Using Machine Learning to Predict Tornado Ratings

1. Introduction

The purpose of this preliminary research is to use machine learning to predict the Enhanced Fujita rating of tornadoes. There was some limited success in the models that were used during this stage of research. Due to the limited success, more research is needed to try to improve the models and understand the results more.

2. Data

The National Centers for Environmental Information (NCDC) from the National Oceanic and Atmospheric Administration (NOAA) has a storm events database from January 1950 to July 2018 (National Oceanic and Atmospheric Administration). The data in this database is collected and entered by the National Weather Service (NWS). The files are split into yearly files and are in the comma separated value file format.

Tornado events have been recorded since 1950, however initially the Fujita scale (F-scale) was used to rate these tornadoes. In the early 2000s the scale was replaced by the Enhanced Fujita scale (EF-scale). The initial plan for this research was to convert all the Fujita rankings into the EF scale ratings. With the limited time to perform this research, this plan was not implemented. Due to this fact, it was determined that for this preliminary research that the data would be restricted from January 2008 through July 2018.

3. Data Issues

There are some issues with the data that the researchers had to be aware of. The first is there is 10 years' worth of data that was usable for this project. This most likely played a role into the results of the models used. Another issue is the limitation of rating tornadoes. The EF scale is based on damage they cause, not the wind speeds, though the wind speed is able to be estimated based on damage. The following example is meant to help the readers understand this point. If you take an EF4 tornado that hits an urban area and move it to a corn field, the rating would be ranked lower, EF1 for instance, because it didn't hit any structures. The last issue is the possibility of typos as this data is entered by a human.

4. Data Cleaning

The data cleaning was an involved process. The first step was to combine all the data files into one and ignore all the fields that were unnecessary for the scope of the research. As there are more than just tornado data in the files, anything that was a weather event not labeled as a tornado was removed. The database had a field that labeled each event as a certain weather event. There were also tornado rankings of EFU (unknown), F0, F1, and F2. Any tornado event with these ratings were removed from our data as well. The damage cost to property and crops had some values that were missing, instead of dropping these rows, they were filled in with a value of 0.0K. The values of these two fields were then converted into integers instead of the 1.0K format. The last step of the data cleaning process was to take the tornado rankings and make them one hot encoded for use in the models. After this step was done, the data was split into training and testing data sets with the test data set being 20% of the total data set. Some models needed the data to be scaled, so this step was also completed.

5. Process and Results

The first machine learning algorithm used was the Naïve Bayes classifier. This model is based on the Bayesian theorem and assumes that the input features are independent and that they contribute and equal amount to the model. This model is a simple model to implement and is useful when there is a large data set. Due to its simplicity, it can outperform more complex models. Naïve Bayes computes the posterior probability of some class given a predictor, or input values. The formula for this is $P(c|x) = \frac{P(x|c)P(c)}{P(x)} = \frac{P(x_1|c)\times P(x_2|c)\times \ldots \times P(x_n|c)\times P(c)}{P(x)}$. This formula is computed for each class of the data set for each data point and the class with the highest posterior probability is the prediction of that data point. There are three types of Naïve Bayes model in the Python library scikit learn that was used and Gaussian method was used. The Gaussian method assumes that the data follows a normal distribution. The scaled version of the data set was used in this method.

Classifcation Report

	precision	recall	f1-score	support
0	0.61	0.98	0.76	1661
1	0.46	0.14	0.21	1048
2	0.47	0.18	0.26	312
3	0.23	0.04	0.06	85
4	0.50	0.03	0.06	29
5	0.05	1.00	0.10	2
avg / total	0.54	0.59	0.50	3137

Figure 1 – Naïve Bayes Classifier Classification Report

Figure 1 shows the classification report of the Naïve Bayes model. Precision is the ratio of the correctly predicted positive (correct class) to the total predicted positives (correct and incorrect class). This tells us of all the data points classified as this certain EF rating, how many of them were really this

EF rating. This can be written as a formula as $precision = \frac{true\ postives}{true\ postives + false\ postivies}$. The Naïve Bayes overall had a precision of 0.54 meaning we had 54% of EF scale predictions correct. This figure also shows that EF0s had the best precision with a value of 0.61. This tells us that the model predicted 61% of the ratings it said were EF0s correctly.

