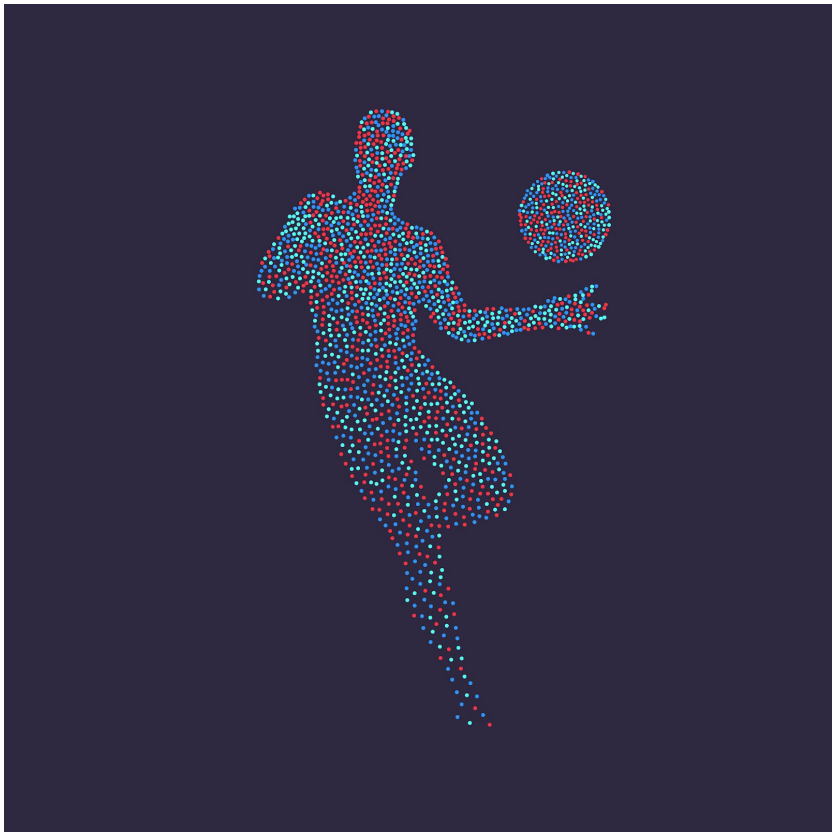

NBA Game Prediction with Non-linear Classifiers

Raven Nuega

CS 4641

Professor Brian Hrolenok

Contents

Introduction to the Dataset

Basketball is by far one of the most dynamic sports in the world today, yet the game seems deceptively simple. Ten players--five on one team, five on the other attempting to score on their respective baskets with a single ball. However, that is simply the tip of the iceberg. Underneath this beautiful game is a plethora of quantifiable properties ranging from points to the game's more mathematically dense statistics. In consequence, I chose a dataset taken from *stats.nba.com* [1] by a Github user [2] that consists of every game played in the last 3 seasons.

```
Index(['Home', 'Away', 'W_PCT', 'REB', 'TOV', 'PLUS_MINUS', 'OFF_RATING',
                  'DEF_RATING', 'TS_PCT', 'Result', 'Date'],
                                dtype='object')
```

Figure 1. Features of *COMBINEDgamesWithInfo2016-19.csv* dataset

The features found in Figure 1 represent team statistics. For instance, win percentage (`'W_PCT'`) is a statistic intrinsic to the team and corresponds to the percentage of total games played that resulted in a win. Another detail worth noting is that the rest of the statistics in the dataset are scaled per 100 possessions, meaning that the values represent performance in a category if the team only had 100 possessions [1].

However, the dataset numbers are processed even further. There are two teams: the home team and away team--each producing their respective statistics. Each statistic from both teams is then standardized using the league mean and standard deviation. After obtaining the z-scores, the away team's z-scores are subtracted from the home team's based on the statistical category (e.g. home reb% z-score - away reb% z-score) [2]. Standardization enables proper statistical comparison between the two teams, which allows analysts to evaluate the differences in performance.

The underlying supervised learning problem has taken shape. Given games as samples and the difference between the home team and away team's date-specific standardized scores, **what is the distribution of home team wins and home team losses?** Since the results are a mapping of inputs to binary labels representing whether the

home team wins (1) or loses (0), the problem is that of classification. Hence, classifiers such as support vector machines, random forests and multi-layered neural networks will be implemented to find the optimal hypothesis.

