Jennifer Barry, Landon Grim, Claudia Moeller
CSE 5243 - Dr. Van Hulse
4/24/2017

Final Report

Introduction

In recent years, sports such as basketball have seen an increase in the use of data analytics. In 2013, the National Basketball Association (NBA) installed player-tracking systems into each team's arena. This data, and subsequent analysis, has led to a change in strategy in the NBA. In college basketball's tournament "March Madness," over 70 million brackets will be filled out with an estimated $10.4 billion gambled [4]. With so much money on the line, there is a large market for data analysis identifying the main factors which go into deciding a basketball game, and improving the accuracy with which one can predict a game's outcome. In particular, with the NBA playoffs just starting, with what accuracy can we predict professional playoff basketball games be predicted?

Among the most popular factors used in current predictions are what are known as the "four factors." These four factors, as introduced by Dean Oliver, the ESPN Director of Analytics, are shooting, turnovers, rebounding, and free throw proficiency [1].

The metric for shooting is effective field goal percentage, which is a team's adjusted shooting percentage. The formula weights a three-pointer more heavily than a two pointer: (FGM + .5*3PM) / FGA. Turnovers is an estimate of the number of turnovers per 100 possessions. Rebounding is simply an estimate of the percentage of rebounds a team grabs. Finally, free throws is a calculation of free throws attempted over field goals attempted (FTA / FGA) by the team.

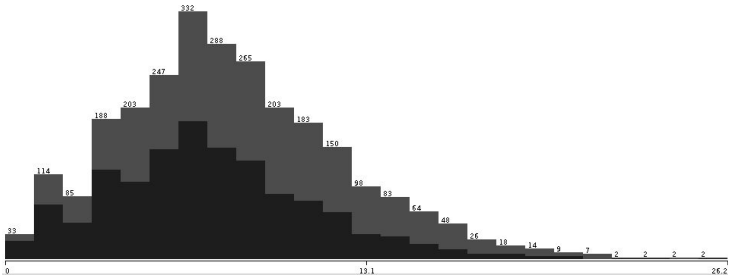
Prior to predicting any games, a dataset needed containing all of the above information must be collected. Fortunately, basketball-reference.com contains the history of all the NBA games which recorded these statistics [2]. The website also conveniently calculated the four factors of each game, however, did not have a dataset one could conveniently download. To obtain all desired information, a custom web scraper was written to fetch statistics for the entirety of the 2016 - 2017 NBA season. This web scraper was written in Ruby and used the Nokogiri gem for scraping the HTML. There was difficulty parsing the data as the actual data in the web page was contained within HTML comments. These needed to be transformed so in order to actually obtain the data. The web scraper parsed the score of the game to determine which team won the game, the four factors, and other advanced basketball statistics including steal percentage, block percentage, and assist percentage. Four CSV datasets were created using this data. The first dataset, 2016_results.csv ("Four Factors"), included the team name, the four factors, and whether the team won or lost. The next dataset, 2016_results_opp_team.csv ("Four Factors w/ Opposing Teams"), included the four factors, whether the team was home or away, assist percentage, block percentage, steal percentage, and whether the team won or lost. The third dataset, 2016_advanced_statistics.csv ("Advanced Stats"), included what was in the first dataset, however it also included the opposing team's four factors as well. Finally, the last data set, 2016_advanced_results_opp_team.csv ("Advanced Stats w/ Opposing Teams"), included what was in the second dataset, but it also included the other advanced statistics of the other team as well. All of these datasets include 2,666 rows of data.

Exploratory Data Analysis

Exploratory data analysis was performed on the generated datasets using the data visualization tools in the Weka explorer GUI. The majority of the attribute distributions were normal, but some of the more interesting distributions are displayed below.

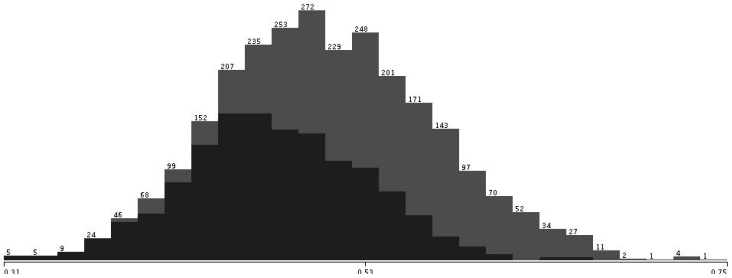
Blocks per game had the most skewed distribution, with a few high values skewing the distribution to the right. This could be explained by the fact that blocks per game can be greatly affected by one player. Teams with high-performing players could be skewing the distribution.

