Machine Learning: Classification algorithms

Practical application 1

Juan Luis Ruiz-Tagle

December 16, 2019

Contents

1 Introduction

In the NBA league there exists very high competition amongst players. There is a lot of money in the game and only the very best get to play on this league. Even fewer manage to stay around for a significant amount of time. There is usually a younger, faster, stronger and more skilled player ready to substitute those who do not make the grade. Therefore, there is a high degree of rotation in this league. The NBA Rookie dataset contains information about **1328** players, and whether or not they managed to survive more than 5 years in the NBA. **21 features** are provided for each player.

2 Problem description

The main goal of this assignment is to experiment with different machine learning classification methods and study the advantages and drawbacks of each of them on the classification task. We will start testing supervised machine learning methods and then take a look at some unsupervised ones. As well we will use univariate and multivariate feature selection, with filtering and wrapping, to check if the selected metrics improve.

2.1 Dataset description

Our data contains 21 features, which are the following:

- **Name**: Name of the player
- **GP**: Number of games played
- **MIN**: Number of minutes played
- **PTS**: Average number of points scored per game played
- **FGM**: Number of field goals made.
- **FGA**: Number of field goal attempts
- **FG%**: Percentage of field goal attempts scored.
- **3P Made**: Number of 3-point goals scored.
- **3PA**: Number of 3-point goals attempted.
- **3P%**: Percentage of 3-point goal attempts scored
- **FTM**: Number of free throws scored.
- **FTA**: Number of free throws attempted.
- **FT%**: Percentage of free throws goal attempts scored.
- **OREB**: Number of offensive rebounds.

- **DREB**: Number of defensive rebounds.
- **REB**: Number of 3-points scored.
- **AST**: Number of assists.
- **STL**: Number of steals.
- **BLK**: Number of blocks.
- **TOV**: Number of turnovers.
- **TARGET_5Yrs**: Outcome. 1 if career length ¿= 5 years. 0 otherwise

As we can see, there are some columns which are collinear with others. Specifically $FG\% = 100\frac{\dot{FGM}}{FGA}$, $3P\% = 100\frac{3P\dot{M}ade}{3PA}$, $FT\% = 100\frac{F\dot{T}M}{FTA}$, $REB = OREB + DREB$ This issue will be addressed later on when we use filtering.

2.2 Data cleaning

Before we start our analysis, we have to be sure that our data is properly treated and cleansed. By looking at it, we find out some few **missing values** in the 3P feature. We could drop them but we since we already know the relationship between this feature and *3P% Made* and *3PA* we fill them by computing the percentage between these features for those cases. We also **check for duplicates** (12 found) and drop them. As well, we **normalize** the dataset since some of the algorithms will perform better with the data normalized. At last, we split the dataset in **training and test sets**, with a 70% - 30% ratio respectively.

3 Methodology

In this work we will be conducting exploratory research in order to compare different algorithms in the task of binary classification. We don't have a research question, we will just discuss how these algorithms are performing in each case, and if they improve with feature selection methods. The tools we will be using are mainly python altogether with some standard well known libraries (scikit-learn[1] for machine learning algorithms, numpy[2] for numerical vector manipulation and pandas[3] for working with the data as dataframes) and a pair of smaller packages to use in special cases. The code used to complete this assignment is available on Github.

3.1 Metrics

In order to measure the performance of the algorithms we will select five metrics[4]:

- **Accuracy**: Accuracy is a valid choice of evaluation for classification problems which are well balanced and not skewed. In our dataset, the classes are balanced as 60% and 40%. (TP+TN)/(TP+FP+FN+TN)

- **Precision**: It is a useful metric when we want to be very sure of our prediction. (TP)/(TP+FP)
- **Recall**: This metric can be interesting to see how many players which actually stay more than 5 years are we able to predict correctly (TP)/(FN + TP)
- **F1 score**: We want it to have a balance between precision and recall. Probably this will be our most insightful metric.
- **Binary Crossentropy (log-loss)** : Log Loss takes into account the uncertainty of your prediction based on how much it varies from the actual label. This gives us a more nuanced view of the performance of our model. Minimizing Log Loss gives greater accuracy for the classifier.

