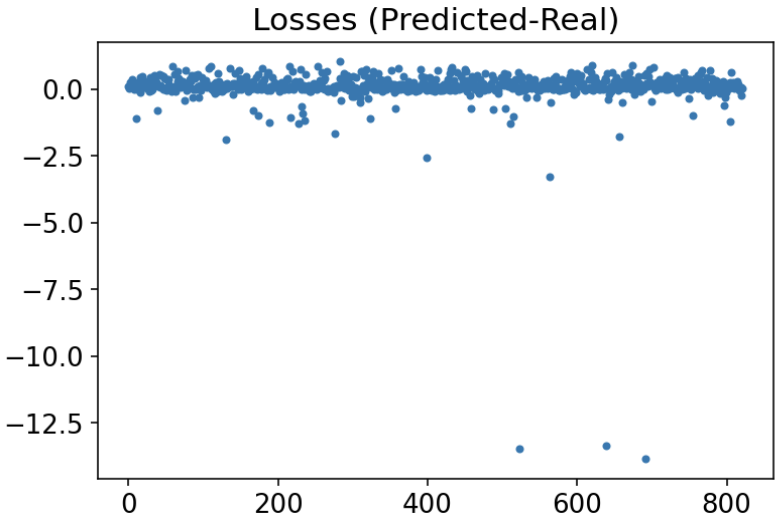

Figure 13. Losses between the predicted and real values of the NYC dataset.

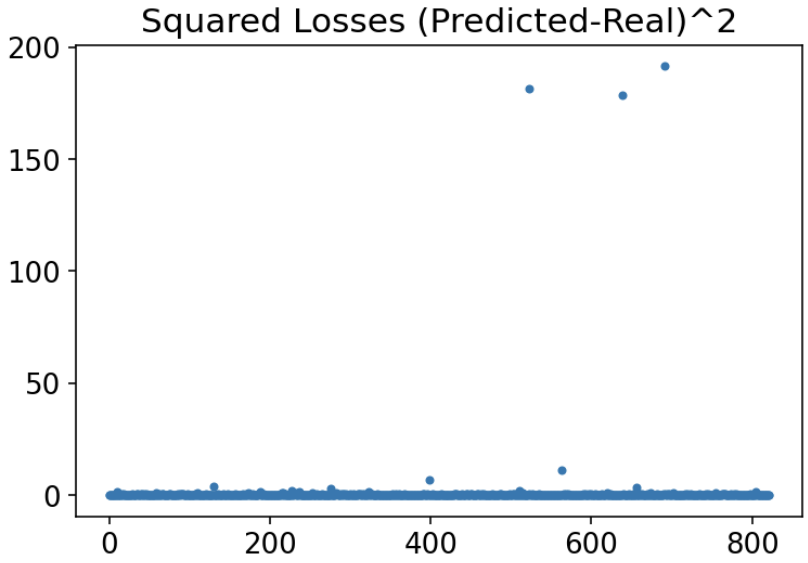

Figure 14. Losses between the predicted and real values of the NYC dataset squared.

For the NN model trained on the Boston dataset, the figure for the real and predicted values for precipitation using the testing dataset on the trained model can be found in Figure 15 below, where the blue dots signify data from the testing dataset, and the red dots represent predictions from the trained model. In this distribution, there was a greater spread of datapoints in the training data as can be seen in Figure 6, or rather, the training data was not as saturated with lower precipitation values with a few extreme outliers like that of the NYC dataset, which resulted in the predicted values of the training dataset being more representative of the real values. This is reflected in Figure 15 below.

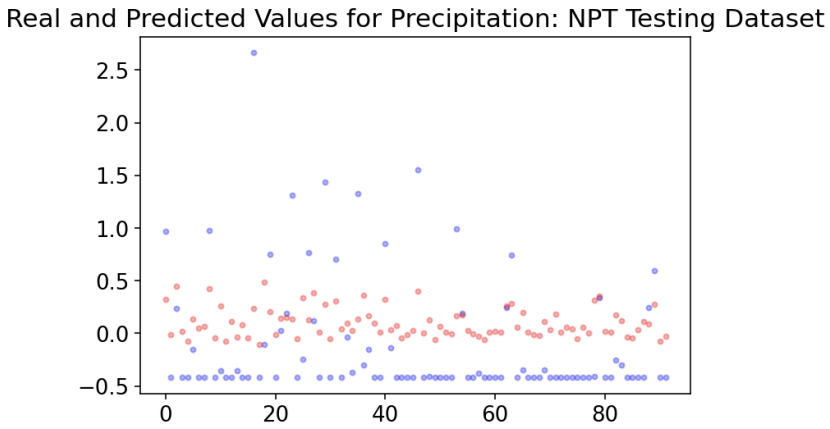

Figure 15. Real and predicted values for precipitation from the testing dataset using the NN trained on the Boston dataset.