Blocks Per Game



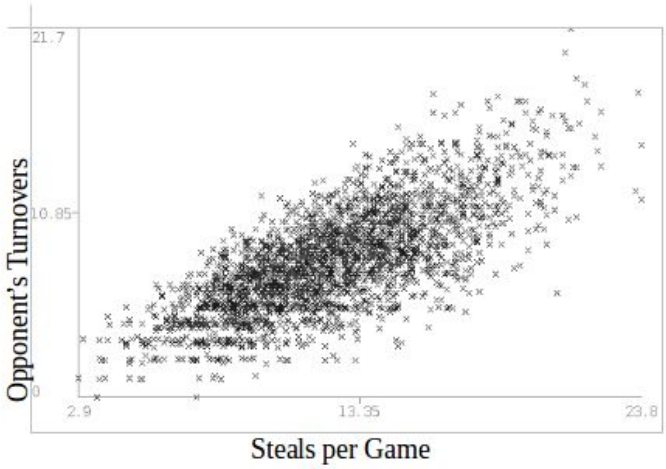
When comparing the distributions winning teams' attributes to losing teams', effective field goal percentage had the most noticeable differences. The histogram below uses black for losing team frequency, and grey for winning team frequency. It shows that winning teams are significantly more likely to have higher effective field goal percentages.

Effective Field Goal Percentage



Turnovers and steals were only two attributes that showed signs of correlation. This graph shows the potential correlation.

Opponent's Turnovers vs Steals per Game



This apparent correlation is intuitive because a steal implies an opponent turnover. However, turnovers can be caused by opponent mistakes, like throwing the ball out of bounds, so the attributes cannot be considered duplicates.

Goals

Through comparison of the four datasets described in the *Introduction*, this project aims to determine the highest accuracy with which one can predict the outcome of a basketball game for a particular team. The outcome of predictions generated knowing only the "Four Factors of Basketball," will be compared to those which leverage additional information. This will allow conclusions to be drawn about whether the four factors are truly indicative of a game's outcome. Additionally, games will be predicted both with and without complementary information about the opposing team in order to determine whether accuracy can be improved by taking into account the difficulty of the opponent. Finally, the accuracy of predictions for the outcome of future games will be analyzed. The most successful algorithms available online successfully predict games between 60-70% of the time, and this is the figure to which the models developed through this endeavor will be compared [3]. Ultimately, the project will strive to present the most accurate model possible for game prediction.

Techniques and Algorithms

A variety of different models will be developed and tested to determine which is most accurate. Each algorithm will be trained separately on each of the four available datasets, with the goal of determining the most predictive attributes. The classification algorithms that will be tested are: support vector machine, decision tree, naive bayes, artificial neural network, and an ensemble learner. The ensemble learner will be composed of the most accurate of the other classifiers. Implementations of these algorithms from both the weka.classifier library and the Python scikit learn library will be leveraged and compared. 10-fold cross-validation will be used to test the classifiers, and performance measures, including accuracy and f-measure, will be collected.

The performance measures of models trained and tested using only the 4 factors will be compared to those of models which classified based on a larger set of attributes. Similarly, models trained and tested with opposing team statistics will be compared to those without. These comparisons will determine

whether the same algorithm performs better or worse with additional information, and whether the most accurate type of algorithm changes when the additional information is added. Through this evaluation, the team will be able to determine whether only the 4 factors are ideal for predicting the outcome of games, or whether it is possible to develop a more accurate prediction based on additional information. The overall most accurate model will be fit to the entire data set on which it performed best with no hold-outs, and used to predict the outcomes of future games. The test data set in this case will be generated from team averages, and (if necessary based on earlier comparisons) information about the game itself such as whether the team will be playing at home or away. This performance will be compared to popular game prediction sites to yield conclusions about the relative performance of the resulting classifier. At its conclusion, the project will result in the creation of a model for predicting future games, and its accuracy to-date will be reported.