Metrics

Now that the machine learning problem is identified and the data is explored, the performance metrics have to be established. One of the most informative and intuitive tools used in analyzing classification algorithms is confusion matrices.

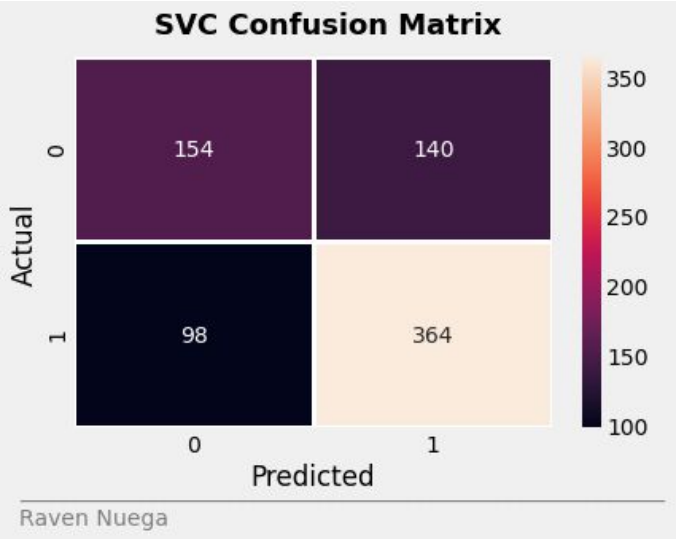

Figure 2. Confusion matrix for Support Vector Classifier on test data

A confusion matrix is a 2x2 square that displays four properties: true positives, true negatives, false positives, and false negatives [3]. This matrix assesses the true performance of a classifier because it combines all relevant classification metrics together. One metric taken out of context could misinform people on the true performance of the classifier. The list of classification metrics used to measure model performance are as follows:

- Predictive accuracy -- Accuracy can be calculated by counting the number of samples predicted correctly on the test dataset divided by the number of samples in the test dataset [3].
- Recall -- Recall measures the proportion of actual positives that were labeled by the classifier as positive. In other words, it is the number of true positives divided by the number of actual positives (true positive or false negative) [3]. This is integral to measuring performance of a dataset because a high score minimizes false negatives.
- Precision -- Precision measures the percentage of samples labeled positive that were actually positive. In essence, the metric finds the proportion of true positives from the group of predicted positives (true positives or false positives) [3]. It is important to score high on precision because it means that false positives are minimized.
- F1 Score -- F1 score is ultimately a combination of both precision and recall scores. Instead of taking the arithmetic mean, it takes the harmonic mean of both precision and recall scores [3]. The following formula shows how to calculate F1 score:

  F1 Score = 2 * Precision * Recall / (Precision + Recall) = 2*3*100/103 = 5%

  A harmonic mean is used as opposed to an arithmetic mean to avoid instances in which the difference between precision and recall values is high.

Description of Algorithms

Since the problem is a classification problem, the project uses three classifiers: **Random Forests with Bagging**, **Support Vector Classifier**, and a **Multi-Layer Perceptron Classifier**.

The first algorithm used in the project is Random Forests with Bagging, which is an algorithm that fits a collection of decision trees to the data. The algorithm can be broken down into two parts: bootstrap and aggregation. The bootstrap action is the act of randomly sampling the dataset **with replacement.** In addition, the split points are selected randomly instead of optimally. After all trees are trained, the algorithm aggregates the results and outputs the majority prediction [4]. The idea is to train an ensemble of weak

learners and combine them to form a strong learner. This combats the issue of overfitting that a single, optimally-split decision tree typically faces. Relevant hyperparameters include the following:

- n_estimators -- the amount of decision trees to train [5]. A high number of decision trees smoothes the output and adds bias [4] , while reducing the number estimators adds variance making the hypothesis space too expressive.
- max_features -- the number of features to consider when splitting a node in a decision tree [5]. Using all features for each learner makes for a more complex hypothesis space and using low number of features breeds high bias.
- bootstrap -- whether samples are drawn with replacement[5]. Setting it to true allows the learner to have non-unique samples which could increase bias. Setting it to false with max samples equal to number of samples permits the learner to access all samples, which could increase variance.