There are many other metrics which we have not considered since they are not useful for analysing classification predictions.

4 Supervised Learning: all features

4.1 Results

The first task consist in applying a set of algorithms, both probabilistic and non-probabilistic, using all the features available at our disposal. A total of 11 algorithms have been trained with all the features in our dataset and tested their performance. Figure 1 is a table with the results of each algorithm for each metric. Instead of discussing each algorithm individually, I decided to compare them to each other to bring to light their differences in the following section. The upcoming tables that will show up in this report will be formatted to ease readability. Each column will be green-scaled depending on the values in its cells, except for the one with highest value, whose background will be red. Negative values will be printed with a red font as well.

4.2 Discussion

Several interesting points arise when looking closely at Figure 1:

4.2.1 Our winner and our loser

First of all, we quickly identify **Logistic Regression** as the best performing algorithm, since it has highest f1-score and accuracy. This algorithm performs very well since we are in a binary classification task. Should be noted that it has the lowest binary cross-entropy (log loss), and this is the cost function which it is trying to minimize. In the other hand **Naive Bayes** (GaussianNB) which is as well a probabilistic classifier, is the worst of them all. The reason of this is the aforementioned multicollinearity across many of features. The Naive Bayes assumption is precisely that all the variables are conditionally independent, which is not the case, so it performs very bad.

	f1_score	accuracy_score	precision_score	recall_score	log_loss
KNeighborsClassifier	73.59%	65.83%	72.82%	74.38%	11.8015
MLPClassifier	80.95%	74.49%	77.52%	84.70%	8.81184
SVC	80.41%	74.03%	77.74%	83.27%	8.96919
DecisionTreeClassifier	69.95%	62.41%	71.64%	68.33%	12.9817
RandomForestClassifier	76.92%	69.93%	75.60%	78.29%	10.3854
AdaBoostClassifier	79.39%	72.44%	76.14%	82.92%	9.51993
LogisticRegression	81.56%	75.17%	77.74%	85.77%	8.57581
GaussianNB	62.64%	61.96%	84.34%	49.82%	13.1389
LinearDiscriminantAnalysis	81.54%	74.94%	77.14%	86.48%	8.65449
QuadraticDiscriminantAnalysis	67.09%	64.46%	82.38%	56.58%	12.2735
RIPPER	64.22%	65.06%	92.42%	50.40%	14.0588

Figure 1: Performance metrics with all features included

4.2.2 Discriminant Analysis

The second best performing algorithm is **LDA**, with a very high recall and a good balance between precision and recall. Log-loss is as well very low for LDA. However, its "cousin" **QDA** has a much worse performance, since it is trying to find quadratic relationships were there are none. We can see that it has a high precision but a very low recall.

4.2.3 The trees

The three tree algorithms are placed one after another on purpose on the table, to show how each metric is improving as we add ensemble techniques. The single **DecisionTree** is not performing specially well, probably affected by the multicollinearity problems. It uses CART algorithm, very similar to C4.5 (CART does not use rule sets to decide where to split the data). But we see that putting many of these working together in a **RandomForest** make accuracy and f1_score increase by 7 points. **AdaBoost** does it even better, getting all the way up to 72.44% accuracy and 79% f1_score. Binary cross-entropy diminishes progressively as well.

4.2.4 The rest

SVC and **MLP** both perform very well, with incredibly similar scores in all the metrics. I cannot come up with a suitable explanation of why is this the case. The SVC is using an RBF kernel and the MLP a relu activation function with a hidden layer of 100 neurons. **KNN** is set to search for 5 nearest neighbors with a uniform euclidean distance (minkowski = 2). It has a good f1-score, but

log-loss is very high. To finish, the **RIPPER** algorithm, which has taken an extremely large amount of time to train compared to the others, performs very bad. It has a very high precision which means nothing since the recall is very low. Log-loss is the highest of all the algorithms.