Implementation and Testing

For the Python implementations of each algorithm, the scikit-learn library was invoked. The following classes and parameters were used:

- Decision Tree: sklearn.tree.DecisionTreeClassifier
    - Splitting criterion specified as Entropy rather than default of GINI as a result of consistent, significant improvement in accuracy.
- Naive Bayes: sklearn.naive_bayes.GaussianNB
    - Only available parameter is prior probabilities for the classes, which are unavailable for the data sets being tested.
- Support Vector Machine: sklearn.svm.SVC
    - Kernel selected as ‘linear,’ all other parameters were only relevant to other kernel settings, not relevant to the datasets, or did not noticeably improve performance, and were thus left to default.
- Artificial Neural Network: sklearn.neural_netowork.MLPClassifier
    - All parameters were left to the default because other options did not cause a significant or consistent change in the performance of the classifier. In some cases, different values prevented convergence in a timely manner.
- Adaboost Classifier: sklearn.ensemble.AdaBoostClassifier
    - Support Vector Machine used as the base estimator since it is the most accurate of the other classifiers
    - Algorithm changed to SAMME in order to support use of SVM
- Voting Classifier: sklearn.ensemble.VotingClassifier
    - All of the non-ensemble classifiers above (decision tree, naive bayes, artificial neural network, and support vector machine) were used as estimators

These algorithms were also tested with the weka.classifier library. Various parameters for the different models were attempted, and the most successful models were reported.

- Decision Tree: weka.classifiers.trees.J48
    - Three models were trained for each dataset:
        - unpruned= false, reducedErrorPruning = true

        - unpruned = false, reducedErrorPruning = false
        - unpruned = true, reducedErrorPruning = false
    - unpruned is true when the tree is not pruned and reducedErrorPruning is true when the model uses specifically reduced error pruning
    - The third set of parameters gave the best results.
- Support Vector Machine: weka.classifiers.functions.SMO
    - The complexity parameter was optimized by the weka.classifiers.meta.CVParameterSelection classification library
    - The kernel used was weka.classifiers.functions.supportVector.PolyKernel
    - The calibration model used was weka.classifiers.functions.Logistic
- Naive Bayes: weka.classifiers.bayes.NaiveBayes
    - The default parameters were used.
- Artificial Neural Network: weka.classifiers.functions.MultilayerPerceptron
    - Models were trained with a learning rate of 0.3 and 0.1
    - The learning rate of 0.1 was more successful.
- Ensemble Classifier: weka.classifiers.meta.AdaBoostM1
    - An learner was trained with support vector machines and neural networks (with a 0.3) because they received the best results.

In both Python and weka, each dataset was pre-formatted by normalizing the continuous attributes to a range of [0,1] to prevent the skews observed in *Exploratory Data Analysis* to disproportionately affect the results. Additionally, the two datasets introducing "additional statistics" contained a categorical attribute, which was binarized for the purpose of classification.

All algorithms used were trained and tested individually on each of the four datasets using 10-fold cross-validation. The Python scikit-learn library provides a module which allows each model to easily be cross-validated on a provided data set, with an optional parameter for the number of "folds." A prediction was generated for each record in every dataset, and compared to the actual class for that record. The scikit-learn "metrics" library provided tools to generate accuracy and confusion matrices, and F-Measures were calculated from the confusion matrices. Similarly, the weka explorer GUI provides x-fold testing (a parameter of 10 was used) and a wide range of evaluation metrics, including a confusion matrix, f-measure and accuracy.

Model Analysis

Python

The following are the results of the Python scikit-learn implementations for each classifier tested, segregated by dataset:

Four Factors:

Decision Tree

Accuracy: 0.649662415604

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	853	480
	Win	454	879

F-measure: 0.646212121212

Naive Bayes

Accuracy: 0.718304576144

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	968	365
	Win	386	947

F-measure: 0.720506140677

Artificial Neural Net

Accuracy: 0.727306826707

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	959	374
	Win	353	980

F-measure: 0.725141776938

Support Vector Machine

Accuracy: 0.733683420855

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	992	341
	Win	369	964

F-measure: 0.736451373422

Ensemble (Adaboost)

Accuracy: 0.720930232558

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1050	283
	Win	461	872

F-measure: 0.738396624473

Ensemble (Voting)

Accuracy: 0.726931732933

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	994	339
	Win	389	944

F-measure: 0.731958762887

Most Accurate Classifier: Support Vector Machine

Accuracy: 0.733683420855

F-Measure 0.736451373422

Four Factors with Opposing Teams:

