Using Machine Learning Techniques To Improve Rainfall Prediction

Besant Nandra
University of Central Florida
besantnandra@knights.ucf.edu

Fnu Tulha
University of Central Florida
tulha@knights.ucf.edu

Abstract

Predicting rainfall is an important task in todays world. Not only does it have many implications for the general population, but also for meteorologists as it helps them identify and predict severe weather. This paper is acts as an extension of Coblenzs previous work [1], which analyzed atmospheric features to predict rainfall. We applied techniques of dimensionality reduction to a dataset of atmospheric features and assessed how it impacted prediction accuracy. We found that using a smaller subset of features not only improved performance, but also increased accuracy. We discuss these findings at the end of our paper, and talk about potential future work to this project.

1 Introduction

Many traditional methods of predicting rainfall do not incorporate machine learning tactics. Furthermore, most of those methods analyze historical rainfall patterns rather than actual atmospheric data. Coblenz applied machine learning tactics to a set of atmospheric features to assist in predicting rainfall [1]. He achieved high levels of accuracy, however his performance was lackluster. This project focused on applying the machine learning methods to a similar dataset of atmospheric features, but focused on using dimensionality reduction to improve performance.

Dimensionality reduction is a common technique in machine learning to improve the performance of algorithms. While machine learning methods are highly accurate and effective, they lack the efficiency to be applicable in many real world situations. The idea behind dimensionality reduction is to select a smaller subset of features to work with. Ideally, this smaller subset of features should yield similar levels of accuracy to the original dataset, while increasing efficiency drastically.

There are a plethora of atmospheric conditions collected at almost every hour of every day. Analyzing all these features for large areas is incredibly expensive and impractical. While meteorologists like to use some of these features in predicting rainfall, not much consensus about specific features exists.

In this project we took a subset of twelve features, which includes the following: cloud water, Haines index, precipitable water, wind pressure, surface pressure, relative humidity, wind temperature, surface temperature, cloud cover, total ozone, u-component wind velocity, and v-component wind velocity. We performed k-nearest-neighbors (kNN) and support vector machine (SVM) algorithms to analyze these features and predict rainfall. We then used principal component analysis (PCA) on our features and recalculated the accuracies with kNN and SVM.

2 Related Work

As mentioned earlier, the motivation for this project comes from Coblenzs previous work [1]. He analyzed rainfall features using kNN, SVM, nave Bayes classifier, and random forests. He found that random forests yielded the best accuracy, however he did not perform much work in dimensionality reduction.

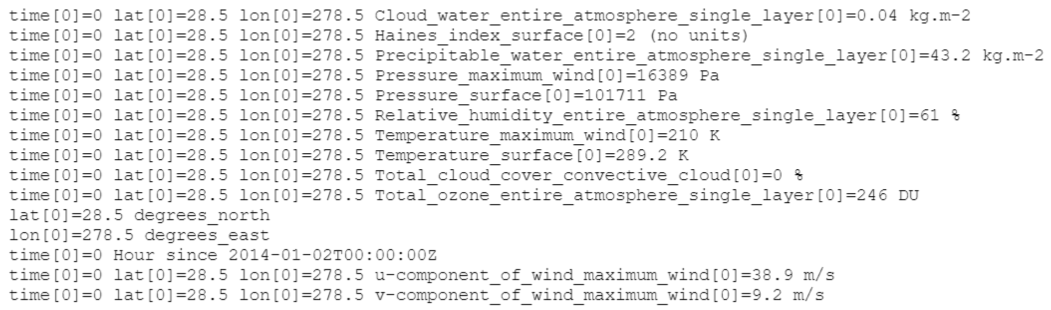
Viana and Sansigolo [2] applied a multiple discriminant analysis (MDA), along with PCA to identify high-variance features in rainfall in Southern Brazil. They analyzed variance among atmospheric features and used this knowledge to produce a better method for forecasting rain.

Du, Liu, Yu, and Yan [3] applied SVM and particle swarm optimization (PSO-SVM) to prediction rainfall and then used PCA for feature selection. However, their main focus was on proving the success of the PSO-SVM algorithm, and not so much on feature selection.

3 Design and Methodology

The first step in approaching this project was collecting data. The Global Forecasting Model (GFS) [4] provides extensive historical data on many different kinds of atmospheric conditions. They have databases setup where this data can be refined in scope, and then downloaded. However, they have specific file formats for storage efficiency purposes, such as GRIB and NetCDF. We worked with NetCDF files since it had easy tools in Linux to work with. We were able to perform a simple batch conversion of all our NetCDF files to ASCII text files using the NetCDF Operator (NCO) [8], which we then used to perform the calculations in this project.

