NBA Challenge

Margot Barbet
Victor Sanh

1 Introduction

The goal of this challenge is to be able to predict the winning team of a NBA game, based on eleven different features describing the game from the beginning to the end of the second quarter, i.e. from the beginning to half-time. Thus, this is a classification challenge: for each game the prediction is either 1 if the invited team wins and 0 otherwise.

To realize this challenge, we have in hand play-by-play datasets from the last 15 NBA seasons which represents 12,576 NBA games in our training dataset and 4,192 games in the testing dataset. For each game, we have 11 variables describing the difference between the two teams on those 11 features for every second from the beginning to half-time, ie during 1440 seconds. Those 11 variables are: score, offensive rebound and defensive rebound (describing players that retrieve the ball), offensive foul and defensive foul, assist (describing players that pass the ball to their teammates in a way that leads to a score), lost ball, bad pass, block, steals (describing players that take possession of the ball while the opposing team had it) and miss.

Therefore, the goal of our work in this challenge is to propose a model in order to predict the winning team of a NBA game.

2 Feature Engineering

The first step of any prediction work is to analyze features and create others in order to make machine learning algorithms work. Thus, we first looked at the correlations between the different features and the winning team in order to identify the most important features.

Figure 1 represents the accuracy obtained by using only one variable to predict the winning team with respect to each second of the game. As we may have expected the score but also the number of assists and rebounds seem to have a significant and positive influence on the winning team. Using the winning team at half-time (i.e. score at half-time) leads to 72% submission accuracy. On the opposite, the number of missed shots seems to have an important negative influence on the winning team, meaning that the higher the number of missed shots, the higher the probability of the other team to win. Finally, the number of defensive fouls alone does not seem to have any impact on the winning team. Therefore, we have decided to exploit those features in order to create new ones that will be better for prediction. By searching for the most common basketball statistics that best describe the match, we chose to compute seven more features:

- Successful free throws: free throws are unopposed shots from behind the free throw line and worth one point

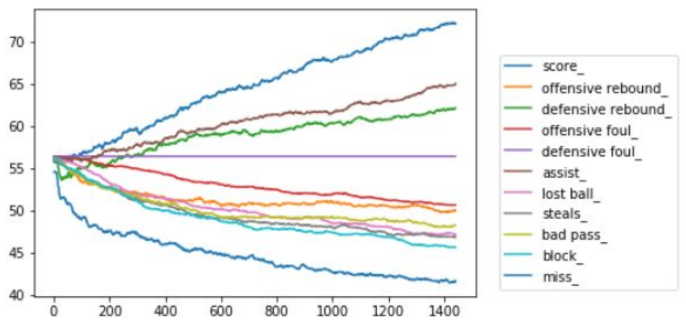


Figure 1: Use of one variable to predict the winner of the game

- Three points: successful shots made from beyond the three-point line and worth 3 points
- Two points (optional): successful shots from within the three-point line, it is worth 2 points
- Field goal: any successful shot other than free throws, it is worth 2 or 3 points
- Total rebound: sum of defensive and offensive rebound
- Total foul: sum of defensive and offensive foul
- Diff points: the point difference from one second to the next one

Figure 2 describes the correlation between features, including our new features and the winning team at the end of the second quarter. As we notice, our new features, and especially field goals, seem to be highly correlated with the winning team and therefore may help improving the prediction.