Decision Tree

Accuracy: 0.84096024006

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1126	207
	Win	217	1116

F-measure: 0.841554559043

Naive Bayes

Accuracy: 0.918229557389

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1224	109
	Win	109	1224

F-measure: 0.918229557389

Artificial Neural Net

Accuracy: 0.955738934734

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1269	64
	Win	54	1279

F-measure: 0.955572289157

Support Vector Machine

Accuracy: 0.958364591148

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1278	55
	Win	56	1277

F-measure: 0.958380202475

Ensemble (Adaboost)

Accuracy: 0.936234058515

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1248	85
	Win	85	1248

F-measure: 0.936234058515

Ensemble (Voting)

Accuracy: 0.951237809452

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1293	40
	Win	90	1243

F-measure: 0.952135493373

Most Accurate Classifier: Support Vector Machine

Accuracy: 0.958364591148

F-Measure 0.958380202475

Advanced Stats:

Decision Tree

Accuracy: 0.659789947487

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	888	445
	Win	462	871

F-measure: 0.661945583302

Naive Bayes

Accuracy: 0.747561890473

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1033	300
	Win	373	960

F-measure: 0.75428988682

Artificial Neural Net

Accuracy: 0.752813203301

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1001	332
	Win	327	1006

F-measure: 0.752348741075

Support Vector Machine

Accuracy: 0.758814703676

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1022	311
	Win	332	1001

F-measure: 0.760699665054

Ensemble (Adaboost)

Accuracy: 0.706676669167

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1044	289
	Win	493	840

F-measure: 0.727526132404

Ensemble (Voting)

Accuracy: 0.756564141035

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1055	278
	Win	371	962

F-measure: 0.764769844146

Most Accurate Classifier: Support Vector Machine

Accuracy: 0.758814703676

F-Measure 0.760699665054

Advanced Stats with Opposing Teams:

Decision Tree

Accuracy: 0.830832708177

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1105	228
	Win	223	1110

F-measure: 0.830514844044

Naive Bayes

Accuracy: 0.867966991748

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1157	176
	Win	176	1157

F-measure: 0.867966991748

Artificial Neural Net

Accuracy: 0.953863465866

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1269	64
	Win	59	1274

F-measure: 0.953776775648

Support Vector Machine

Accuracy: 0.955738934734

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1274	59
	Win	59	1274

F-measure: 0.955738934734

Ensemble (Adaboost)

Accuracy: 0.688672168042

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	918	415
	Win	415	918

F-measure: 0.688672168042

Ensemble (Voting)

Accuracy: 0.941860465116

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1298	35
	Win	120	1213

F-measure: 0.943656852054

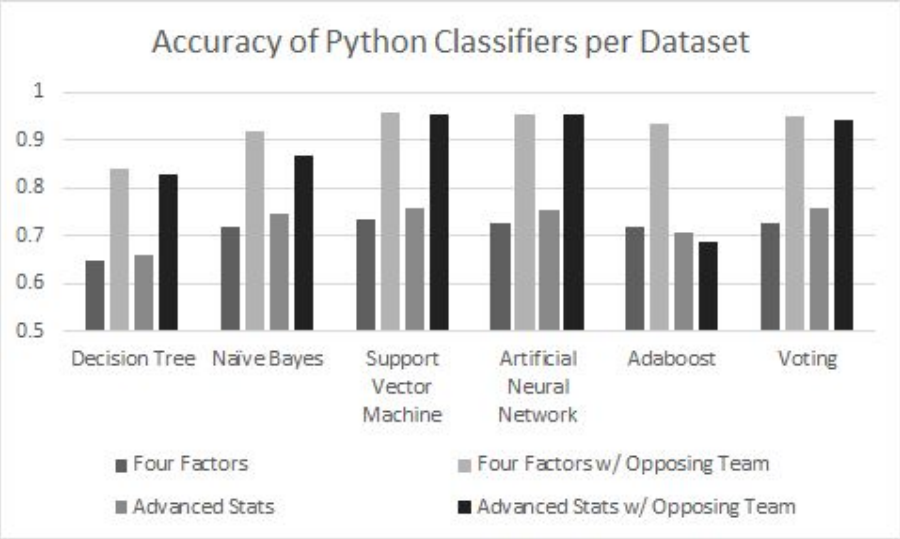
Most Accurate Classifier: Support Vector Machine

Accuracy: 0.955738934734

F-Measure 0.955738934734

Summary of Python Results:

Dataset	**Classifier**	**Accuracy**	**F-measure**
Four Factors	Support Vector Machine	0.73	0.74
Four Factors w/ Opposing Teams	Support Vector Machine	0.96	0.96
Advanced Stats	Support Vector Machine	0.76	0.76
Advanced Stats w/ Opposing Teams	Support Vector Machine	0.96	0.96



The support vector machine algorithm outperformed all other algorithms tested for all four datasets. The Majority Voting Classifier and Artificial Neural Net performed nearly as well, however both are more complex than the Support Vector Machine. In nearly all cases, the datasets which included information about the opposing team resulted in higher accuracy than those which did not. Without opposing team information, the Advanced Stats dataset resulted in higher accuracy, however when opposing team information was included, the Four Factors dataset resulted in equal or higher accuracy. Due to the lower dimensionality of the Four Factors w/ Opposing Team dataset, and the lack of additional accuracy, it is favored over the Advanced Stats with Opposing Teams dataset.

Weka

The following are results using the weka library through the weka GUI. For models that have adjustable parameters, the best results are displayed.

Four Factors:

Decision Tree

Accuracy: 0.702176

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	937	396
	Win	398	935

F-measure: 0.702

Naive Bayes

Accuracy: 0.715679

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1001	332
	Win	426	907

F-measure: 0.715

Artificial Neural Net

Accuracy: 0.726557

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1027	306
	Win	423	910

F-measure: 0.727

Support Vector Machine

Accuracy: 0.735184

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1001	332
	Win	374	959

F-measure: 0.735

Ensemble (SVN)

Accuracy: 0.736309

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	999	334
	Win	369	964

F-measure: 0.736

Ensemble (Neural Network)

Accuracy: 0.723181

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1066	267
	Win	471	862

F-measure: 0.722

Most Accurate Classifier: Ensemble Learner using SVM's

Accuracy: 0.736309

F-Measure 0.736

Four Factors with Opposing Team Stats:

Decision Tree

Accuracy: 0.831583

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1107	226
	Win	223	1110

F-measure: 0.832

Naive Bayes

Accuracy: 0.914854

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1221	112
	Win	115	1218

F-measure: 0.915

Artificial Neural Net

Accuracy: 0.954614

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1274	59
	Win	62	1271

F-measure: 0.955

Support Vector Machine

Accuracy: 0.957239

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1275	58
	Win	56	1277

F-measure: 0.957

Ensemble (SVM)

Accuracy: 0.955364

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1275	58
	Win	61	1272

F-measure: 0.955

Ensemble (Neural Network)

Accuracy: 0.948987

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1272	61
	Win	75	1258

F-measure: 0.949

Most Accurate Classifier: Support Vector Machine

Accuracy: 0.957239

F-Measure 0.957

Advanced Stats:

Decision Tree

Accuracy: 0.699175

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	966	367
	Win	435	898

F-measure: 0.699

Naive Bayes

Accuracy: 0.749437

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1038	295
	Win	373	960

F-measure: 0.749

Artificial Neural Net

Accuracy: 0.744186

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1050	283
	Win	399	934

F-measure: 0.744

Support Vector Machine

Accuracy: 0.754689

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1023	310
	Win	344	389

F-measure: 0.755

Ensemble (SVM)

Accuracy: 0.754314

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1019	314
	Win	341	992

F-measure: 0.754

Ensemble (Artificial Neural Network)

Accuracy: 0.732933

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	986	347
	Win	365	968

F-measure: 0.733

Most Accurate Classifier: Support Vector Machine

Accuracy: 0.754689

F-Measure 0.755

Advanced Stats with Opposing Teams:

Decision Tree

Accuracy: 0.821455

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1091	242
	Win	234	1099

F-measure: 0.821

Naive Bayes

Accuracy: 0.867967

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1161	172
	Win	180	1153

F-measure: 0.868

Artificial Neural Net

Accuracy: 0.939235

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1253	80
	Win	82	1251

F-measure: 0.939

Support Vector Machine

Accuracy: 0.956864

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1276	57
	Win	58	1275

F-measure: 0.957

Ensemble (SVM)

Accuracy: 0.951988

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1267	66
	Win	62	1271

F-measure: 0.952

Ensemble (Artificial Neural Network)