Support Vector Classifiers use support vector machines to find the optimal hyperplane that separates classes and maximizes the distances between sample points [6]. The hyperplane can have a soft margin which allows misclassification of training samples adding bias, or a hard margin in which the algorithm is not tolerant of misclassifications. SVC's are quite effective on binary classifications especially with balanced data. Relevant hyperparameters include the following:

- C -- the regularization parameter, particularly L2 regularization [7]. Lower C value will allow a larger-margin separating hyperplane, while a higher value produces a smaller margin. If margin is too large, the model underfits, whereas a very small margin overfits data.
- kernel -- used to map data to space to find hyperplane [7]. A linear kernel will work for linearly separable data. Non-linear kernels map data to higher dimensions to find hyperplane. The hypothesis space complexity depends on the hyperplane and hence, the selected kernel.

- gamma -- kernel coefficient [7]. Higher value coefficients tend to fit the data and thus lead to overfitting if too high. Lower value coefficients increase bias and could cause underfitting.

The last classifier is the Multi-Layer Perceptron Classifier or a deep neural network. Given the features and labels of the dataset, the hidden layers of perceptrons equipped with non-linear approximators (e.g. sigmoid) learn the best hypothesis by optimizing weights. These weights make up the linear summation of each perceptron and hence, the prediction of the network. The main idea is to minimize the loss function and provide a vector of class probabilities. Relevant hyperparameters include the following:

- solver -- optimizer tasked with minimizing error [8]. Since the data is large, the only two options are SGD and ADAM. The two differences that affect hypothesis space is learning rate and sample selection. SGD consists of different settings for learning rate, whereas ADAM employs its own learning rate setup. SGD randomly samples data, which increases bias and ADAM does not.
- learning_rate and learning_rate_init -- learning_rate_init sets the coefficient used in backpropagation to update weights [8]. If the learning rate is set too high, the optimization could diverge. If too low, the optimization could take too long. The learning_rate parameter configures the learning rate to change given certain conditions or to not change, which could prevent overfitting. This is only applicable to SGD.
- alpha -- L2 regularization parameter [8]. Higher values of alpha lowers variance and prevents overfitting. Lower values lowers bias and prevents underfitting.

Hidden_layer_size is not considered because it is a hyperparameter related to network structure, not the training algorithm. While structure influences performance, it is not considered in optimization because a neural network is dependent on its structure [9]. Once the architecture is altered, it is a different network to be optimized. Activation

function is also not tuned because it does not directly affect hypothesis space complexity only convergence and the time it takes [8].

Tuning Hyperparameters

Before tuning the hyperparameters of each classifier, the dataset's non-linearity requires justification. The most intuitive test for nonlinearity is a simple eye test; therefore, the features were reduced to 2D and 3D via PCA transformation in order to plot the data. Recall from homework 3 that PCA is the process of finding the orthogonal axes of maximal variance among the features and projecting the data onto said axes.

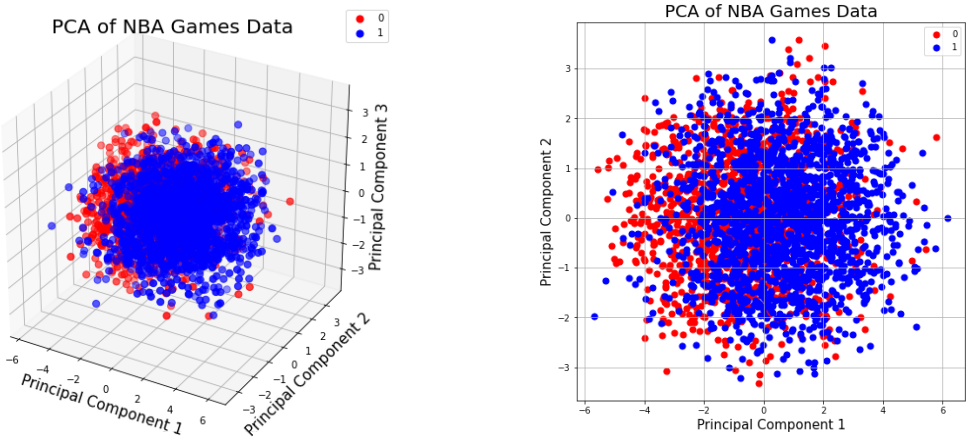

Figure 3. Three-dimensional and two-dimensional scatter plots of features

Data is regarded *linearly separable* if there exists a hyperplane between the two classes of data. According to these plots, the data is non-linear. In addition, the data was processed by a pair of linear classifiers: single-perceptron and SVM with linear kernel. The former yielded 47% accuracy on the training data while SVM had 64% accuracy. Recall that a single perceptron defines a decision boundary between two labels, yet it suffers from

non-linear data i.e. XOR target concept requires two hyperplanes to separate data [10]. In conclusion, the data is proven not only to be non-linear but also non-trivial because it consists of more than a 1000 data points with 8 features.