The losses between the predicted and real values of the Boston data are depicted below in Figure 16, and the squared losses for the Boston data can be seen in Figure 17. The overall RMSE for the Boston NN model was calculated to be 0.55.

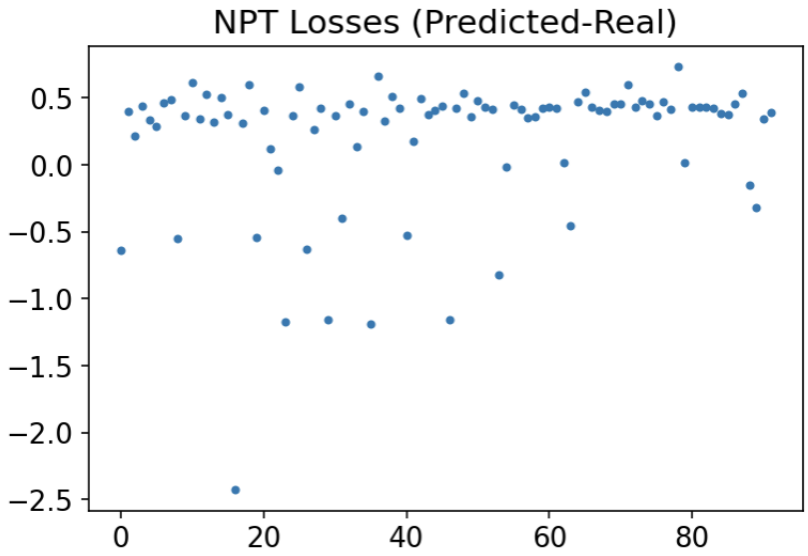

Figure 16. Losses between the predicted and real values of the Boston dataset.

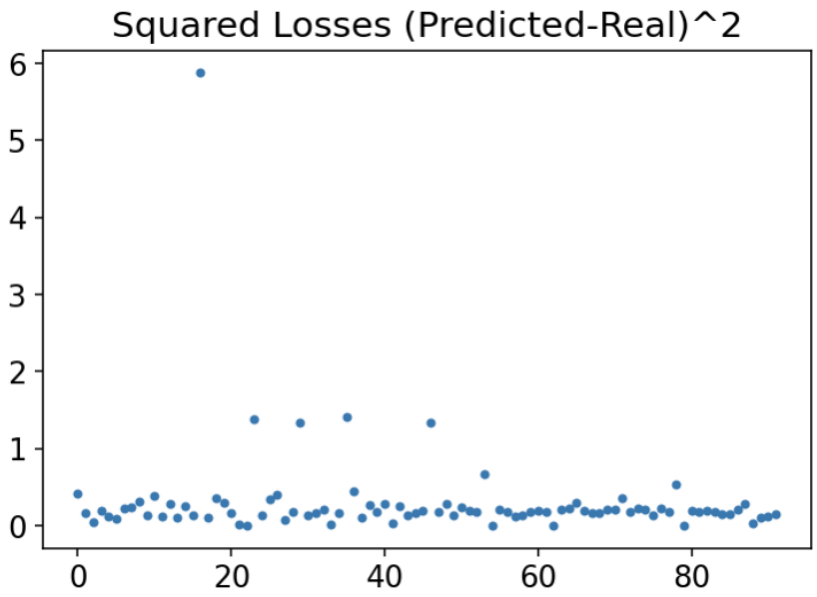

Figure 17. Losses between the predicted and real values of the Boston dataset squared.

Transfer Learning – Boston

For the Transfer Learning model pretrained on the NYC dataset and adapted to the Boston dataset, the figure for the real and predicted values for precipitation using the testing dataset on the trained model can be found in Figure 18 below, where the blue dots signify data from the testing dataset, and the red dots represent predictions from the trained model. Comparing Figure 18 to Figure 15, it is evident that the distribution of predicted values is more similar to the distribution of real values when using the Transfer Learning model.