Accuracy: 0.935859

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	1246	87
	Win	84	1249

F-measure: 0.936

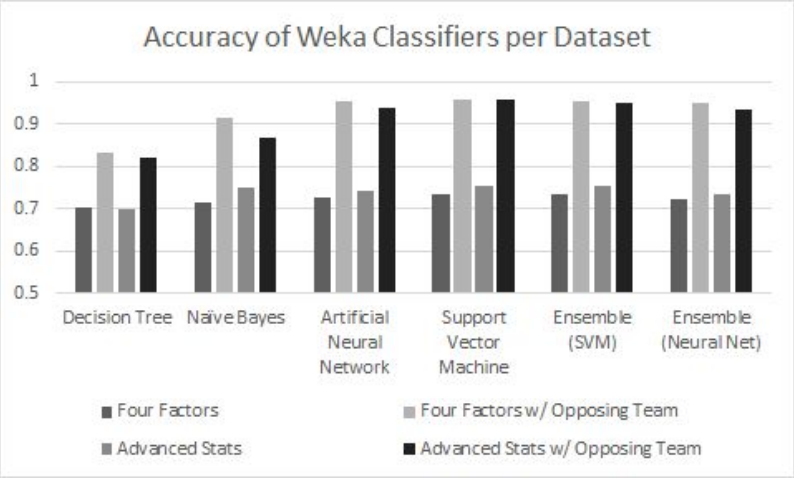
Most Accurate Classifier: Support Vector Machine

Accuracy: 0.956864

F-Measure 0.957

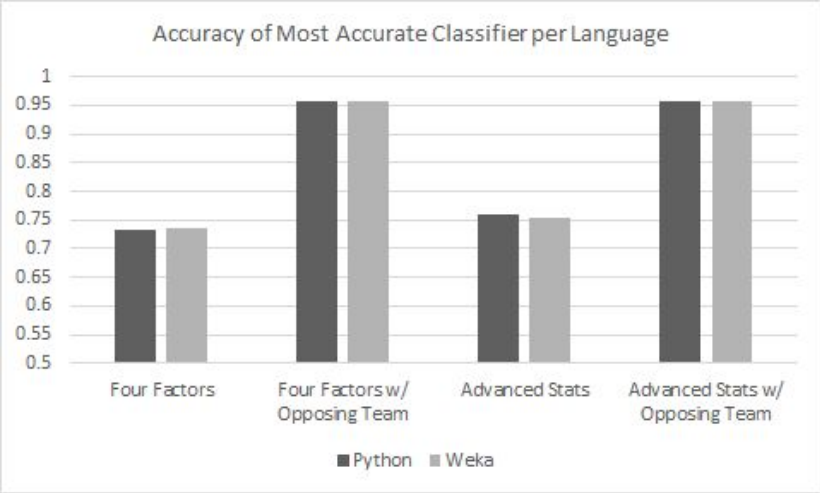
Summary of Weka Results:

Dataset	**Classifier**	**Accuracy**	**F-measure**
Four Factors	Adaboost with SVM's	0.74	0.74
Four Factors w/ Opposing Teams	Support Vector Machine	0.96	0.96
Advanced Stats	Support Vector Machine	0.75	0.75
Advanced Stats w/ Opposing Teams	Support Vector Machine	0.96	0.96



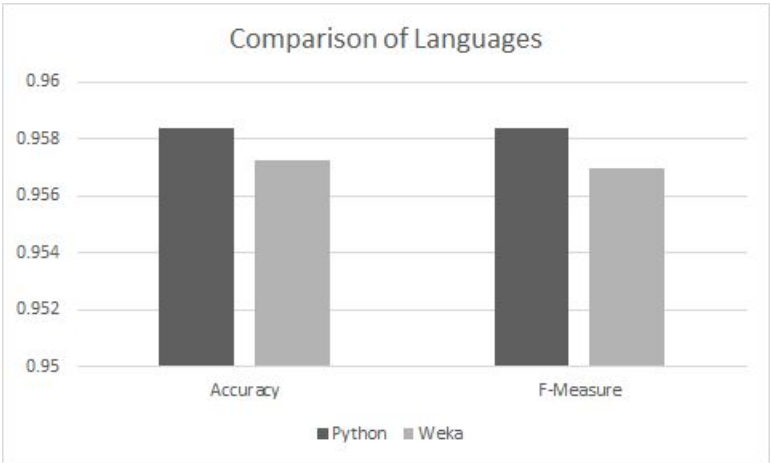
The support vector machine(SVM) was the most accurate of the models with all datasets except the four factors, for which the ensemble learner made of support vector machines was the most accurate. Artificial neural networks and ensembles of neural nets were also very successful. However, SVM's are much less complex than the other top performing models. Similarly to the python results, the SVN implementation will be evaluated as the most optimal model for this task.