Hyperparameter tuning was conducted using sklearn's GridSearchCV, which leverages cross validation to find the optimal hyperparameters for each classifier. The cross validation utilized three folds for training and returned the mean score of each performance metric listed in the metrics section. Since the data is balanced, the classification problem prioritizes true positives and accuracy is prioritized [11].

	estimator	bootstrap	max_features	n_estimators	mean_accuracy	mean_f1	mean_recall	mean_precision
0	BaggingClassifier	True	2	25	0.617284	0.684125	0.720307	0.651416
1	BaggingClassifier	True	2	50	0.611993	0.677954	0.709579	0.649089
2	BaggingClassifier	True	4	50	0.611993	0.67631	0.704215	0.650568
3	BaggingClassifier	True	4	25	0.611552	0.676759	0.706513	0.6496
4	BaggingClassifier	True	1	50	0.606702	0.669071	0.691188	0.648484
5	BaggingClassifier	False	2	50	0.60582	0.669236	0.69272	0.647589
6	BaggingClassifier	False	2	25	0.601411	0.666225	0.691188	0.643065
7	BaggingClassifier	True	1	25	0.599647	0.668626	0.701916	0.638567
8	BaggingClassifier	True	4	10	0.594797	0.636223	0.616092	0.65833
9	BaggingClassifier	True	6	25	0.593915	0.662855	0.694253	0.634494
10	BaggingClassifier	True	6	50	0.59127	0.657365	0.681226	0.635204
11	BaggingClassifier	False	4	50	0.586861	0.651859	0.672031	0.633016
12	BaggingClassifier	True	2	10	0.585538	0.628579	0.609962	0.648871
13	BaggingClassifier	True	1	10	0.584215	0.630357	0.616092	0.645425
14	BaggingClassifier	False	2	10	0.583774	0.628506	0.612261	0.645942
15	BaggingClassifier	False	4	10	0.582892	0.625208	0.604598	0.647502
16	BaggingClassifier	False	4	25	0.578042	0.645903	0.668966	0.624684
17	BaggingClassifier	True	6	10	0.567901	0.610357	0.588506	0.634229
18	BaggingClassifier	False	6	50	0.559083	0.626644	0.643678	0.611008
19	BaggingClassifier	False	6	25	0.55291	0.622816	0.642146	0.605052
20	BaggingClassifier	False	6	10	0.544533	0.595347	0.583142	0.608668
21	BaggingClassifier	False	1	25	0.529541	0.595642	0.602299	0.58957
22	BaggingClassifier	False	1	50	0.525573	0.591063	0.596169	0.586669
23	BaggingClassifier	False	1	10	0.52425	0.588354	0.590805	0.586293

Figure 4. Bagging Classifier results from hyperparameter tuning

According to Figure 4, the enabling bootstrap generally performs better across all scores the learners can have duplicate samples. It is also important to note that num_features = 1.0 is equivalent to using all eight features. With that being said, two features paired with bootstrap and at least 25 estimators had the highest accuracy and f1 scores.