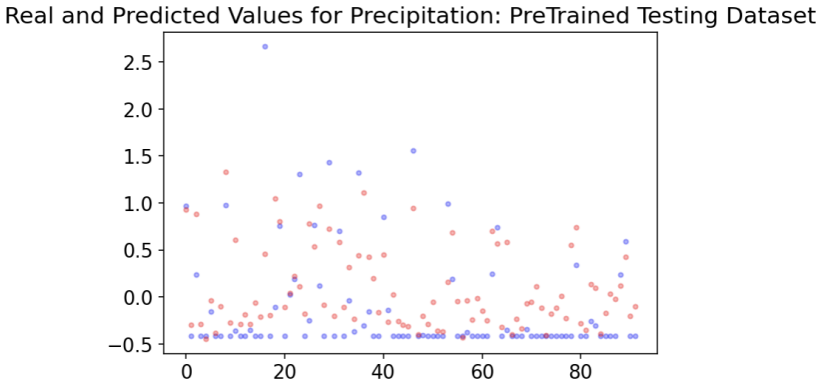

Figure 18. Real and predicted values for precipitation from the testing dataset using the Transfer Learning model pretrained on the NYC dataset and tuned to the Boston dataset.

The losses between the predicted and real values of the Boston data are depicted below in Figure 19, and the squared losses for the Boston data can be seen in Figure 20. The overall RMSE for the Transfer Learning model was calculated to be 0.50.

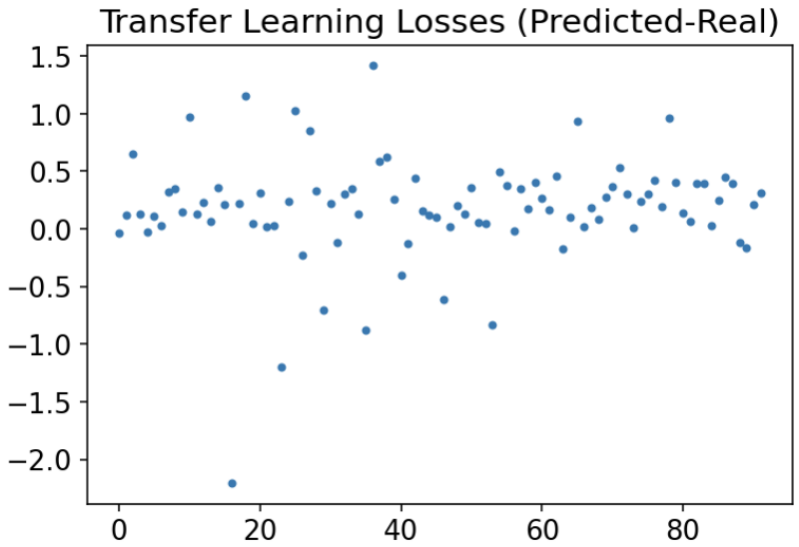

Figure 19. Losses between the predicted and real values of the Boston dataset.

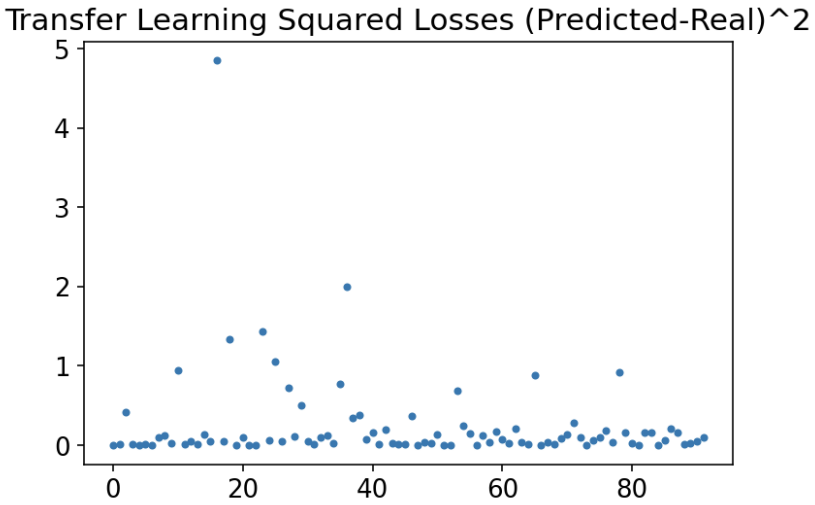

Figure 20. Losses between the predicted and real values of the Boston dataset squared.