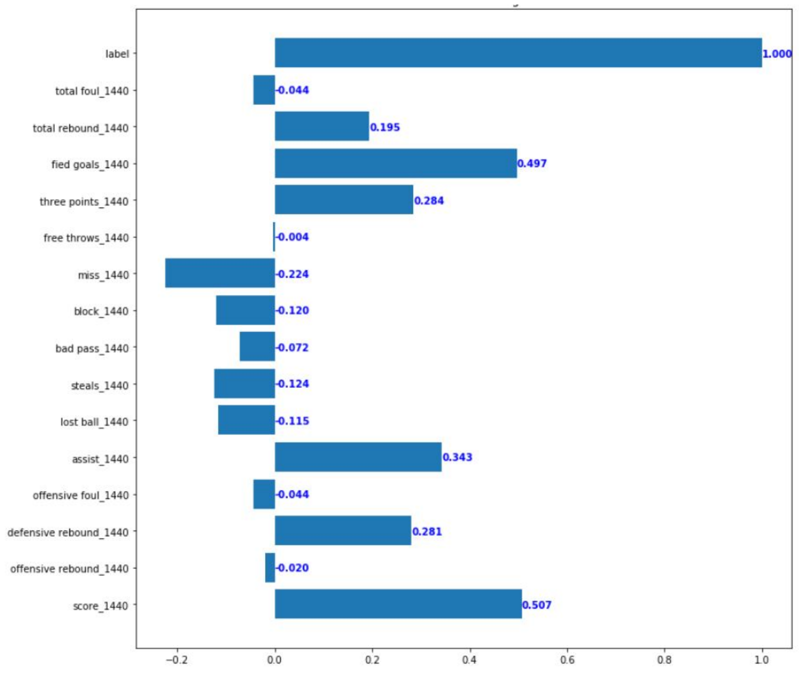


Figure 2: Correlations of features at half time with winning team

3 Some classic classifiers

We first decided to study some classic machine learning classifiers widely used for binary classification such as a logistic regression, a random forest classifier and a gradient boosting classifier.

3.1 Logistic Regression

The logistic regression is especially adapted to binary classification problems since it is based on the sigmoid function. Indeed, in this model, we assume that for a given observation x, the output y follows a Bernoulli law with parameter $\theta = \sigma(w^T x)$ where w is a weighting vector and σ the sigmoid function. We then are able to learn the weighting vector w using *Newton's optimization method* (for instance).

By tuning the regularization parameter, we obtained the highest validation accuracy for $C = 1e{-}5$, and thus, we chose to keep this parameter.

Then, we used 5-folds validation in order to evaluate our model. We were able to found a validation accuracy of 72.15% +/- 1.62% while we found a training accuracy of 72.44% +/- 0.41%.

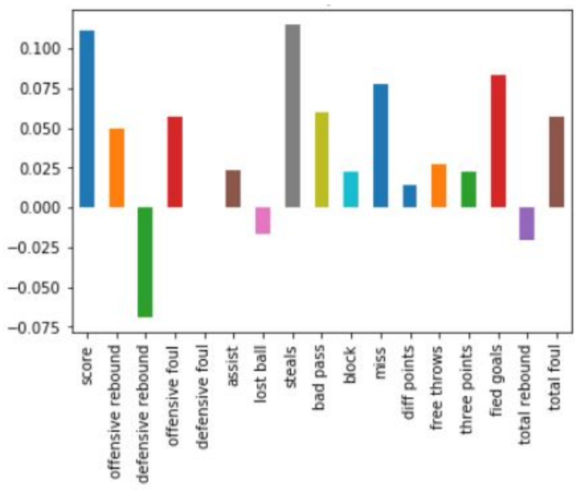


Figure 3: Coefficients of the Logistic Regression

We focused on the main features used by the classifier by analyzing the coefficient of the logistic regression. To do that, for each of the 17 features, we aggregated each of its 1440 coefficients (there are 1440 points for each feature). We used the same method for the other classifiers throughout the rest of our work.

As we can see on figure 3 both score and steals have a large influence on the prediction. The coefficients' signs also bring some interesting information. Indeed, while features such as score or field goals are used in a positive way in order to predict the winning team, features such as lost ball are more characteristic of the defeat of the team. Besides, we found that the number of defensive fouls is not identified by the model as being important for prediction. However, we are not able to explain the signs of some features' coefficients such as the one of "miss". Indeed, intuitively, we expect that the number of unsuccessful shots has a negative influence on the team's victory and we are then unable to explain the positive sign of its coefficient. In the same way, we are surprised by the negative coefficients of variables "offensive rebound" and "total rebound".

3.2 Random Forest Classifier

Random Forests are widely used in machine learning and in classification. This model aims to construct a multitude of decision trees, a forest of trees, during the training and output either the mean of the predictions of all trees or the class that appears most often among all trees.

In order to obtain the highest validation accuracy, we tuned all parameters. Since we have a lot of data, we realized that we needed a lot more estimators than the default parameters. Indeed, we found that we need at least 200 estimators in order to have good predictions, which represents 200 trees in the forest.

We then also evaluated this classifier using 5-folds cross validation. It leads to a validation accuracy of 73.84% +/- 1.25% and a training accuracy of 96.90% +/- 0.11%, showing that the models is still overfitting the training set. By submitting this model, we obtained a testing accuracy of 73.77%. Besides, by looking at how trees make their decisions, we are able to sort the variables in order of importance and therefore to evaluate our features. Figure 4 displays the importance of features. The higher the score, the more important the variable is in the prediction. As before, for each feature, we aggregated the score of its 1440 score (meaning that we summed them). Thus, we notice that *score* and *fields goals* stand out clearly as the main important features showing that the algorithm relies heavily on these features to make its predictions. Although this result seems intuitive, indeed the more the team scores, the more likely this team is to win, it reinforces our choice of variables and then reinforces our feature engineering study. In contrast, some features seem to have little or no influence on prediction: the number of defensive fouls or the difference of points for instance.