We chose to collect atmospheric data on the following features for the entirety of 2014 in Orlando, Florida: cloud water, Haines index, precipitable water, wind pressure, surface pressure, relative humidity, wind temperature, surface temperature, cloud cover, total ozone, u-component wind velocity, and v-component wind velocity. After conversion, the text files all had the same exact format, as shown in figure 1.



```
time[0]=0 lat[0]=28.5 lon[0]=278.5 Cloud_water_entire_atmosphere_single_layer[0]=0.04 kg.m-2
time[0]=0 lat[0]=28.5 lon[0]=278.5 Haines_index_surface[0]=2 (no units)
time[0]=0 lat[0]=28.5 lon[0]=278.5 Precipitable_water_entire_atmosphere_single_layer[0]=43.2 kg.m-2
time[0]=0 lat[0]=28.5 lon[0]=278.5 Pressure_maximum_wind[0]=16389 Pa
time[0]=0 lat[0]=28.5 lon[0]=278.5 Pressure_surface[0]=101711 Pa
time[0]=0 lat[0]=28.5 lon[0]=278.5 Relative_humidity_entire_atmosphere_single_layer[0]=61 %
time[0]=0 lat[0]=28.5 lon[0]=278.5 Temperature_maximum_wind[0]=210 K
time[0]=0 lat[0]=28.5 lon[0]=278.5 Temperature_surface[0]=289.2 K
time[0]=0 lat[0]=28.5 lon[0]=278.5 Total_cloud_cover_convective_cloud[0]=0 %
time[0]=0 lat[0]=28.5 lon[0]=278.5 Total_ozone_entire_atmosphere_single_layer[0]=246 DU
lat[0]=28.5 degrees_north
lon[0]=278.5 degrees_east
time[0]=0 Hour since 2014-01-02T00:00:00Z
time[0]=0 lat[0]=28.5 lon[0]=278.5 u-component_of_wind_maximum_wind[0]=38.9 m/s
time[0]=0 lat[0]=28.5 lon[0]=278.5 v-component_of_wind_maximum_wind[0]=9.2 m/s
```

Figure 1: Shows the details for how the variables were organized in each text file.

Due to the similar format of all the files, we were able to perform string parsing to extract the important integers out of the file. The data from the GFS database, however, did not include the labels of whether or not it actually rained on a given day. We initially used the Next Generation Weather Radar (NEXRAD) [5] to collect the labels, however their data was given as a map of precipitation thresholds, rather than actual concrete labels. To facilitate data collection, we instead took advantage of weather undergrounds [6] large database of historical precipitation data. They do not include an exhaustive list of atmospheric features like the GFS database, however they do include labels of whether or not it rained. We had to manually collect the data from these two databases, since there was no easy batch download available. We were left with a two-dimensional matrix of features, with the last feature being the label of whether or not it rained.

This two-dimensional matrix was used as the input for both our kNN and SVM algorithms. This input style is ideal for performing these algorithms, and it allowed us to focus on modifying parameters as needed. After performing these algorithms on the full dataset, we then applied PCA to select a subset of features based on variance. We then re-ran our algorithms for all dimension subsets, and compared their accuracies.

4 Results

Evaluating most machine learning classifiers is done using an ROC curve. Thus, we performed our algorithms and generated these curves to evaluate their effectiveness. For both algorithms, we applied a ten-fold cross validation to our dataset.

We first performed kNN on our dataset. We ran kNN using Euclidean distances for a k varying between one and 200, leaving us with the graph shown in figure 2 that told us a k of seven was optimal.