Extreme Gradient Boosting – NYC

For the XGBoost model using the NYC dataset, the figure for the real and predicted values for precipitation using the testing dataset on the trained model can be found in Figure 21 below, where the blue dots signify data from the testing dataset, and the red dots represent predictions from the trained model. Compared to the data points in Figure 12, it appears that the data points

in Figure 21 are more affected by the outliers, as there seems to be an upward shift of red, predicted values in Figure 21 than in Figure 12.

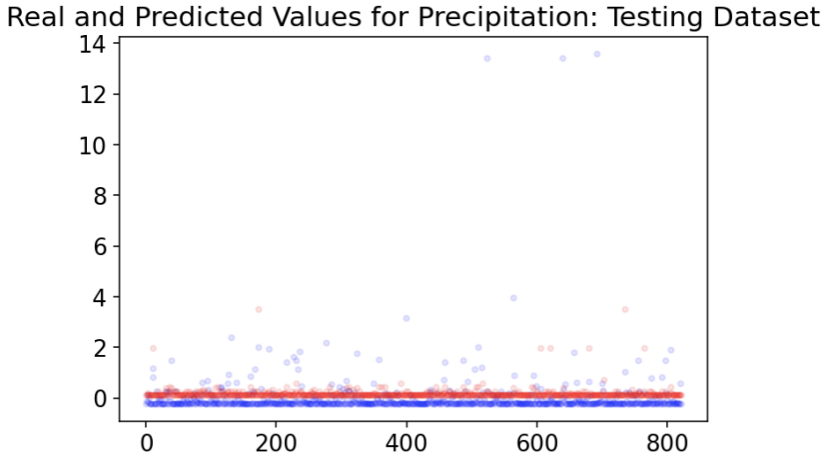

Figure 21. Real and predicted values for precipitation from the testing dataset using the XGBoost model on the NYC dataset.

The losses between the predicted and real values of the NYC data are depicted below in Figure 22, and the squared losses for the NYC data can be seen in Figure 23. The overall RMSE for the XGBoost model was calculated to be 0.92.

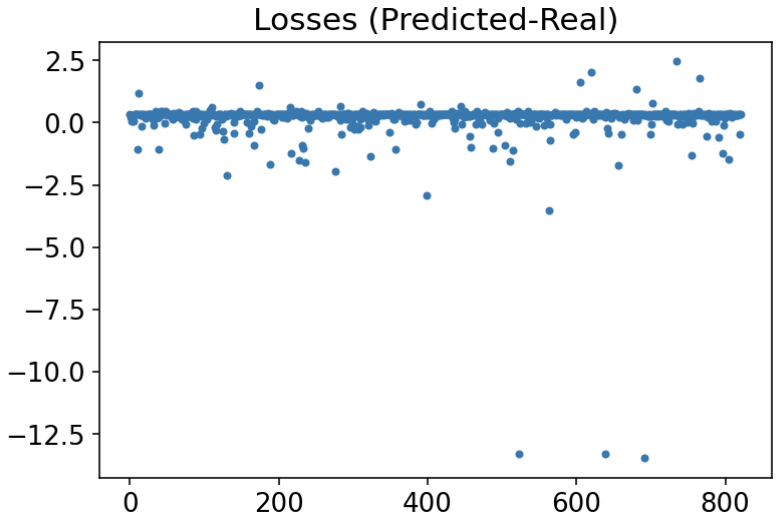

Figure 22. Losses between the predicted and real values of the NYC dataset.