	estimator	C	gamma	kernel	mean_accuracy	mean_f1	mean_recall	mean_precision
0	SVC	1	auto	rbf	0.642857	0.712676	0.769349	0.664126
1	SVC	1	0.001	rbf	0.641534	0.721428	0.806897	0.652483
2	SVC	50	0.0001	rbf	0.641534	0.716405	0.786973	0.657655
3	SVC	10	0.0001	rbf	0.640653	0.718224	0.796169	0.654281
4	SVC	10	0.001	rbf	0.637125	0.729602	0.851341	0.638518
5	SVC	50	0.001	rbf	0.634039	0.727647	0.849808	0.636253
6	SVC	50	auto	poly	0.617725	0.726731	0.883525	0.617476
7	SVC	10	auto	poly	0.616843	0.730058	0.900383	0.6142
8	SVC	1	auto	poly	0.614638	0.734616	0.927203	0.608593
9	SVC	10	auto	rbf	0.61067	0.67943	0.716475	0.646348
10	SVC	50	auto	rbf	0.58157	0.646542	0.665134	0.629625
11	SVC	50	0.001	poly	0.575397	0.730479	1	0.575397
12	SVC	10	0.001	poly	0.575397	0.730479	1	0.575397
13	SVC	10	0.0001	poly	0.575397	0.730479	1	0.575397
14	SVC	1	0.0001	poly	0.575397	0.730479	1	0.575397
15	SVC	1	0.001	poly	0.575397	0.730479	1	0.575397
16	SVC	50	0.0001	poly	0.575397	0.730479	1	0.575397
17	SVC	1	0.0001	rbf	0.574956	0.730123	0.999234	0.575209

Figure 5. Support Vector Classifier results from hyperparameter tuning

For Support Vector Classifier, the rbf kernel was more accurate, and pairing rbf with a softer margin (low C value) yields higher accuracy scores. However, a polynomial kernel with a low C produces higher f1 scores. In both cases, the largest gamma value (auto = ⅛) is optimal. Since the dataset is balanced, true positives/negatives are prioritized. However, f1 scores are equally vital so the second row parameters were selected.

	estimator	alpha	learning_rate	learning_rate_init	solver	mean_accuracy	mean_f1	mean_recall	mean_precision
0	MLPClassifier	0.0001	constant	0.001	sgd	0.642857	0.71352	0.77318	0.662666
1	MLPClassifier	1e-05	constant	0.001	sgd	0.642857	0.71352	0.77318	0.662666
2	MLPClassifier	0.1	constant	0.001	sgd	0.642857	0.71352	0.77318	0.662666
3	MLPClassifier	0.1	adaptive	0.001	sgd	0.642857	0.713104	0.771648	0.663049
11	MLPClassifier	0.0001	constant	0.0001	adam	0.642416	0.713293	0.77318	0.662245
10	MLPClassifier	0.1	adaptive	0.001	adam	0.642416	0.708745	0.755556	0.66843
9	MLPClassifier	0.0001	adaptive	0.001	sgd	0.642416	0.712853	0.771648	0.662629
8	MLPClassifier	1e-05	adaptive	0.001	sgd	0.642416	0.712853	0.771648	0.662629
7	MLPClassifier	0.1	constant	0.001	adam	0.642416	0.708745	0.755556	0.66843
6	MLPClassifier	1e-05	constant	0.0001	adam	0.642416	0.713293	0.77318	0.662245
5	MLPClassifier	1e-05	adaptive	0.0001	adam	0.642416	0.713293	0.77318	0.662245
4	MLPClassifier	0.0001	adaptive	0.0001	adam	0.642416	0.713293	0.77318	0.662245
12	MLPClassifier	0.1	constant	0.0001	adam	0.641975	0.712849	0.772414	0.662027
13	MLPClassifier	0.1	adaptive	0.0001	adam	0.641975	0.712849	0.772414	0.662027
14	MLPClassifier	1e-05	constant	0.001	adam	0.638448	0.705985	0.754023	0.664773
15	MLPClassifier	1e-05	adaptive	0.001	adam	0.638448	0.705985	0.754023	0.664773
19	MLPClassifier	1e-05	constant	0.1	adam	0.638007	0.702689	0.743295	0.666395
18	MLPClassifier	0.0001	adaptive	0.001	adam	0.638007	0.705309	0.75249	0.664822
17	MLPClassifier	0.0001	constant	0.001	adam	0.638007	0.705309	0.75249	0.664822
16	MLPClassifier	1e-05	adaptive	0.1	adam	0.638007	0.702689	0.743295	0.666395
20	MLPClassifier	0.0001	constant	0.1	adam	0.637125	0.718653	0.80613	0.649075
21	MLPClassifier	0.0001	adaptive	0.1	adam	0.637125	0.718653	0.80613	0.649075
22	MLPClassifier	1e-05	adaptive	0.0001	sgd	0.636684	0.715075	0.792337	0.651555
23	MLPClassifier	0.1	adaptive	0.0001	sgd	0.636684	0.715075	0.792337	0.651555
24	MLPClassifier	0.0001	adaptive	0.0001	sgd	0.636684	0.715075	0.792337	0.651555

Figure 6. MLP Classifier results from hyperparameter tuning

Lastly, the MLP classifier is tuned. With regards to initial learning rate, the optimal value is 0.001. The next decision to make is between ADAM or SGD. SGD with the constant learning rate of 0.001 delivered the best accuracy. The chosen alpha value is 0.0001.