5 Univariate feature selection

5.1 Results

The second task at hand consisted in applying univariate feature selection to each of the algorithms, to see if the performance metrics actually improved. We selected the 5 best features according to the chi squared χ^2 score function. This function measures the divergence from the distribution expected if one assumes that feature occurrence is independent of the class [5]. 'GP', 'MIN', 'PTS', 'FGA' and 'REB' turned out to be, in this order, the most important features. But this method is only checking them individually, so it is not perfect.

	f1_score	accuracy_score	precision_score	recall_score	log_loss
KNeighborsClassifier	76.09%	67.65%	72.20%	80.43%	11.1722
MLPClassifier	80.41%	74.03%	77.74%	83.27%	8.96919
SVC	80.48%	74.26%	78.19%	82.92%	8.89051
DecisionTreeClassifier	72.27%	64.69%	72.66%	71.89%	12.1949
RandomForestClassifier	74.96%	67.88%	74.82%	75.09%	11.0934
AdaBoostClassifier	78.97%	72.21%	76.59%	81.49%	9.5986
LogisticRegression	80.89%	74.49%	77.70%	84.34%	8.81184
GaussianNB	69.60%	66.97%	84.69%	59.07%	11.4081
LinearDiscriminantAnalysis	81.28%	74.72%	77.24%	85.77%	8.73317
QuadraticDiscriminantAnalysis	72.62%	68.56%	82.06%	65.12%	10.8574
RIPPER	64.23%	63.45%	86.88%	44.17%	14.0315

Figure 2: Performance metrics with only 'GP', 'MIN', 'PTS', 'FGA' and 'REB' features included

5.2 Discussion

We can perfectly see how the selected features are not directly related to each other in the formulae we presented in the dataset description. Figure presents a table with the metrics of the algorithms using only this subset of features. Apparently, the number of played games is the most decisive feature of all. This makes sense: the better you are, the more you get to play. The number of minutes is as well very relevant and it has certain connection to the number of

games played, but they are not completely correlated. For example, a player could play in many games but only 5 minutes in each because he deceives the coach over and over again. With these subset of features we have a simpler model. However, the metrics haven't got much better. To understand this we can take a look at Figure 3. In it we can see how most of the classifiers stay around the same values. In general, The notable exceptions are Naive Bayes and QDA, which experience an important improvement in almost all the metrics. Naive Bayes increases f1-score by 7%, accuracy by 5% and recall by 9%. Log-loss decreases by 1.7 points. This might be explained by the removal of correlated variables.

	f1_score	accuracy_score	precision_score	recall_score	log_loss
KNeighborsClassifier	2.50%	1.82%	-0.62%	6.05%	-0.629392
MLPClassifier	-0.54%	-0.46%	0.22%	-1.42%	0.157348
SVC	0.07%	0.23%	0.45%	-0.36%	-0.0786797
DecisionTreeClassifier	2.33%	2.28%	1.02%	3.56%	-0.78676
RandomForestClassifier	-1.97%	-2.05%	-0.78%	-3.20%	0.708084
AdaBoostClassifier	-0.42%	-0.23%	0.44%	-1.42%	0.0786706
LogisticRegression	-0.67%	-0.68%	-0.04%	-1.42%	0.236026
GaussianNB	6.96%	5.01%	0.36%	9.25%	-1.73087
LinearDiscriminantAnalysis	-0.26%	-0.23%	0.10%	-0.71%	0.0786742
QuadraticDiscriminantAnalysis	5.53%	4.10%	-0.32%	8.54%	-1.41616
RIPPER	0.01%	-1.61%	-5.53%	-6.23%	-0.0273003

Figure 3: Difference in performance using univariate filter vs using all features $(univariate - df)$

6 Multivariate feature selection

The next step is to investigate a the set of techniques which go beyond univariate selection, that is the ones which consider the effect of removing a subset of variables at the same time.