Comparison



The chart above compares the accuracies of the most accurate Python and Weka classifiers for each dataset. In the case of the Four Factors dataset, the classification algorithm differs (Support Vector Machine vs Adaboost with SVM as the base classifier), however in all other cases both languages generated the highest accuracy with the Support Vector Machine. This indicates that SVM is the optimal model of those tested, independent from the dataset in nearly all cases.

Both languages exhibited similar behavior between datasets, with the highest accuracies coming from datasets including Opposing Teams information. Between Advanced Stats w/ Opposing Teams and Four Factors w/ Opposing Teams, the Four Factors w/ Opposing Teams set yielded higher accuracies by an extremely small margin. In addition to these higher accuracies, it is the ideal dataset over Advanced Stats w/ Opposing Teams because it has a smaller number of attributes. As Occam's razor implies, given that the two sets yield nearly equal results, the simpler of the two is better.

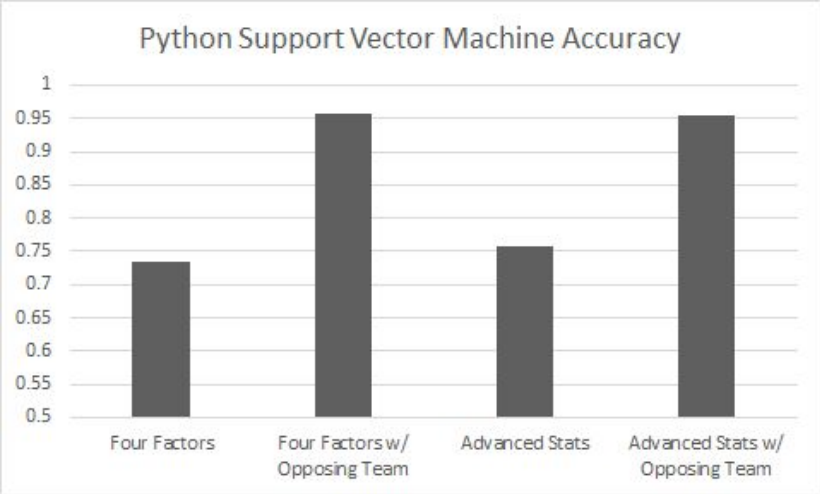


Within the Four Factors w/ Opposing Teams dataset, the Python implementation of the Support Vector Machine outperforms Weka, again by an extremely small margin. The bar graph above compares the

highest accuracy and f-measure achieved by Python to those achieved by Weka. Given the goals of the project, which require a single optimal model to be selected, the Python implementation will be used in the final analysis.

Conclusions

This project sought to answer three primary questions about basketball game prediction, outline in the *Goals* section.



Accuracy of the Support Vector Machine by Dataset

Can a game be predicted accurately using only the Four Factors? It would appear not. Of all four datasets, the Four Factors set yielded the lowest accuracy when cross-validated. This is true for the Python SVM, shown above, and in all but one of the algorithms tested in the *Model Analysis* section.

Can accuracy be improved by accounting for the performance of the opponent? The results strongly suggest that it can. For the Python SVM, the accuracy improved by approximately 20% when the opposing teams' information was added to their respective datasets. While not all models saw such a significant increase in accuracy, there is only one case in which the "Opposing Teams" datasets were outperformed by their counterparts. This is the same algorithm (Python's Adaboost) that served as the exception to the "Four Factors" question. It is suspected that the extremely high dimensionality of the "Advanced Stats w/ Opposing Teams" dataset played a role in this low performance, potentially due to overfitting..

Finally, how does the optimal algorithm compare to current game predictors? In order to draw conclusions about the accuracy of the chosen model relative to that of popular game predictors, Python's scikit-learn Support Vector Machine was trained using the "Four Factors w/ Opposing Teams" dataset. As previously mentioned, this dataset was chosen as the optimal training set since it yielded the highest accuracy when used in cross-validation.The lower dimensionality of the chosen training set also provides an advantage over the Advanced Stats w/ Opposing Teams set, which performed nearly as well. After fitting the model to the training dataset, it was used to predict the outcome of previously unseen games: the 2017 playoffs (playoff_statistics_predictions.csv). This dataset contains the same attributes as Four

Factors w/ Opposing Teams. In order to simulate prediction of future games, the attributes' values are an average of past team performance, with the exception of "class," which reflects the actual outcome of the game. These averages were obtained from the training data set (the 2016-2017 regular season), which in this case reflects the most up-to-date team statistics.