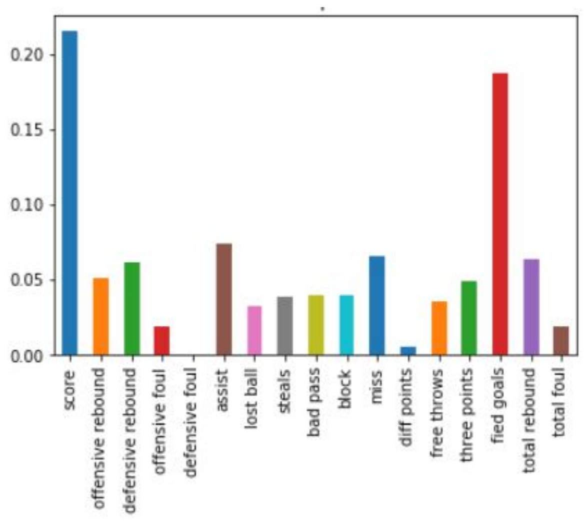


Figure 4: Feature importance in Random Forest model

3.3 Gradient Boosting Classifier

Gradient Boosting is another technique widely used in machine learning, for both classification and regression problems. The boosting technique consists in aggregating several models to obtain a single final result. The different models are built sequentially and the boosting model attributes a weight for each prediction by evaluating them. The goal is to assign a larger weight to difficult predictions so that the model focuses on these predictions on the next round. The specificity of the gradient boosting is that it uses the gradient of the loss function for calculating weights when building models.

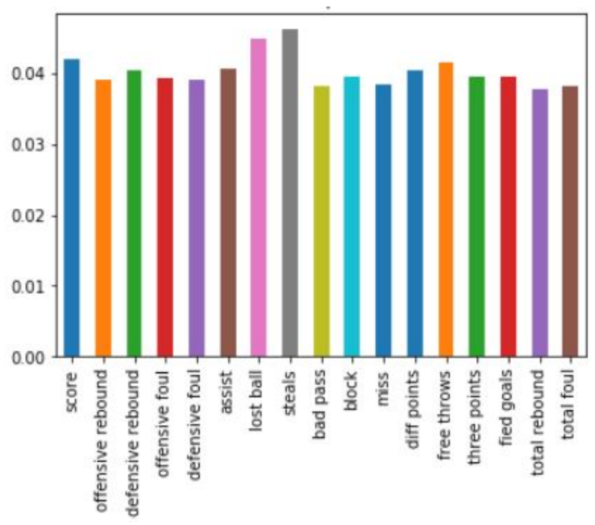


Figure 5: Feature importance in Gradient Boosting model

In order to reduce the computational time, we average each feature over 10 second sub-sequences. However, the computational time remains long so we have decided to evaluate this model using a classic cross-validation instead of using k-folds. Figure 5 then displays the weight of each feature in the gradient boosting model. As previously, the weight of one feature is the aggregation of its 1440 weights. The Gradient Boosting classifier seems to use equivalently each of the 17 features in its prediction. It might explain the good results of this classifier: 74.32% of validation accuracy and 75.01% of submission accuracy. However, the model still overfits the training set since it achieves more than 99% of training accuracy. Playing on the parameters to reduce overfitting surprisingly degraded the validation (and submission) accuracy. We then performed a principal component analysis to project the observations on the two first principal components using all features as shown in figure 6. The first and second components respectively explain 66% and 13% of the variance. We then notice that there are two clusters: a first one on the left and a second one on the right that are overlapping each other.

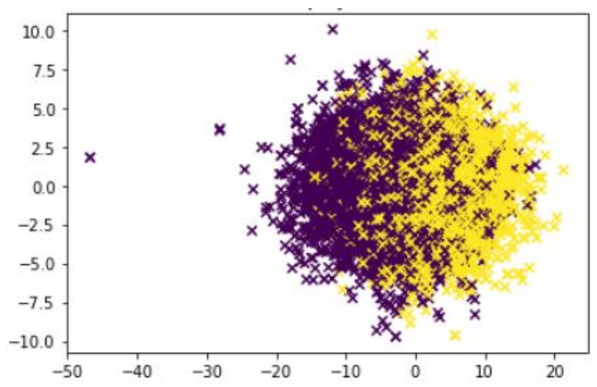


Figure 6: Principal Component Analysis using all features

4 Use of frequency features

In order to capture the variations of each feature during the game, we decided to mix time and frequency features. Each feature can indeed be seen as an element of a time series describing one of the variables throughout the game. Therefore, our idea was to extract frequency features on the signals and to couple them with time features in order to improve our prediction.