On my Macbook Pro, both the Bagging Classifier and SVC took less than 15 seconds to finish. As expected, the MLP Classifier took the longest to fit at around 50 seconds.

```
Running GridSearchCV for MLPClassifier.
Fitting 3 folds for each of 36 candidates, totalling 108 fits
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent
workers.
[Parallel(n_jobs=3)]: Done  44 tasks      | elapsed:   24.8s
[Parallel(n_jobs=3)]: Done 108 out of 108 | elapsed:   50.8s finished
Running GridSearchCV for BaggingClassifier.
Fitting 3 folds for each of 24 candidates, totalling 72 fits
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent
workers.
[Parallel(n_jobs=3)]: Done  44 tasks      | elapsed:    4.5s
[Parallel(n_jobs=3)]: Done  72 out of  72 | elapsed:    7.5s finished
Running GridSearchCV for SVC.
Fitting 3 folds for each of 18 candidates, totalling 54 fits
[Parallel(n_jobs=3)]: Using backend LokyBackend with 3 concurrent
workers.
[Parallel(n_jobs=3)]: Done  54 out of  54 | elapsed:   14.1s finished
```

Figure 7. Hyperparameter tuning times on 2016 Macbook Pro

Comparing Algorithm Performance

In order to test the optimized classifiers, they were evaluated with unseen data. The metrics were produced via cross validation with three folds. The mean score and the 95% confidence interval of the score estimate is then displayed and evaluated. Also, a confusion matrix is plotted for each classifier to illustrate the metrics in terms of true positives/negatives and false positives/negatives. Using the following information, the

classifier that produces the best accuracy and f1 score combination should be chosen in order to maximize true positives/negatives and minimize false positives/negatives [11].

```
Accuracy (cross validation score): 0.676 (+/- 0.020)
f1 (cross validation score): 0.759 (+/- 0.019)
recall (cross validation score): 0.838 (+/- 0.038)
precision (cross validation score): 0.695 (+/- 0.011)
```

	Home	Away	Result	Date	Prediction
2603	Portland Trail Blazers	Miami Heat	0	02/05/2019	1
739	Chicago Bulls	Houston Rockets	0	03/10/2017	1
216	Phoenix Suns	Philadelphia 76ers	1	12/23/2016	1
2297	Portland Trail Blazers	Dallas Mavericks	1	12/23/2018	1
1707	Dallas Mavericks	Oklahoma City Thunder	0	02/28/2018	0
...	...	...	...	...	...
869	Denver Nuggets	New Orleans Pelicans	0	03/26/2017	1
1057	Phoenix Suns	New Orleans Pelicans	0	11/24/2017	0
2364	Charlotte Hornets	Dallas Mavericks	0	01/02/2019	1
839	Washington Wizards	Atlanta Hawks	1	03/22/2017	1
2453	Utah Jazz	Detroit Pistons	1	01/14/2019	1

756 rows × 5 columns

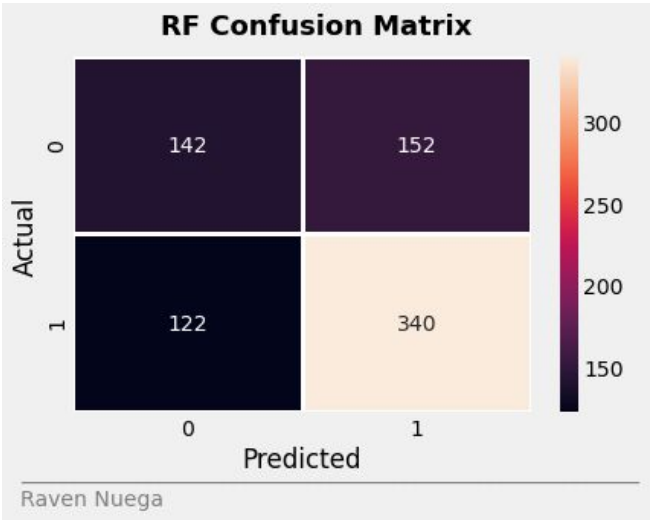

Figure 8. Random Forest with Bagging results and confusion matrix

```
Accuracy (cross validation score): 0.671 (+/- 0.019)
f1 (cross validation score): 0.764 (+/- 0.007)
recall (cross validation score): 0.874 (+/- 0.044)
precision (cross validation score): 0.680 (+/- 0.028)
```