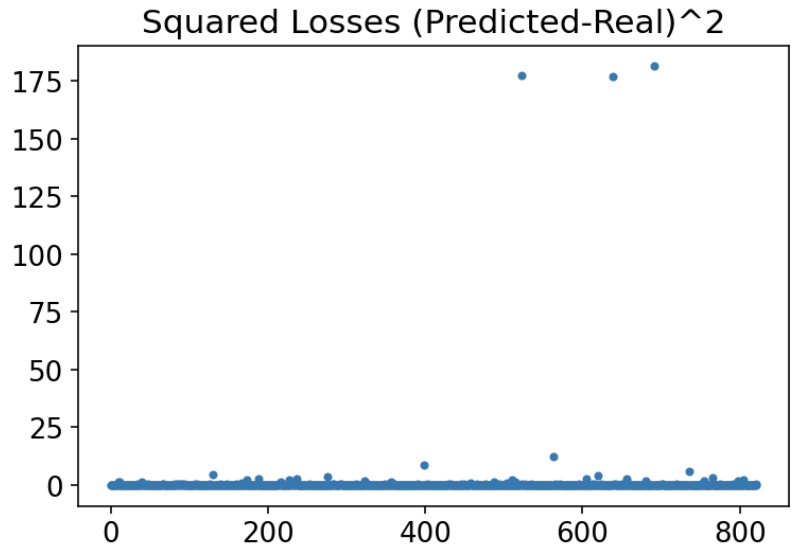

Figure 23. Losses between the predicted and real values of the NYC dataset squared.

Looking at feature importance, it appears that the XGBoost model assigns zero importance for the inputs of temperature and cloud cover, and between the remaining predictors of humidity and pressure, places a greater importance on humidity. The feature importance F score chart can be found below as Figure 24, where f1 represents humidity and f3 represents sea level pressure.

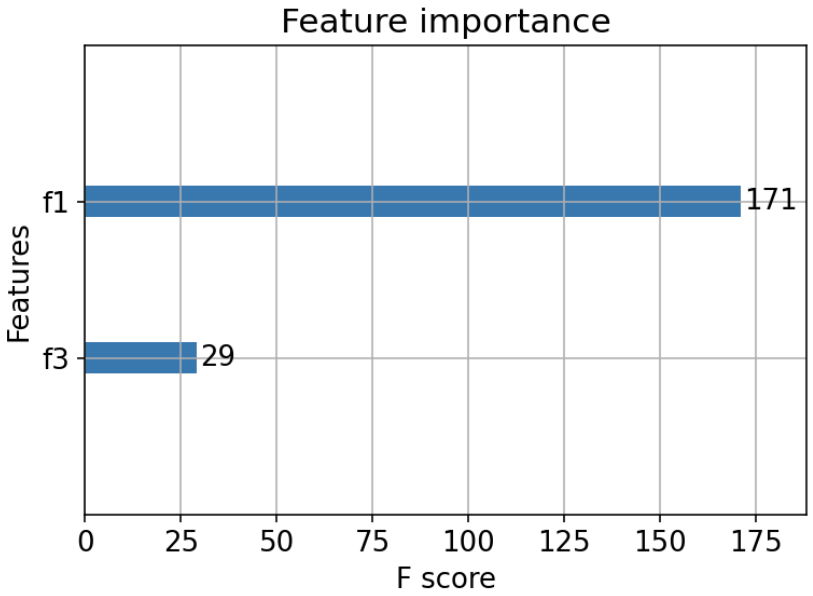

Figure 24. Feature importance breakdown chart comparing F scores of the different parameters for the XGBoost model using the NYC dataset.

Discussion

For the NYC data, it appears that the NN and XGBoost model perform about the same, with RMSE of 0.90 and 0.92, respectively. It is possible that the slight 0.02 difference is due to random chance; however, given that the 0.90 and 0.92 represent approximately a standard deviation in error, this relatively high error is representative of a poorly representative model. This is perhaps due to the fact that a large portion of this error comes from the presence of outliers, of which there is not a great proportion in the dataset. As a result, the model may be overfitted to data that dominates the dataset and unable to fully take into account minute differences that may yield in these outliers. It is also possible that the predictors for these outliers in precipitation are not represented in the dataset to begin with, as precipitation may be correlated to many other variables beyond the predictors of temperature, humidity, cloud cover, and sea level pressure.

For the Boston data, it is evident that the Transfer Learning model is able to predict precipitation with a lower RMSE than the not-pre-trained NN model, with a RMSE of 0.50 and 0.55, respectively. This is likely due to the fact that the Transfer Learning model implicitly has access to a larger dataset, leveraging the hidden layers trained on the larger NYC dataset. It is possible that the models trained on the Boston data has a lower RMSE than the models trained on the NYC data because the Boston data has less extreme outliers. However, given that the 0.50 and 0.55 are still representing 0.50 and 0.55 standard deviations of error, it is possible that these non-negligible RMSE values are a result of insufficient data.