4.1 Fourier Transform

First, as frequency features, we decided to use the coefficients of the Discrete Fourier Transform. We decided to focus on the first 200 frequencies that we coupled with time features. In order to reduce the number of those time features, we aggregated them in range of 10 seconds. Therefore, we obtained 5,848 new features (17*200 + 17*144): for each of the 17 features we took their first 200 Fourier coefficients and we coupled them with time features aggregated by range of 10 seconds.

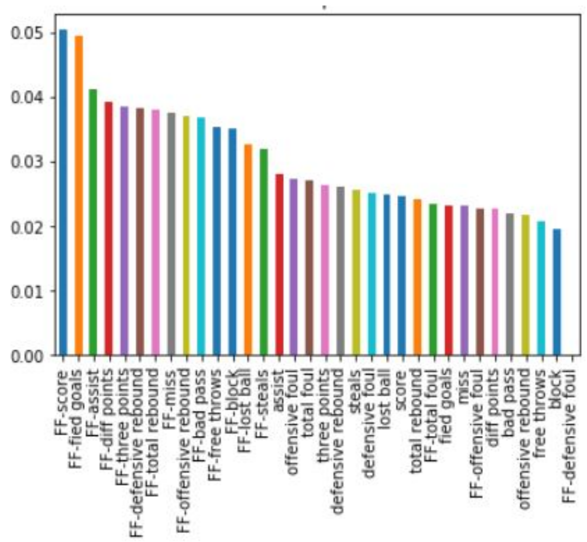


Figure 7: Feature importance with frequency features using Fourier Transform and a Random Forest classifier

On top of these time and frequency features, we used a Random Forest Classifier in order to predict the winning team. Evaluated on the test set, this method reaches 73.19% of accuracy while we had 73.99% (+/- 1.26%) of validation accuracy and 96.49% (+/- 0.07%) of training accuracy. Using the Random Forest classifier, we were able to evaluate the different features according to their importance in the prediction. Figure 7 displays the ranking of features by importance in the classification. The names of the form "FF-{name}" represent the frequency features while the other names represent time features. Here again, for each feature in time and frequency, we aggregated their coefficients in order to rank them. We notice that the classifier seems to use more frequency features than time features which reinforces us in our method. It is worth noting that the features that are the most important in frequency are not those that are the most important in time. Besides, it is also interesting to note that the model does not use at all the frequency feature "FF-defensive foul". To perform dimension reduction, we projected the observations of the 14 variables that have the highest influence on the two first principal components using a Principal Component Analysis as represented in figure 8 where the two clusters seem to totally overlap each other. In this case, 69% and 11% of the variance are respectively explained by the first and second principal components.

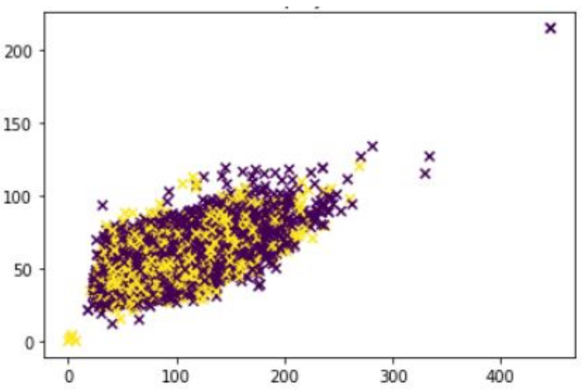


Figure 8: Principal Component Analysis for the 17 main features using Fourier Transform

4.2 Wavelet Coefficients

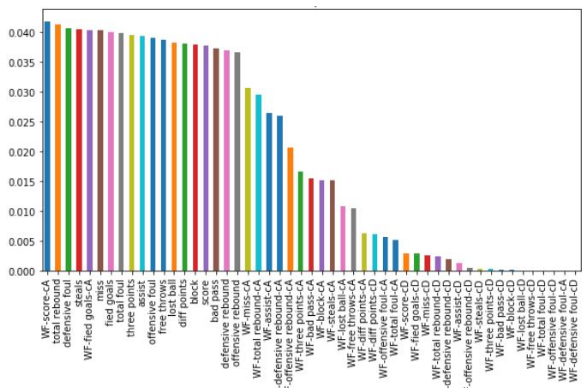


Figure 9: Feature importance with frequency features using Wavelet Coefficients and a Random Forest classifier