	Home	Away	Result	Date	Prediction
2603	Portland Trail Blazers	Miami Heat	0	02/05/2019	1
739	Chicago Bulls	Houston Rockets	0	03/10/2017	0
216	Phoenix Suns	Philadelphia 76ers	1	12/23/2016	1
2297	Portland Trail Blazers	Dallas Mavericks	1	12/23/2018	1
1707	Dallas Mavericks	Oklahoma City Thunder	0	02/28/2018	1
...	...	...	...	...	...
869	Denver Nuggets	New Orleans Pelicans	0	03/26/2017	1
1057	Phoenix Suns	New Orleans Pelicans	0	11/24/2017	0
2364	Charlotte Hornets	Dallas Mavericks	0	01/02/2019	1
839	Washington Wizards	Atlanta Hawks	1	03/22/2017	1
2453	Utah Jazz	Detroit Pistons	1	01/14/2019	1

756 rows × 5 columns

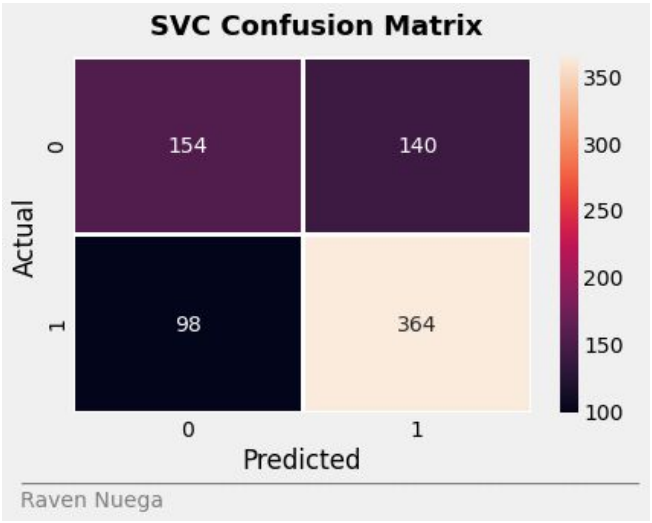

Figure 9. Support Vector Classifier results and confusion matrix

```
Accuracy (cross validation score): 0.677 (+/- 0.013)
f1 (cross validation score): 0.756 (+/- 0.008)
recall (cross validation score): 0.816 (+/- 0.016)
precision (cross validation score): 0.703 (+/- 0.015)
```

	Home	Away	Result	Date	Prediction
2603	Portland Trail Blazers	Miami Heat	0	02/05/2019	1
739	Chicago Bulls	Houston Rockets	0	03/10/2017	0
216	Phoenix Suns	Philadelphia 76ers	1	12/23/2016	1
2297	Portland Trail Blazers	Dallas Mavericks	1	12/23/2018	1
1707	Dallas Mavericks	Oklahoma City Thunder	0	02/28/2018	0
...	...	...	...	...	...
869	Denver Nuggets	New Orleans Pelicans	0	03/26/2017	1
1057	Phoenix Suns	New Orleans Pelicans	0	11/24/2017	0
2364	Charlotte Hornets	Dallas Mavericks	0	01/02/2019	1
839	Washington Wizards	Atlanta Hawks	1	03/22/2017	1
2453	Utah Jazz	Detroit Pistons	1	01/14/2019	1