Recall is the ratio of the correctly predicted positive (correct class) to all the actual class. In our case we are saying of all the EF scales that are correct, how many did we label. This can be written as a formula of $recall = \frac{true\ positives}{true\ positives + false\ negatives}$. Here we have 59% correct EF scales that the model labeled as such. The individual rating with the best recall is EF5 with a value of 1.00. This means that out of all the correct EF5s we predicted all as EF5.

F1 score is the weighted average of precision and recall. This means it takes both false positives and false negatives into account. This score is usually a better measure than accuracy. Here we have a value of 0.50, so this tells us we classify 50% of the data correctly. Overall, this model doesn't tell anything other than it can guess EF scales right about half of the time.

The second model that was used was Logistic Regression. Logistic regression is related to linear regression, the biggest difference is linear regression is to forecast values while logistic regression does classification tasks. This model uses a linear equation with independent predictors to predict a value. This value can be from negative infinity to positive infinity. For classification, we only want values of 0 to 1. The way we get the values to become 0 and 1 is to use a sigmoid function. The linear equation is $z = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_x + \cdots\ \theta_n \cdot x_n$ and the sigmoid function is $g(x) = \frac{1}{1 + e^{-x}}$. The variable θ are the weights of each input feature. The logistic regression combines these, and we get the formula of

$h = g(z) = \frac{1}{1+e^{-z}}$. What happens is the model takes the output of z that is computed from the linear equation and puts it into the sigmoid which gives us values between 0 and 1.

Logistic regression uses a cost function and specifically it uses a logarithmic loss function to calculate the cost of misclassifying. Gradients are used to update the weights of each input feature to get a more accurate result. The process of logistic regression is to initialize the weights to 0, calculate the linear equation, use the linear equation output as input into the sigmoid function and then calculate the cost. Once the cost is calculated, it is used to calculate the gradients and multiply the gradients by a learning rate. This continues until convergence at a minima. This minima tells the model that the weights are at appropriate values to get the best results. When there are more than two classes, logistic regression uses a one versus all approach. This method uses the exact same method as a two-class model, however the difference is that it is run multiple times for the same data. The only difference is what class is being used. For example, it would pick out EF0 and then run logistic regression on that with all the other scales are one class. It will then go to the next class and do that for all the classes.

Classifcation Report

	precision	recall	f1-score	support
0	0.69	0.94	0.80	1661
1	0.51	0.39	0.44	1048
2	0.25	0.05	0.08	312
3	0.25	0.05	0.08	85
4	0.50	0.03	0.06	29
5	1.00	0.50	0.67	2
avg / total	0.57	0.64	0.58	3137

Figure 2 – Logistic Regression Classification Report

Figure 2 is the classification report from the model. All the average values have increased showing us that this model is preforming better. In this model we notice that the precision for EF5s goes to 1.00 meaning we have no false positives for this class. However, the recall goes

down meaning we have false negatives this time. Overall, this model predicts tornado ratings much better than Naïve Bayes.

A third model of Support Vector Machines was used. The way this model operates is by plotting each data point as a point in n-dimensional space. The n-dimensions is the n number of features. In the case of this research that would be an 8-dimensional space. The value of each feature is the coordinate value. SVMs then perform classification by finding the hyperplane that differentiates the classes the best. SVMs also maximize the margin. This means the SVM wants to choose the hyperplane that has the biggest margin between each class. It can be thought of we want the classes to be far away from the hyperplane. SVMs also use a kernel trick where it takes kernel functions to do complex data transformation to make the data separable and then find the hyperplane. Three kernels that can be use is the linear kernel, the polynomial kernel and the radial-basis function (RBF) kernel.

All three kernels were tested with this data set, polynomial did not seem to want to finish running, so it was removed from consideration. That left the linear and RBF kernels. After running both, the results showed the RBF kernel was the better result.