In a second step, we used the wavelet decomposition to capture the variations of the signal. After trying several Wavelet families (Haar Wavelets and Daubechies wavelets for instance) we finally chose to use the biorthogonal wavelets that lead to better results. Then, in the same way as before, we focused on the 200 highest frequencies that we have coupled with time frequencies. We then obtained 400 frequency features for each descriptive variable: 200 features for both approximation and detail coefficients. Besides, in order to reduce the number of time features, we also aggregated time features in range of 10 seconds. Thus, we obtained 9,248 features: 400x17 frequency features and 144x17 time features. We then used a Random Forest classifier on top of these features. Using 5-folds cross validation, this model was able to achieve a training accuracy of 96.43% +/- 0.18% and validation accuracy of 73.43% +/- 1.23%. Figure 9 illustrates the ranking of the features by importance according to the Random Forest classifier. The names of the form "WT-{name}" represent the frequency features. However, we note that among the 19 most important features, the model used only two frequency features in its prediction: the approximation coefficients of score and fields goals. Besides, most of the frequency features seem to have little or no influence on the prediction. Figure 10 illustrates the projection of the 19 most important features on the two first principal components using a Principal Component Analysis. We are able to distinguish two clusters: a yellow one on the right and a purple one on the left. The first two components are able to capture most of the variance of the dataset since they account for 79% of the variance: respectively 66% and 13% for the first and second components.

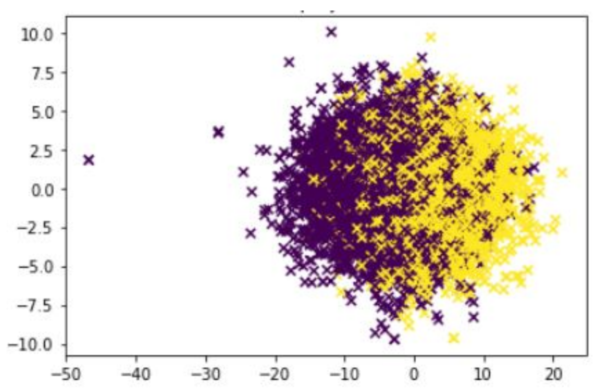


Figure 10: Principal Component Analysis for the 19 main features using Wavelet coefficients

5 Neural Network

We explored the use of neural networks to predict the winning team and see that they are really powerful tools to learn representations.

5.1 Convolutional Neural Networks

Following the work of [7] and [1] in *Automated Speech Recognition* to learn representations from sequences, we used convolutional neural networks to perform classification. Our inputs are formatted so that each match is described by a matrix of size 18x1440 (17 previously used features plus two points score for each of the 1440 seconds). We trained several architectures with one or two convolutional layers followed by one or two dense layers and the classification layer (softmax layer), resulting in not too deep networks. Several architectures were compared using 5-fold cross validation. We also used dropout ([5]) and Batch Normalization to improve validation score.

The best model trained has the following architecture:

- Batch Normalization
- 16 Convolutional Filters of size 18x10 with *relu* activation
- Dropout of 0.75
- Dense layer with 50 hidden units and *relu* activation
- Dropout of 0.5
- Classification Layer: dense layer with 2 hidden units and *softmax* activation

We used early stopping to prevent overfitting: the network was trained for 10 epochs using Adam learning scheme [4] and categorical cross entropy loss. It leads to a 73.85% +/- 0.45% training accuracy compared to a 72.95% +/- 0.55% validation accuracy, which is a slightly better score than the basic benchmark.

We extracted the features learnt by the last layer of the network to analyze the representations learnt: each observation of size 18x1440 is given to the network and the 50 units of the last layer are exploited to represent each of the observations. It is a way to perform dimension reduction and feature extraction. Figure 11 shows the projections of the train and validation observations represented by those 50 features on the two first principal components (Principal Components Analysis is learnt on the training set with the reduced representation, and then applied both to training and validation set). Principal components 1 and 2 respectively capture 75% and 15% of the variance. The network has clearly learnt some structured representation from the data as we have a strong separation between class 0 and 1. Moreover, this structure is preserved on the validation set which strengthens our belief that the network generalizes correctly. On both figures, we see that there is a strong concentration of observations at the edge of the "V" which are not clearly separated on one the components.