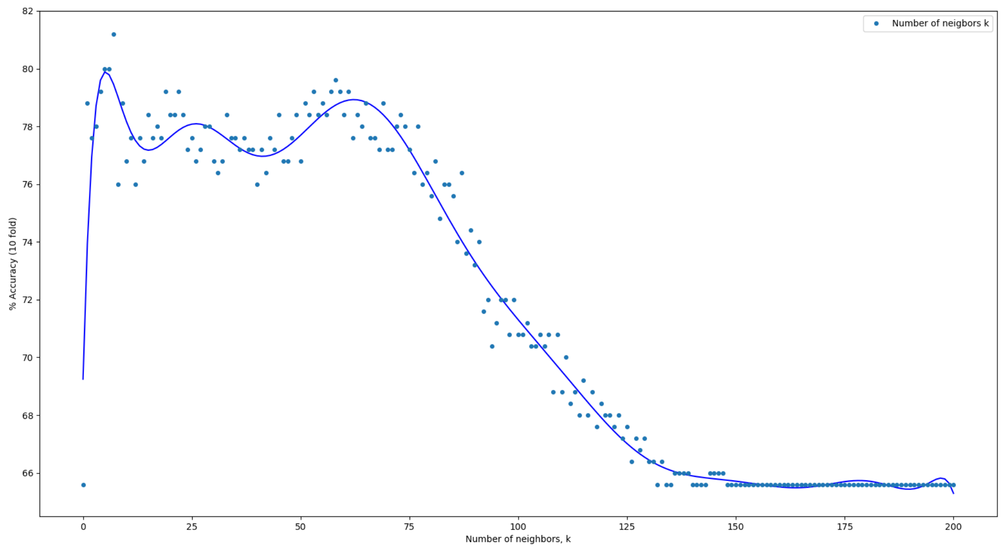


Figure 2: kNN performed for k = 1 to 200.

Using a k of seven, we ran kNN ten times, leaving us with the true positive (TP) and false positives (FP) rates show in table 1, which allowed us to generate the ROC curve shown in figure 3.

Table 1: kNN TP and FP rates

TP	0.800	0.714	0.250	0.667	0.556	0.444	0.583	0.636	0.700	0.800
FP	0.200	0.000	0.095	0.125	0.063	0.000	0.000	0.071	0.067	0.300

We then performed SVM on our dataset ten times using non-linear kernels. We used the scikit library [7] for Python to perform this using the default parameters. It generated the TP and FP rates shown in table 2, which led us to the ROC curve shown in figure 4.

Table 2: SVM TP and FP rates

TP	0.222	0.857	0.818	0.800	0.667	0.588	0.333	0.714	0.833	0.444
FP	0.000	0.000	0.000	0.050	0.063	0.000	0.053	0.111	0.105	0.000

Both ROC curves were above the y = x line, meaning that our algorithms were successful. Taking into account the different scalings of the x-axes, it is clear that SVM generally outperformed kNN. However, the goal of our project was focused on the implications of dimensionality reduction, so that was our next step.

Using scikit, we implemented PCA on our dataset to generate variances for all of our features. The goal of PCA is to select features that have the highest variance, and exclude those with lower variances. The idea is to choose the features that are most significant for classifying data. Thus, we were left with table 3 showing the variances for all of our variables.

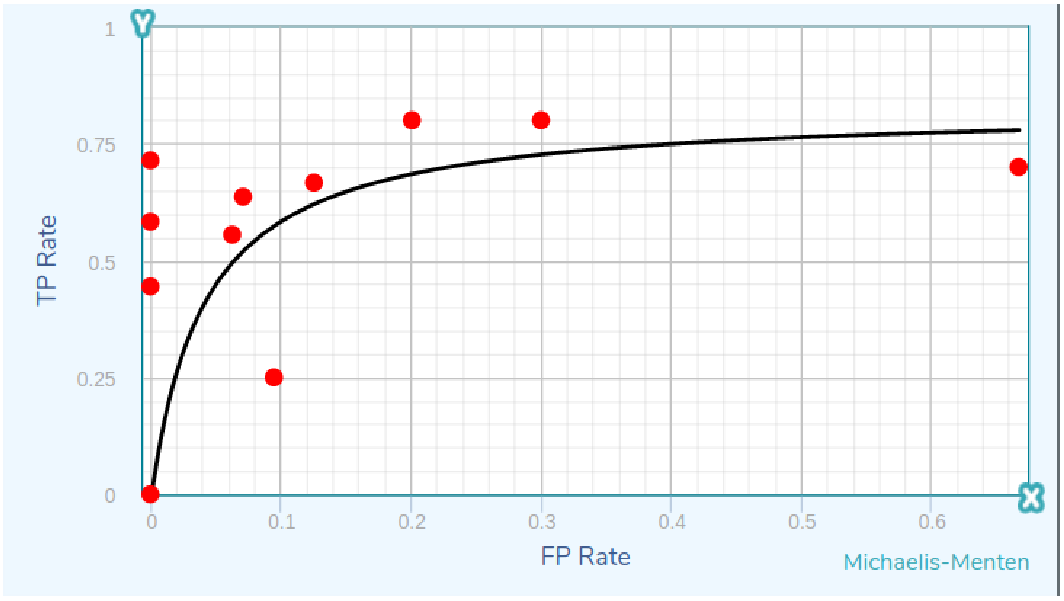


Figure 3: ROC curve generated from kNN algorithm runs.