Classifcation Report

	precision	recall	f1-score	support
0	0.75	0.87	0.81	1661
1	0.56	0.55	0.55	1048
2	0.50	0.21	0.29	312
3	0.46	0.28	0.35	85
4	1.00	0.03	0.07	29
5	0.00	0.00	0.00	2
avg / total	0.66	0.67	0.65	3137

Figure 3 – Support Vector Machine Classification Report

The SVM model once again improve the results from the previous two models. There is one downfall of this model, that is this model did not predict any EF5 tornadoes when it should have classified two. This can be an issue with using this model over the other two.

The final model used was a Decision Tree. A decision tree model is as it sounds, a tree of decisions. For instance, there can be a question of is tornado width less than half a mile. If the data you are trying to classify has a width that is less than half a mile you go down the yes branch, otherwise you go down the no branch. This continues until a leaf node is hit. A leaf node is the final node of a branch. In order to grow a tree, there are steps that decide what features to include in the tree and what conditions to use for the selected features. The model chooses splits based on which feature will explain the most data.

Lastly, the model needs to know when to stop, which is called pruning and there are two pruning methods. The first pruning method is the most accurate but is more time consuming. This is the post-prune where the model goes until it can go no further, and at this point it removes nodes that overfit the data. The other method, which was used in this research, is pre-pruning. Here we tell the model how many nodes to create before stopping the model. This is the fastest method, but not the most accurate. Pre-pruning takes trial and error to find the best result. In both cases tested, with and without EF4 and EF5, 4 nodes below the root was the most accurate.

Classifcation Report

	precision	recall	f1-score	support
0	0.83	0.77	0.80	1661
1	0.62	0.52	0.57	1048
2	0.53	0.05	0.09	312
3	0.50	0.11	0.17	85
4	0.00	0.00	0.00	29
5	0.00	0.00	0.00	2
avg / total	0.71	0.59	0.63	3137

Figure 4 – Decision Tree Classification report

Decision tree model again increased the overall results in terms of precision. It decreased the recall and F1-score. This could be based on the fact that it did not predict any EF4 or EF5 tornadoes, which is once again a problem.

Based on the results found in all four models and noting that the support size for EF4 was only 29 samples and the support size for EF5 was only 2 samples, all tornadoes ranked as those ratings were dropped from the data frame to see if the results of all the models would improve. In general, the models improve slightly or stayed approximately the same. The Naïve Bayes Classifier decreased from 0.54 precision score to a 0.53. The Logistic Regression model improved from 0.57 to 0.63. The Support Vector Machine model improved from 0.66 to a 0.69. Finally, the Decision tree improve from 0.71 to 0.75. By remove EF4 and EF5 the Logistic Regression model improve the most with Decision Tree improving the second most. In this instance, choosing the Decision Tree model to predict tornadoes of EF0 to EF3 is the best option.

6. Conclusion

Overall, while the precision of the data improved from model to model, and again from remove EF4 and EF5 tornadoes, the other metrics didn't change much. This leads to the belief that machine learning algorithms used are not able to predict all tornadoes the best. Each model had strengths on individual rankings, but not overall. This could deal with the fact that there are

not enough data points. If more time was allowed, tornadoes from 1950 to 2007 may have been able to be converted to EF scale from the F scale.

There is more research to be done to get any concrete evidence on machine learning and predicting tornado ratings. An example is that Neural Networks were attempted but were not able to come up with good results. This could relate to the regularizations being used, the functions being use, the number of hidden layers, and any combination of these options. Ensemble models were used as well, but again were not producing results. Ensemble models were only run with equal weights for each individual model, if attempts were made to weight them differently, different results could have been produced.

The lack of number of data points for EF4 and EF5 also could be a limiting factor. With only having a handful of data points, models cannot get a good understanding of those two ratings. This can possibly be improved by converting F to EF scale again. Removing input features is another option to use. Dimensionality reducing models were not used and should be ran to determine the features that should be used and those that should be ignored.

In conclusion, more research needs to be done to get better results. There are multiple ways of attempting to get these results to improve. This research can not give any evidence as to why one model is better than the other and also allow these models to be used to help NWS figure out future tornado ratings.

References

National Oceanic and Atmospheric Administration. n.d. <https://www.ncdc.noaa.gov/stormevents/ftp.jsp>.