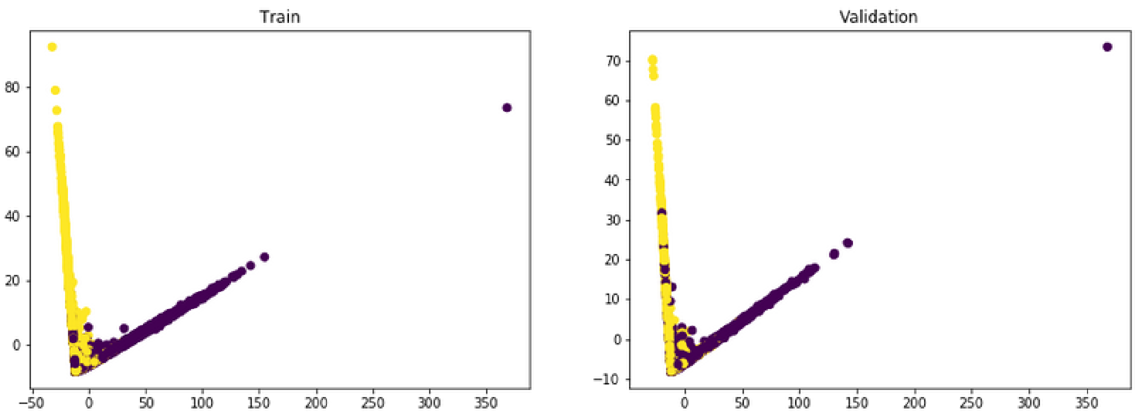


Figure 11: Principal Component Analysis on the reduced features

We built a classifier on top of these reduced features. Intuitively, a classifier will tend to cut the "V" in two parts. Figure 12 shows the predictions compared to the ground truth using a *Random Forest* classifier. The classifier achieves 73.25% accuracy on the validation set which is slightly better than the accuracy of the raw neural network. The classifier has mainly separated the "V" into two parts. Improving the score mainly implies improving the representation for it to become more discriminative (especially around the edge of the "V") and in a much lesser extend, improving the classifier.

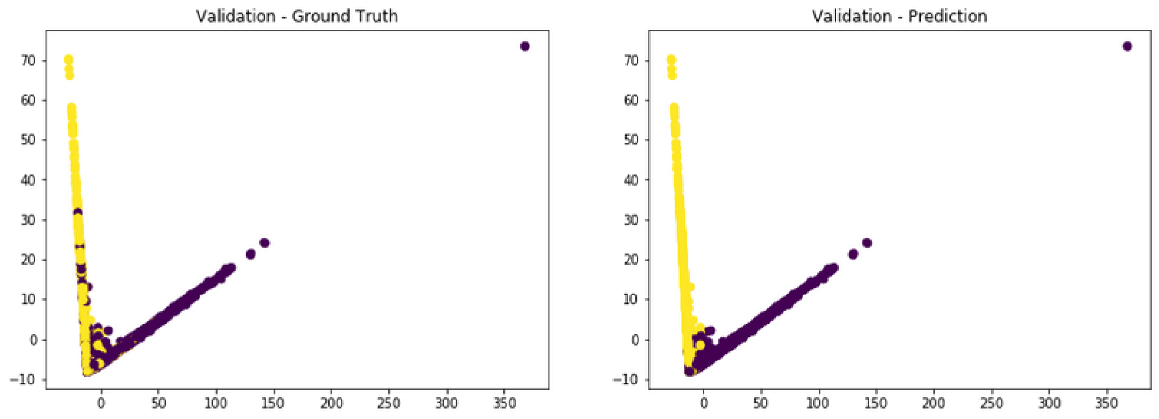


Figure 12: Comparison between ground truth and predictions labels on validation set with classifier trained on top of reduced features from CNN

5.2 Recurrent neural networks

Inspired by the wide use of Long Short Term Memory (LSTM) networks in *Natural Language Processing* ([6], [2] and [3]) for as diverse tasks as machine translation or automated speech recognition, we explored the use of LSTM to extract and build a representation from sequences.