Regarding the feature importance chart, it is possible that temperature and cloud cover do not play a large role in contributing to the precipitation prediction as New York City does not have a specific rainy season and rains year-round, and New York City also has many cloudy days that do not result in rain, as well as many sunny days that do result in rain. It is possible that there is a large importance placed on humidity not necessarily because humid conditions lend themselves to greater precipitation, rather, it is possible that this is reverse causal, in that precipitation results in a larger value for humidity measured. Regardless, there is a great amount of importance placed on humidity in determining precipitation in the XGBoost model. It is also understandable that pressure also plays a role in precipitation, given that low pressure systems often result in precipitation, and high pressure systems largely do not [8].

Error Analysis and Future Work

As mentioned above, it is possible that a large portion of the RMSE comes as a result of outliers, especially since a large percentage of annual precipitation, especially in the past two decades, have been the result of intense, single-day events [1]. These large deviations from the baseline can also be visualized above (i.e. in Figure 5), wherein there are sparse events, outliers, far away

from the denser region of the baseline. These outliers are not reflected in the predicted values, and therefore produce large errors, which heavily contribute to the overall RMSE.

Future work may look to study these intense, single-day events by isolating for these events. A potential means to do such may be to find the mean of the data and remove any data points that are less than one or two standard deviations from the mean, then source more data across a greater number of years to predict intense, single-day events for the future.

For future work looking to better predict the day-to-day weather trends, the same data processing can be completed, except the outliers would be omitted from the analysis. These models would seek to predict day-to-day "normal" precipitation levels, and occlude the intense, single-day events.

Overall, a greater amount of data could prove to train more robust models. Future work could work towards training models on a larger dataset to ensure that the data in all training, validation, and testing groups are representative. Additionally, greater hyperparameter tuning, perhaps not done manually but completed algorithmically, could also improve the accuracy of the models.

Conclusion

In this project, two NN models, one Transfer Learning model, and one XGBoost model were built and trained using weather data from NYC and Boston to predict rainfall in NYC and Boston using predictors of temperature, humidity, sea level pressure, and cloud cover. Hyperparameters were tuned by hand using a trial-and-error method to try to improve the accuracy of the models. For the NYC data, the NN and XGBoost models produced predictions with similar RMSE, and for the Boston data, the Transfer Learning model produced slightly more accurate precipitation predictions. In the XGBoost model, it appeared that humidity and sea level pressure were given the greatest amount of importance. Overall, the models could be improved by better handling outliers (in an ethical manner), increasing the amount of data available, and tuning hyperparameters algorithmically, and future research may also look into predicting the occurrence of intense, single-day precipitation events instead of overall day-to-day precipitation.

Acknowledgements

I would like to thank Professor Pierre Gentine and Teaching Assistant Weiwei Zhan for the kind help and support throughout the completion of this project and course. I immensely appreciate the guidance and grace given to me throughout this semester, and as always, I am grateful for their patience and understanding.

References

[1] "Climate Change Indicators: Heavy Precipitation." *United States Environmental Protection Agency,* 1 Aug. 2022, https://www.epa.gov/climate-indicators/climate-change-indicators-heavy-precipitation.

[2] Tabari, H. Climate change impact on flood and extreme precipitation increases with water availability. Sci Rep 10, 13768 (2020). https://doi.org/10.1038/s41598-020-70816-2.

[3] Federica Zennaro, Elisa Furlan, Christian Simeoni, Silvia Torresan, Sinem Aslan, Andrea Critto, Antonio Marcomini, Exploring machine learning potential for climate change risk assessment, Earth-Science Reviews, Volume 220, 2021, 103752, ISSN 0012-8252, https://doi.org/10.1016/j.earscirev.2021.103752.

[4] "Weather Query Builder." *VisualCrossing,* 12 Jan. 2022, https://www.visualcrossing.com/weather/weather-data-services.

[5] "What are neural networks?" *IBM*, 2023, https://www.ibm.com/topics/neural-networks.

[6] Johansen, Dan. "Transfer Learning from a business perspective." *Dan Rose AI*, 30 July 2022, https://www.danrose.ai/blog/transfer-learning-from-a-business-perspective.

[7] "How XGBoost Works." *AWS,* 2023, https://docs.aws.amazon.com/sagemaker/latest/dg/xgboost-HowItWorks.html.

[8] "The Highs and Lows of Air Pressure." *Center for Science Education*, 2023, https://scied.ucar.edu/learning-zone/how-weather-works/highs-and-lows-air-pressure.