The results of the test are included in the dataset playoffs_results.csv.

As shown in the image above, the predictions are consistent for a single game (adjacent records in which the team name in the first is the opposing team in the second, and vice versa). This is the expected behavior for the classification model.

Team Name	Opposing Team	Result	Prediction
IND	CLE	loss	loss
CLE	IND	win	win
UTA	LAC	win	loss
LAC	UTA	loss	win
MEM	SAS	loss	loss
SAS	MEM	win	win
MIL	TOR	win	loss
TOR	MIL	loss	win
CHI	BOS	win	loss
BOS	CHI	loss	win
POR	GSW	loss	loss
GSW	POR	win	win
OKC	HOU	loss	loss
HOU	OKC	win	win
ATL	WAS	loss	loss
WAS	ATL	win	win
IND	CLE	loss	loss
CLE	IND	win	win
MEM	SAS	loss	loss
SAS	MEM	win	win
CHI	BOS	win	loss
BOS	CHI	loss	win
UTA	LAC	loss	loss
LAC	UTA	win	win
MIL	TOR	loss	loss
TOR	MIL	win	win
POR	GSW	loss	loss
GSW	POR	win	win
OKC	HOU	loss	loss
HOU	OKC	win	win
ATL	WAS	loss	loss
WAS	ATL	win	win
CLE	IND	win	win
IND	CLE	loss	loss
SAS	MEM	loss	win
MEM	SAS	win	loss

A portion of playoffs_predictions.csv formatted to highlight (in)correct predictions

The only games that were misclassified in the playoffs were considered upsets by most experts, as well. For example, in two cases Chicago, the last seed in the East, beat Boston, the first seed in the East. The two seed in the West, San Antonio, also lost to the underdog Memphis Grizzlies twice, which again is another surprise [5]. As the playoffs continue, games will be played between more closely seeded teams, which may significantly impact accuracy. The overall performance measures for the test are indicated below:

Playoffs Test Data:

Accuracy: 0.678571428571

F-measure: 0.678571428571

Confusion Matrix:

		Predicted	
		Loss	Win
Actual	Loss	19	9
	Win	9	19

The model achieved an accuracy of approximately 68%, which is at the upper end of the models referenced by Lee Richardson [3], however does not outperform the industry standard. Based on the results obtained using the test dataset, it is not possible to disprove that an accuracy of 70% is the best possible accuracy for predictions of future games. However, given the success of the model generated through this project relative to those used in practice, the team concludes that the approach taken to train the model and predict games will yield reliable results.

References

[1] Using the Four Factors. Retrieved 2017, April 10. http://www.sportingnews.com/ncaa-basketball/news/march-madness-2017-how-to-pick-an-ncaa-tournament-bracket-using-the-four-factors-of-basketball-success-stats/14xacw0ak7g9n1gzblode9oyjp.

[2] Basketball Reference. Retrieved 2017, April 3. basketball-reference.com.

[3] How Predictable is the NBA?. Retrieved 2017, April 3. http://nyloncalculus.com/2015/02/02/predictable-nba/.

[4] March Madness 2017: 70 Million Brackets, $10.4 Billion in Bets Expected. Retrieved 2017, April 4. http://bleacherreport.com/articles/2697846-march-madness-2017-70-million-brackets-104-billion-in-bets-expected.

[5] NBA playoffs 2017 live scores. 2017, April 22. http://www.cbssports.com/nba/news/nba-playoffs-2017-live-scores-first-round-series-results-updates-bracket-schedule-matchups/

Contributions

- Jennifer Barry was responsible for the Python portion of the project, including its description in *Implementation and Testing*, and processing of its results in the *Analysis* section. She also contributed the comparison of the languages' models, and the *Conclusion* section, due to the fact that one of the Python implementations proved to be the most accurate.
- Landon Grim was responsible for the web scraper, and thus the generation of the four training datasets, and the Playoffs testing dataset. Additionally, he contributed the *Introduction* section.
- Claudia Moeller was responsible for the Weka portion of the project, including its description in *Implementation and Testing*, and processing of its results in the *Analysis* section.

The *Goals* and *Techniques and Algorithms* sections were largely derived from the project proposal, and should be considered a team effort.