As previously, several architectures were trained to find a good combination of parameters. However, modelling all the seconds of the match is not necessary. We reduced the dimentionality of an observation by averaging each of the features on sub-sequences of length 10,15, 20 or 24 seconds. For instance, considering 10 sec sub-sequences, an observation will be represented by a vector of size 18x144 rather than 18x1440. The best model found was trained on 20 seconds sub-sequences aggregation and has the following architecture:

- LSTM with hidden state size of 125
- Dropout of 0.75
- Dense layer with 75 hidden units and *relu* activation
- Dropout of 0.75
- Classification Layer: dense layer with 2 hidden units and *softmax* activation

Categorical cross entropy loss was used along with Adam learning scheme. The network was trained for 30 epochs using a batch size of 32. It leads to the following 5-fold cross validation results: 73.10% +/- 0.28% on train and 72.04% +/- 1.01% on validation. This is slightly lower than with CNNs.

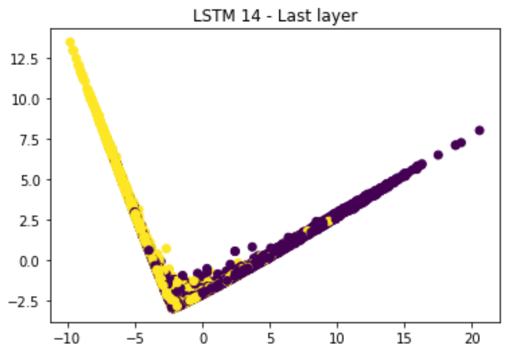
The most interesting point here is the representation learnt by the network on the last layer. We took the same neural network (described previously) to study the influence of the dense layer at the end of the LSTM. Figure 13 shows the PCA projections of the reduced features learnt by the network with or without dense layer at the output of the LSTM (if there is no dense layer, the output of the LSTM is the reduced representation learnt). Several interesting lessons should be formulated:

- First, figure 13a displays a similar "V" structure to what we have observed with CNNs. It streghens our belief that on one hand, the data has a particular latent structure and regularity that is discriminatory and neural network (if well designed) can capture this latent representation. However, the angle of the "V" seems bigger which suggests that this representation is more discriminative.
- The last dense layer has a clear and strong effect on the representation learnt. There is no obvious structure in figure 13b except that there is a yellow mass on the left and another purple mass on the right, and these two masses are really imbricated.
- The dense layer enables high level discrimination and feature engineering. Training a classifier on top of these reduced features confirms this intuition: a *Random Forest* performs 2 to 3% better on the representation with dense layer. Compared to CNN reduced representation, we gain on average 0.5%.

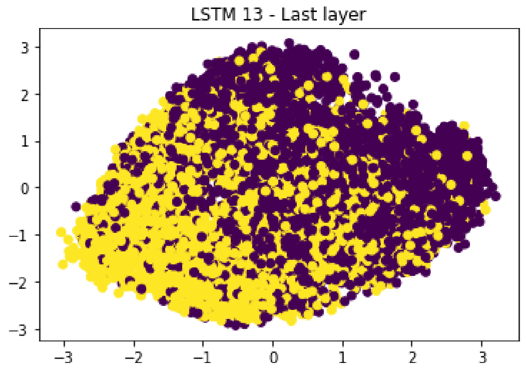
Following again the use of neural networks in Natural Language Processing and particularly [2], we finally settled on bi-directional LSTMs. Bi-directional LSTMs are simply two LSTMs that are trained on the same time: the first one learns the sequence from the beginning to the end, when the other learns the sequence from the end to the beginning, the two LSTM outputs being stacked at the end.

We trained the following architecture:

- Bi-directional LSTM with each hidden states of size 175 and recurrent dropout of 0.25
- Dropout of 0.75



(a) With Dense Layer after LSTM



(b) Without Dense Layer after LSTM

Figure 13: PCA projection on the reduced features: with or without dense layer after LSTM

- Dense layer with 150 hidden units
- Dropout of 0.5
- Classification Layer: dense layer with 2 hidden units and *softmax* activation

The network was trained for 15 epochs using batch size of 64. It leads to 72.05% +/- 0.25% training accuracy and 73.65% +/- 0.94% validation accuracy, which is the best validation accuracy obtained so far by a neural network. Figure 14 shows the PCA projection of the latent structure captured by the Bi-LSTM network. Once again, we have this ”V” structure, but it seems noisier that previously even if the validation score is higher. Figure 15 shows the predictions obtained by a fine-tuned *Random Forest* classifier that leads a 74.48% submission score which is a strong improvement of everything we have seen up to now for neural networks. This is the second best model (on submission) we have trained.