6.1 Filtering

6.1.1 Results

First we will consider filtering. There are several posible techniques, but the one chosen here is **Correlation-based Feature Selection** (CFS in short from now on). It evaluates subsets of features on the basis of the following hypothesis: "Good feature subsets contain features highly correlated with the classification,

yet uncorrelated to each other".[6] After executing the algorithm we obtain the following subset of features to use in our training process: FT%', 'MIN', '3P%', 'FG%', 'GP' and 'BLK'. We train all our models exclusively with this features and we obtain the following results, which can be seen in Figure 4. We can see as well the difference between this table and the one obtained training with all the features in Figure 5. Note how CFS has decided on its own to include 6 features instead of 5.

	f1_score	accuracy_score	precision_score	recall_score	log_loss
KNeighborsClassifier	77.57%	70.62%	75.85%	79.36%	10.1493
MLPClassifier	81.48%	74.94%	77.32%	86.12%	8.65449
SVC	80.80%	73.80%	76.10%	86.12%	9.04788
DecisionTreeClassifier	68.35%	60.14%	69.49%	67.26%	13.7685
RandomForestClassifier	74.78%	67.88%	75.18%	74.38%	11.0934
AdaBoostClassifier	78.92%	71.53%	75.00%	83.27%	9.83465
LogisticRegression	80.74%	74.03%	76.85%	85.05%	8.9692
GaussianNB	76.32%	71.30%	80.88%	72.24%	9.91327
LinearDiscriminantAnalysis	81.08%	74.49%	77.17%	85.41%	8.81184
QuadraticDiscriminantAnalysis	77.80%	71.53%	77.66%	77.94%	9.83462
RIPPER	58.19%	63.70%	91.05%	45.87%	14.0069

Figure 4: Performance metrics with only 'FT%', 'MIN', '3P%', 'FG%', 'GP' and 'BLK' features included

6.1.2 Discussion

After a quick glance at Figure 5 we rapidly realise that the multivariate feature selection is more powerful than its univariate counterpart. Once again, the subset of features selected by CFS are not dependent on each other in an intuitive way. It is worth mentioning how CFS has realised the importance of the ratios 'FT%','3P%', 'FG%'. It makes a lot of sense that the ratio *score/attempts* of the different types of throws has greater impact on the target variable than the very number of times a player has actually scored. If you get to play more, you probably score more points, but you are not necessarily better since you had many more chances to score. Regarding the algorithms, we can see how both **Naive Bayes** and **QDA** benefit once again of the appropriate feature selection. Naive Bayes improves a 14% in f1-score and an insane 22% in recall. QDA sees an 11% and 21% raise in those metrics. We have to point out that even if they have become much better, they still perform worse than most of the other algorithms. The **Random Forest Classifier** came off badly from the changes in the feature selection model. It drops f1-score by 4.1% and accuracy and recall

	f1_score	accuracy_score	precision_score	recall_score	log_loss
KNeighborsClassifier	3.77%	5.24%	4.28%	3.20%	-1.80957
MLPClassifier	6.42%	4.10%	-5.62%	17.44%	-1.41611
SVC	2.43%	7.74%	9.92%	-9.96%	-2.6751
DecisionTreeClassifier	1.06%	0.46%	-0.53%	2.49%	-0.157343
RandomForestClassifier	-4.16%	-4.78%	-3.23%	-4.98%	1.65221
AdaBoostClassifier	-0.47%	-0.91%	-1.14%	0.36%	0.314713
LogisticRegression	-0.55%	-0.91%	-1.00%	0.00%	0.314711
GaussianNB	13.68%	9.34%	-3.46%	22.42%	-3.22568
LinearDiscriminantAnalysis	-0.46%	-0.46%	0.03%	-1.07%	0.15735
QuadraticDiscriminantAnalysis	10.71%	7.06%	-4.72%	21.35%	-2.4389
RIPPER	-1.01%	0.55%	-2.06%	-3.56%	0.0134361

Figure 5: Difference in performance using multivariate filter vs using all features ($multivariate - df$)

by almost 5% for no apparent reason.

6.2 Wrapping

6.2.1 Results

Wrapper methods evaluate each possible subset of features with a criterion consisting of the estimated performance of the classifier built with this subset of features[7]. This technique basically takes into account the model architecture to select a subset of features. I selected **Sequential Feature Selector** (SFS) algorithm as a wrapper method. It was in its forward flavour and configured without floating. The subset of features that SFS selected for each algorithm are shown on Figure 6. After training each algorithm with its corresponding subset of features we get the results seen on Figure 7. The comparison between the multivariate filter and the wrapping method can be seen in Figure 8

