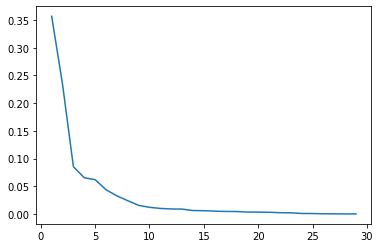
**Binning Dataset**

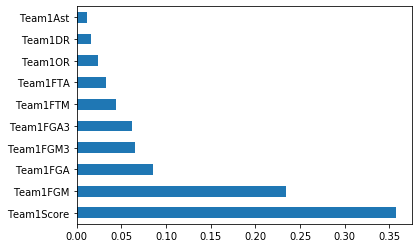
We binned the dataset by running Microsoft Business Intelligence clustering algorithm to determine how the dataset features should be binned. We went through each attribute and binned it according to the clusters that the algorithm discovered. The main idea in how we binned our data was to look at each distinct cluster and bin according to its mean +/- the standard deviation. This varied a little bit as some bins were left with few values, so we tried to keep the bins relatively equal in size.

**Principle Component Analysis**

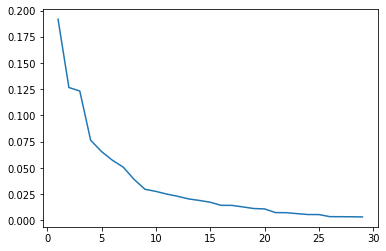
We used principle component analysis (PCA) to select the best features. We applied PCA to both our unbinned data and also our binned data. We had to drop the Team1 and Team2 attributes before running PCA because those two attributed explained 100% of the variance. After removing those two attributes PCA found that about 5 to 10 attributes explained the majority of the variance for the unbinned dataset and 10-15 for the binned datasets.



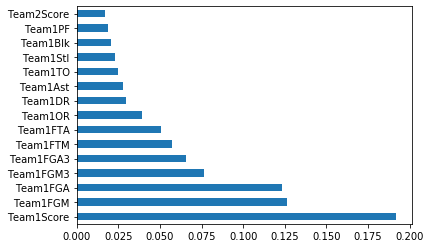
Variance explained for unbinned dataset



Top 10 attributes of variance explained for unbinned dataset



Variance explained for binned dataset



Top 15 attributes of variance explained for binned dataset

Its important to note that for some reason PCA would find that Team1’s attributes were more important that Team2’s attributes of the same statistic even though the team that won the game was randomly selected to be either Team1 or Team2. We believe the reason for this is because the label “Win” indicates whether Team1 wins or loses the match.

For PCA’s binned data we selected the top 10 attributes in the bar chart with their team ID along with their mirror stats for team 2. For PCA’s unbinned data we selected the top 9 attributes plus the team ID and team 2 mirror stats.

**TensorFlow Artificial Neural Networks**

We originally were using Microsoft Business Intelligence, but because of the limited control we had over fine-tuning the neural networks we switch to using TensorFlow in Python, which gave us more control to optimize the neural networks.

We made numerous neural networks for this project experimenting with a variety of different structures, hyper parameters with different attributes from the dataset. We fine-tuned each of the neural networks to produce the best results for each of the models. Early stopping was used for each model and training was stopped after the next three epochs produced a higher validation loss than the one before it. The relu activation function performed the best on all models. Using a dropout layer of 20% right before the output layer gave the best performance for all models. Binary cross entropy was used as the loss function and the optimizer was adam. Batch normalization improved the performance of almost all of the unbinned models, which provides regularization of the data passing between the layers by normalizing the batch distribution around the mean and batch normalization was not used on any of the binned datasets because it reduced performance. The output layer of all the models used a sigmoid activation function with only one output neuron to determine if Team1 won or lost the match. The same random seed was fixed so we could reproduce the results. The training size 65628 out of 87504 for 75% of the dataset with a validation split of 20% and the testing size was 21876 of 87504 for 25% of the dataset. The number of layers and the number of neurons in each layer varied for every model.

**Microsoft Business Intelligence**

In Microsoft Business Intelligence (MSBI) we ran all of the learning algorithms available, this includes naïve bayes, decision trees, clusting, association rules, logistic regression, and a neural network. All the algorithms in MSBI used a training set of 61253 for 70% of the dataset and a testing set of 26251 for 30% of the dataset. All MSBI classification algorithms and association rules ran on binned data

**Results**

One of the things we found was that binning our data usually reduced our learning algorithms performance and sometimes had similar results, but never had better results than our unbinned data.

**Improvements**

Some areas that we would have liked to improve on the project, but we ran out of time would have been to do integrate the players statistics into our dataset. We believe adding more attributes to our dataset would have increased our algorithms performance. Another area would be to do a time series analysis of the matches with deep learning. This would have allowed us to incorporate more deep neural networks such as recurrent neural networks and convolutional neural networks that have the potential to learn much better than artificial neural networks. Deep learning has significant advantages over other learning algorithms. One of these benefits is that very large dataset won’t suffer from overfitting like other learning algorithms suffer from. Another benefit is that they both can be used in time series analysis. If time wasn’t as issue for this project, we would have been able to complete these two issues.

**Discussion**

Neural networks, adaBoost, and support vector machines performed the best. This is not surprising because these three learning algorithms are some of the most powerful algorithms available. In the vast majority of cases binning our data reduced our algorithms performance. Binning our data set had the advantage of reducing the time needed to run the algorithms.

MSBI’s learning algorithms had limitations that we did not encounter in TensorFlow and sk-learn. MSBI learning algorithms couldn’t run on continuous attributes and all of our attributes were continuous data except the label. This required that we had to bin our data if we were to use MSBI’s learning algorithms, but our algorithms performed better in TensorFlow and sk-learn unbinned. We had greater ability to fine-tune our learning algorithms in TensorFlow and sk-learn than we did in MSBI.