Title: Contrasting Weather Predictions using ANN, RNN, and Gradient Boosting

Catherine Chu, Paul Chirkov, Divik, Verma, Henry Moor

Summary of Project:

Our project uses 3 neural networks: Artificial Neural Networks (ANN) and Recurrent Neural Networks (RNN) to capture the relationships in Weather data. We use a dataset found from Kaggle, listed in more detail in the sources section that details time series data for Weather. Since the data is time series dependent, we used RNNs to accurately reflect the temporal dependencies and patterns in weather over time. Additionally, we also wanted to contrast the use of ANNs, which can capture relationships in tabular data by learning from the input features to predict a target variable (we chose max_temperature). Finally, after multiple attempts to train the ANN, we tried using Gradient boosting, closely following the example file given in Math76: Mathematics and AI in the file problem set.

Abstract: We use RNNs to predict three critical features for each city in the dataset, based on the values of all the other features. We wanted to, given cloud cover, humidity, pressure, etc., predict the maximum, minimum, and mean temperature on a given day. RNNs are typically the preferred method to capture time series-dependent data. We also use ANNs just to compare the accuracy differences.

We then train an ANN and gradient boosting model on the data from Heathrow from 2000 to 2010. There were a total of 3654 daily observations from 18 places. We chose to train the data on Heathrow because Heathrow has similar fluctuation and temperature levels that correspond and can be extrapolated to places such as Stockholm, Oslo, and Kassel. Figure 1 demonstrates how the maximum temperatures of these 4 locations are all generalized. Since I had just visited London Heathrow, I chose it although all the predictions should be well able to generalize to the additional locations.

Dataset:
https://www.kaggle.com/datasets/thedevastator/weather-prediction/data?select=weather_prediction_bbq_labels.csv

We found this dataset on Kaggle and chose it for 2 main reasons. The first reason is because it included time series data from the longest time frame (2000 to 2010) with consistent data. It had aggregate data from 18 different locations which we found to be comprehensive. The second reason was because it included the most features. For each of the 18 locations, there are a variety of features that it examines such as cloud cover, humidity, pressure, global radiation, precipitation, sunshine, temperature mean, temperature min, and temperature max. Naturally, we select the features that we find most important to model or predict, but nonetheless, many features give us a larger dataset to choose from.

The original dataset is meant to model the BBQ rates/frequency in different locations throughout different points in the year.
Our project has a very different focus, however, we try to recreate a chart to produce the BBQ rates at the beginning of the Weather.ipynb file to ensure reproducibility.

About the Weather.ipynb file

The code starts by loading in the dataset and reproducing the results from the dataset about the barbeque in Heathrow instead of Dresden to make sure the data is cleaned correctly.
Then, the code starts training an ANN to predict the maximum temperature. It splits the data into features except the last column, which is the maximum temperature. The maximum temperature is the target column. We load the data into pytorch because tensorflow wasn't compatible with mac. Then we defined the ANN first with only 3 layers. We trained for 100 epochs, 10 at a time and analyzed the loss.
In the next section, we tune the hyperparameters to improve accuracy. We try to tune the learning rate but don't get really strong correlation results. We then use 3 methods of validation visualization that we carry on throughout the code. First, we plot the training loss against the test loss. Then, we plot the predicted versus the actual values. Finally, we plot the residuals. We repeat this process with a tuned ANN model that has more layers and drop nodes to prevent overfitting. We plot the same 3 graphs again but don't see much improvement.
Finally, we try the gradient boosting algorithm, which has some risk of overfitting because the training loss and test loss don't have a huge correlation. However, the predicted versus actual values are very promising. We conclude that it does better than the ANN.

About the gru_lstm file

The GRU_LSTM file trains three models. The first of these was a vanilla RNN with no GRU or LSTM as something of a baseline. This model performed remarkably well to begin with, and with more epochs and some slight feature engineering to prevent overfitting, it is likely that this model could become more accurate further still.
The second model was adapted with GRUs to enhance the accuracy. This had the intended effect, causing the RMSE to decrease for virtually every category of prediction.
The last model we trained was a far more complicated model consisting of an RNN equipped with LSTM and GRU layers.