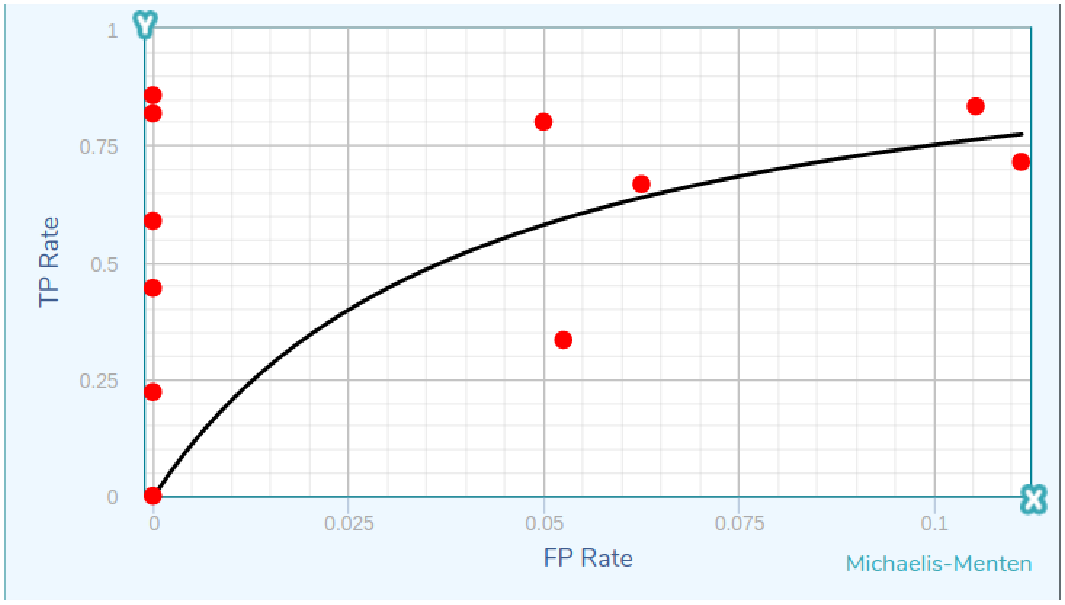


Figure 4: ROC curve generated from SVM algorithm runs.

We then re-ran our kNN and SVM algorithms, but only using a subset of dimensions varying from one to twelve. The dimensions were selected by choosing the highest variance features from the table in descending order. For example, a one-dimensional run only took into account cloud water, whereas a two-dimensional run used both cloud water and Haines index. We generated accuracies for each dimension subset, for each algorithm, and displayed it in the graph shown in figure 5.

As we can see here, SVM and kNN performed similarly in the feature selection cases shown in figure 5.

Table 3: Feature Variances

Cloud Water	0.385
Haines Index	0.217
Precipitable Water	0.116
Wind Pressure	0.096
Surface Pressure	0.055
Relative Humidity	0.041
Wind Temperature	0.038
Surface Temperature	0.022
Cloud Cover	0.016
Total Ozone	0.009
U-Wind Velocity	0.003
V-Wind Velocity	0.002