756 rows × 5 columns

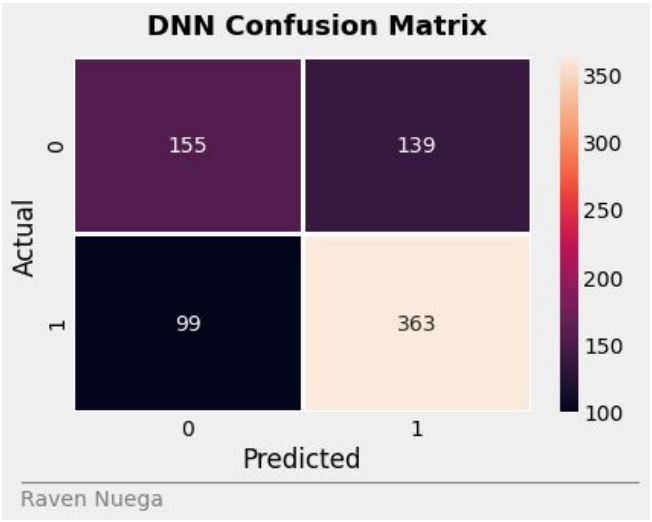

Figure 10. Multi-layer Perceptron Classifier results and confusion matrix

In terms of accuracy, all classifiers performed reasonably well on the test data. Each classifier's test accuracy score is greater than their best training scores due to quality generalization. In terms of accuracy, Random Forest and the Deep Neural Network performed best, which is expected since they classify non-linear data better than Support Vector machines. While DNN has higher mean accuracy, its confidence interval is smaller than Random Forest, which means the latter has higher max accuracy.

With regards to f1 scores, each classifier outperformed its best f1 training score, which is also a sign of good generalization. SVC minimizes false negatives best with a high recall score, while DNN has the highest precision minimizing false positives. However, the highest mean f1 scores consisted of SVC and Random Forest.

The real decision factor is the confidence intervals for both accuracy and f1 score. Both SVC and DNN have similar intervals in f1 score, but DNN has a smaller confidence in accuracy. Random Forest with Bagging has the largest confidence intervals for both metrics. Despite its larger confidence intervals, the mean scores of RF were never the smallest of the group. In conclusion, Random Forest with bagging is the best classifier for this dataset due to its relatively high accuracy and f1 scores.

Conclusion

According to quantitative analysis, Random Forest with bagging is the best classifier. Qualitatively, Random Forest is still the strongest option. Firstly, they are able to classify non-trivial datasets accurately. For instance, Random Forests can handle nonlinear, imbalanced, and/or non-numeric datasets [4]. In NBA analytics, data is rarely organized or linearly separable, making RF a strong choice. In addition, Random Forests are fast and require only a few hyperparameters to be tuned. The only downside is overfitting, yet there are plenty of solutions such as bagging or boosting. Overall, the combination of speed, performance and dataset versatility makes Random Forests a top performer in the field of basketball data analytics and machine learning.

Acknowledgements

[1] NBA. "NBA Advanced Team Stats." stats.nba.com. https://stats.nba.com/teams/traditional/?sort=W_PCT&dir=-1&Season=2018-19&SeasonType=Regular%20Season(April 18, 2020).

[2] Jake Kandell. "NBA-Predict." Github. https://github.com/JakeKandell/NBA-Predict/tree/92dc7593996c710ed7af8fe0596f00dfb5dfdc12(April 17, 2020).

[3] Mohammed Sunasra. "Performance Metrics for Classification Problems in Machine Learning." Medium. https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b(April 17, 2020).

[4] Brian Hrolenok. "Decision Trees." Canvas. https://gatech.instructure.com/courses/97134/files/folder/Lecture%20Slides/Week%203?preview=10288861(April 16, 2020).

[5] Scikit-learn. "Bagging Classifier." scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.BaggingClassifier.html(April 16, 2020).

[6] Brian Hrolenok. “Support Vector Machines.” Canvas. https://gatech.instructure.com/courses/97134/files/folder/Lecture%20Slides/Week%204?preview=10624007(April 16, 2020).

[7] Scikit-learn. “SVC” scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html(April 16, 2020).

[8] Scikit-learn. “MLPClassifier” scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html(April 16, 2020).

[9] Max Zuo, private communication, April 16, 2020.

[10] Kiyoshi Kawaguchi. “Linear Separability and the XOR problem” ece-utep. http://www.ece.utep.edu/research/webfuzzy/docs/kk-thesis/kk-thesis-html/node19.html(April 17, 2020).

[11] Purva Huilgol. “Accuracy vs F-1 score” Medium. https://medium.com/analytics-vidhya/accuracy-vs-f1-score-6258237beca2(April 18, 2020).