6.2.2 Discussion

Several points can be discussed regarding the SFS wrapper. To begin with, the feature subsets selected are very diverse, but all limit themselves to 13 variables (of the original 21). The most popular features were 'GP', '3P Made' and '3P%', used by 7 algorithms each. I was surprised (and disappointed) to see that some algorithms selected too similar features. For example, **KNN** selects both '3P Made' and '3PA', which are highly correlated. If we look at Figure 8, we notice that the wrapper method generally performs worse than the multivariate filter. This was unexpected to me, since wrapper methods have that extra piece of

	GP	FGM	3P Made	3PA	BLK	FG%	3P%	AST	FT%	REB	TOV	OREB	STL
KNeighborsClassifier	x	x		x	x	x							
MLPClassifier				x	x		x	x	x				
SVC	x					x	x		x	x			
DecisionTreeClassifier		x		x				x			x	x	
RandomForestClassifier	x			x							x	x	x	
AdaBoostClassifier	x							x		x			x	x
LogisticRegression	x				x		x	x		x				
GaussianNB	x			x			x	x		x				
LinearDiscriminantAnalysis				x	x		x		x			x		
QuadraticDiscriminantAnalysis	x			x			x	x		x				
RIPPER		x			x		x	x				x		

Figure 6: Features selected by SFS wrapper to train on each algorithm

	f1_score	accuracy_score	precision_score	recall_score	log_loss
KNeighborsClassifier	73.63%	64.92%	70.96%	76.51%	12.1163
MLPClassifier	80.84%	72.89%	73.82%	89.32%	9.36261
SVC	80.94%	74.03%	76.34%	86.12%	8.9692
DecisionTreeClassifier	69.33%	61.50%	70.74%	67.97%	13.2964
RandomForestClassifier	72.27%	64.69%	72.66%	71.89%	12.1949
AdaBoostClassifier	77.35%	70.39%	75.77%	79.00%	10.228
LogisticRegression	80.46%	73.12%	75.23%	86.48%	9.28392
GaussianNB	79.32%	72.44%	76.32%	82.56%	9.51993
LinearDiscriminantAnalysis	80.91%	73.12%	74.18%	88.97%	9.28393
QuadraticDiscriminantAnalysis	79.60%	72.44%	75.64%	83.99%	9.51994
RIPPER	61.66%	62.98%	90.51%	46.51%	14.0341

Figure 7: Performance metrics with features selected by SFS wrapper

information of which algorithm is going to be used. Only **RIPPER** (what a surprise) **Naive Bayes** and **QDA**, once again, seem to slightly benefit from this wrapper method. Nevertheless, we appreciate a substantial improvement of all models when compared to the basic model fitting with all the features included. The winner of this test is the **SVC**, scoring the highest f1-score and accuracy values (81% and 74%).

	f1_score	accuracy_score	precision_score	recall_score	log_loss
KNeighborsClassifier	-3.94%	-5.69%	-4.89%	-2.85%	1.96693
MLPClassifier	-0.64%	-2.05%	-3.49%	3.20%	0.708117
SVC	0.14%	0.23%	0.24%	0.00%	-0.0786778
DecisionTreeClassifier	0.97%	1.37%	1.26%	0.71%	-0.472063
RandomForestClassifier	-2.50%	-3.19%	-2.52%	-2.49%	1.10148
AdaBoostClassifier	-1.57%	-1.14%	0.77%	-4.27%	0.393367
LogisticRegression	-0.28%	-0.91%	-1.62%	1.42%	0.314719
GaussianNB	3.00%	1.14%	-4.56%	10.32%	-0.393336
LinearDiscriminantAnalysis	-0.17%	-1.37%	-2.99%	3.56%	0.472085
QuadraticDiscriminantAnalysis	1.80%	0.91%	-2.02%	6.05%	-0.31468
RIPPER	3.47%	-0.73%	-0.54%	0.64%	0.0271844

Figure 8: Difference in performance using SFS s wrapper vs multivariate filter ($wrapper - multivariate$)

7 Unsupervised learning

Finally we arrive to the third part of our exploratory analysis. In this section we will test different unsupervised learning algorithms (no labels included), one of each of the three existing kinds.

