# MSDS Capstone Project Update #3

Many silly mistakes made by me this week that resulted into a poor performing model. First up was that I didn’t restrict the tornado dataset to include only tornadoes within the same bounding box as I used for the NARR. The second was that I was not adding in the dates with zero tornadoes. The SPC dataset only has days with tornadoes. So, my model was training on days with 1 or more tornadoes. This resulted in a training accuracy of 62%.

In order to fix this first mistake, I had to find a way to extract only the tornadoes within my box. I used the GeoPandas library in Python. With this I first put the tornado dataset into a Pandas data frame then convert it into a Geo-Data frame by telling it that the geometry is the starting longitude and starting latitude columns in the data frame. To make sure I would indeed set it up correctly, I first plotted each point onto a map. Due to the size of the map, the Continental US was all red. I didn’t care about map size as I was just using this as a quick check. I then created another data frame of the corners of my box and converted this into a GeoPandas data frame as well. Plotted this as well just to confirm the points looked good, and to remind myself of the box. I then used the Shapely library to create a polygon using the corners. I was then able to create a new GeoPandas data frame by using geometry.within. When I plotted just this data frame, the only red I saw was where the box would be.

By using groupby function in Pandas, I was able to get a series data type of dates and the number of times it occurred in the data set. Then created a Pandas date range from 1/1/19 to 12/31/13. I then did a re-index of the series with a fill value of 0. Finally, was able to save this as a CSV to have every date having a count.

With this tornado count dataset being right, I retrained my model. However, even though I had done the training/testing data split, I was training using the full dataset and then testing on the testing dataset. This is not what we want to do! So once again a silly mistake made by me. Honestly, it these may have been made due to me feeling a little rushed. However, once I fixed this last mistake, I got a training accuracy of 87.5% and a testing accuracy of 82.2%. So, a much better result. There is a misclassification of 17.8%, which for my dataset, I don’t think is too bad. However, the recall is 0.62%. My theory is there are many days with no tornadoes and less than 5 tornadoes than there are with more than 5 tornadoes. In fact, in the testing dataset there are only 486 days with 5 or more tornadoes and 3,350 days with 5 or less tornadoes. Given this, the model would be more likely to predict no than yes. I expected this to happen given the nature of meteorology. I don’t have this number right now, but I am going to tally the number of yes and no in the full data set. I expect to see a similar result. This said, I still think the results look pretty impressive for meteorology.

I am currently tuning the model. There is a Grid Search function within the scikit-learn library that allows you to run multiple models using different parameter values and find the best result. I have run the model using 20, 40, 60, 80 and 100 for both batch size and number of epochs. The best result, using the training data set, was 87.8% with batch size of 40 and 100 epochs. I then ran the model using this result and got a training accuracy of 87.5% (slight variation was expected). The testing data set with this had an accuracy of 81.2%, which is slightly lower, but not too bad from the initial model. The misclassification was 18.8%, which again is slightly higher than before. The recall was 0.82%, which is better than before. So, there was improvement on classifying days with 5 or more tornadoes with these parameters.

The previous two models used the Adam optimizer. I use scikit-learn again to test different optimizers using the 100 epochs and 40 batch size. Here it determined the best optimizer was Adamax with 87.7% training accuracy. Using these parameters, we got 86.7% training accuracy and a testing accuracy of 81.1%, which is lower, but I can tell from the confusion matrix before calculating recall that it will be a vast improvement. There are 92 true positives this model versus 3 and 4 before! Here the recall is 18.9%, which is a huge improvement. So, this model is already starting to predict actual tornado days with better results.

I will also be tuning learning rate and momentum (beta for Adamax), network weight initialization, the initial layer’s activation function, and the initial layer’s number of neurons. The activation function and number of neurons is just to see what would happen, I’m not entirely sure with my network they will be helpful to tune but won’t know until I try!

I also need to run this through the 00Z data. Overall, however, I am pleased with my results and think I have good results that I can be proud of! I’m sure with more time I would be able to tune even more and play with the hidden layers and so on.