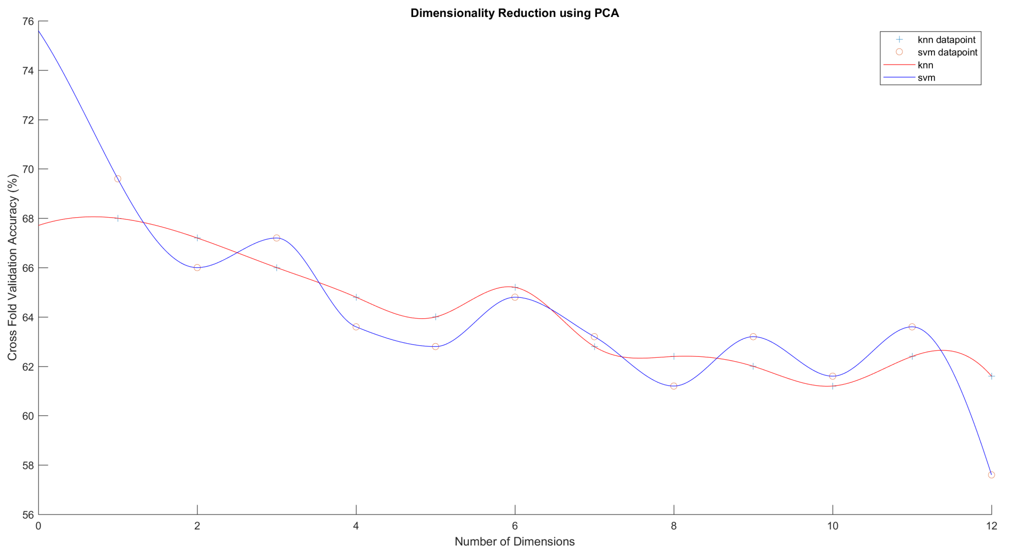


Figure 5: Comparison of kNN and SVM performance after PCA

5 Analysis and Discussion

The original runs using all of our features showed ROC curves for both kNN and SVM that were above the y = x line. This is significant because it proves that our methods were viable and somewhat effective at predicting rain. While SVM outperformed kNN on the full feature set most of the time, both of the algorithms performed similarly on feature subsets after PCA. While we only used two algorithms in this project, it is important to determine the best one for any use-case.

Using PCA dimensional reduction produced interesting results. We can conclude from our graph in figure 5 that out of our twelve feature set, using more than the three highest variance features actually started to cause our accuracy to decrease. If we take a look at table 3, we can see that all features after the third one have incredibly low variances. Usually, accuracy tends to increase with the more dimensions one has. However, since we are working with atmospheric data, having more features does not always lead to more accurate results. We had many noisy features that had low variance and contributed little to predicting rain. Limiting our dimensions to just cloud water, Haines index, and precipitable water produced the most accurate and consistent results, while including the other dimensions caused that to deteriorate in both accuracy and efficiency.

These results have interesting implications for predicting rain in general. We can conclude that analyzing all the features the GFS provides would not only be very inefficient, but it would also harm the accuracy of the classifier. As mentioned earlier, there is not a large consensus of features that are significant in rain prediction. There are some features that are widely used and accepted, but many others that are relatively untested. Our work is a solid example of how to identify significant features in rain prediction.

While our work provides meaningful results, it does come with some weaknesses. The two biggest weaknesses would be our feature set and our data. We only used twelve features provided by the GFS, however our results have made it clear that in order to fully determine the best features to use in predicting rain, it would be best to analyze all the features. This would ideally give us a set of high-variance features that are all impactful in classifying rain. The problem with our dataset is that it could have been much larger. We analyzed 250 days of precipitation data, for the year of 2014 in Orlando, Florida. However, the GFS website does not provide a batch download allowing one to specify the features and location they want. Instead, manual downloading of each individual file is required and that is incredibly time consuming. However, these weaknesses do not detract from the general conclusions we can make about our work, namely, the ability to identify important features in rain prediction.

6 Conclusions and Future Work

In this project, we extended Coblenzs [1] previous work to incorporate PCA dimensionality reduction to a set of features used in rain prediction. We collected our data from the GFS and weather underground, performed kNN and SVM, and then selected high-variance features for re-testing and identifying of significant features. Our results were significant and implied that many atmospheric features are actually noisy and detract from both the accuracy and efficiency of a precipitation classifier.

Given more time, future work on this project would incorporate a few things. Firstly, we would spend more time collecting data. Since it has to be done manually, it would take a long time, however having thousands of days of data as opposed to a few hundred can give our results more meaning. Secondly, we would incorporate more machine learning algorithms. We would want to test out other methods such as a nave Bayes classifier and random forests to compare their performance, especially when used in conjunction with PCA. Thirdly, we would want to expand our feature set. Testing out all the variables the GFS has to offer would be an ideal scenario. The size of the data would be large, however the results would be worth it. Lastly, we would want to test out our methods for more locations than Orlando, Florida. This goes hand-in-hand with collecting more data. Analyzing different locations could provide some interesting results. Perhaps different locations around the world have different subsets of features that are the most significant in predicting rain.

References

[1] Coblenz, Joshua. Using Machine Learning Techniques to Improve Precipitation Forecasting. (2015).

[2] Viana, Denilson and Sansigolo, Clvis. Monthly and Seasonal Rainfall Forecasting in Southern Brazil Using Multiple Discriminant Analysis. (2016).

[3] Du, Jinglin. Liu, Yayun. Yu, Yanan. Yan, Weilan. A Prediction of Precipitation Data Based on Support Vector Machine and Particle Swarm Optimization (PSO-SVM) Algorithms (2017)

[4] NOAA GFS Data Access,
`http://www.nco.ncep.noaa.gov/pmb/products/gfs`

[5] NOAA NEXRAD Data Access,
`https://www.ncdc.noaa.gov/data-access/radar-data/nexrad`

[6] Weather Underground Data Access,
`https://www.wunderground.com/history`

[7] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.
`http://scikit-learn.org/stable/index.html`

[8] NetCDF Operator, Charlie Zender et al., 1998.
`http://nco.sourceforge.net`