7.1 Results

7.1.1 Hierarchical clustering

Hierarchical clustering algorithms are based on the core idea of objects being more related to nearby objects than to objects farther away [8]. These algorithms connect "objects" to form "clusters" based on their distance. The algorithm that we have used is **Agglomerative Clustering**, set up with an euclidan affinity and a "ward" linkage. Of course, it was configured to find only 2 clusters.

7.1.2 Partitional clustering

The idea behind partitional clustering is to divide the objects into k partitions or groups[9]. The number K must be known a priori, and in our case is set to 2. I used **Kmeans**, the most typical unsupervised algorithm in this category, set up with euclidean distance.

7.1.3 Probabilistic clustering

This third kind of clustering is based on distribution models. Clusters can then easily be defined as objects belonging most likely to the same distribution[10]. A convenient property of this approach is that this closely resembles the way artificial data sets are generated: by sampling random objects from a distribution. Our selected algorithm is **Gaussian Expectation Maximisation**, configured with a diagonal covariance type (found out by our algorithm) and 2 components.

The performance metrics of these three algorithms can be seen in Figure 9

7.2 Discussion

The metrics of the unsupervised algorithms are not great. Kmeans is the best without a doubt, with the highest f1-score and an accuracy of 70%. Agglomerative Clustering has the highest precision (93%) but very low recall. This means that the algorithm has clustered almost all the entries together, failing to distinguish the two classes. Binary cross entropy is high on the three algorithms, meaning that the clusters are not very well defined.

	f1_score	accuracy_score	precision_score	recall_score	log_loss
AgglomerativeClustering	26.83%	45.33%	93.62%	15.66%	18.8823
Kmeans	75.05%	70.16%	80.74%	70.11%	10.3066
Gaussian_em	73.18%	58.09%	61.98%	89.32%	14.4767

Figure 9: Performance metrics of unsupervised learning algorithms.

8 Conclusions

After careful consideration, we can draw a set of conclusions from this exploratory analysis, being the first that feature selection is a fundamental process in any data science project. For some algorithms, having redundant information is not a problem, but this is not the general case. What it is clear is that quickly pipelining all the data into the first algorithm that comes to our mind leads nowhere, but to error. Instead, understanding our data and the relationships across its variables can provide useful insights to our study. Of course, it makes sense to use statistical techniques for selecting features, in order to avoid

human bias in what we might think suits best to solve a problem. But one thing doesn't rule out the other, we need to understand our data and let statistics select the important features.

A second conclusion is that Multivariate feature selection performs much better than Univariate feature selection. If we have the computational capabilities, we should lean towards the multivariate alternative since in most of the cases variables have hidden relationships to each other which should not be left out. In our case Correlation-based Feature Selection performed much better than Sequential Forward Selector, but both of them clearly surpassed the Univariate filter.

I can only add that supervised classification is far more precise than the unsupervised, at least in our tests. When possible, the former should be preferred over the latter.

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[2] T. Oliphant, "NumPy: A guide to NumPy." USA: Trelgol Publishing, 2006–. [Online; accessed ¡today¿].

[3] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference* (S. van der Walt and J. Millman, eds.), pp. 51 – 56, 2010.

[4] R. Agarwal, "The 5 classification evaluation metrics every data scientist must know," September 2019. [Online; posted 17-September-2019].

[5] J. Brownlee, *Statistical Methods for Machine Learning: Discover how to Transform Data into Knowledge with Python*. Machine Learning Mastery, 2018.

[6] M. A. Hall, *Correlation-based Feature Selection for Machine Learning*. PhD thesis, 1999.

[7] T. M. Phuong, Z. Lin, and R. B. Altman, "Choosing snps using feature selection," *Journal of bioinformatics and computational biology*, vol. 4, no. 02, pp. 241–257, 2006.

[8] L. Rokach and O. Maimon, "Clustering methods," in *Data mining and knowledge discovery handbook*, pp. 321–352, Springer, 2005.

[9] D. Pelleg and A. Moore, "Accelerating exact k-means algorithms with geometric reasoning (technical report cmu-cs-00105)," 1999.

[10] C. J. Wu *et al.*, "On the convergence properties of the em algorithm," *The Annals of statistics*, vol. 11, no. 1, pp. 95–103, 1983.