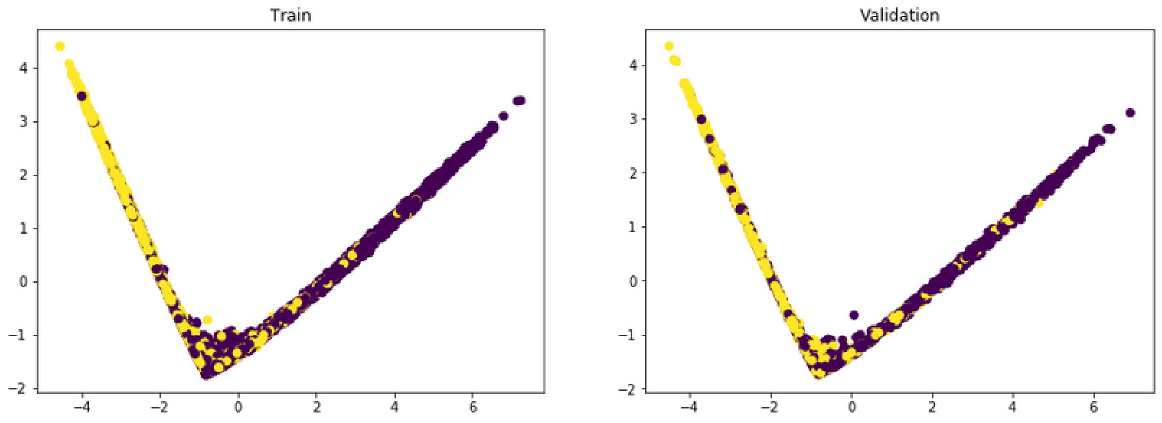


Figure 14: PCA projection on Bi-LSTM reduced features

6 Conclusion

Our work gives a strong baseline for the NBA challenge. On date, we are ranked 7th with a submission score of 75.01% (using a gradient boosting classifier) when the 1st team achieves 75.58% submission

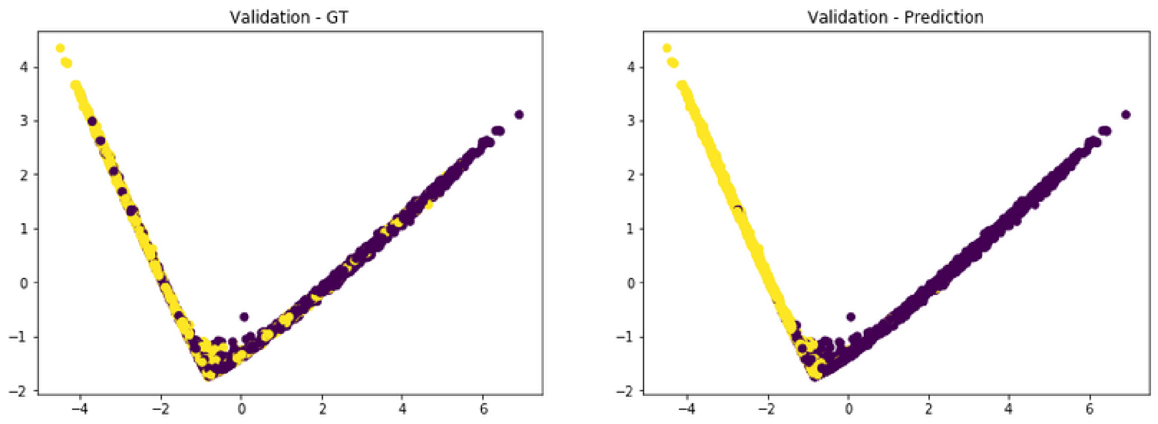


Figure 15: Comparison between ground truth and predictions labels on validation set with classifier trained on top of reduced features from Bi-LSTM

accuracy. The value of our work lies in the models we studied and analyzed, and not so much in the submission score we achieved (reasonable ranking but there is room for improvement). Particularly, the latent structures extracted with the help of neural networks strengthen our belief that the representation we have built is relevant and powerful.

The result suggest that this challenge is pretty simple: it only needs two well-designed features to correctly classify. We already know that the score is heavily correlated with the winning team so one of these two features should be highly correlated with the score. However, we think that neural networks performs worse than classic classifiers mainly because of the insufficiently high number of observations.

We strongly believe that we can significantly gain 1% or even 2% on submission (and thus improve our ranking) by creating other raw high-level descriptive features such as the ball possession ratio, offensive rebound ratio, ratio between attempted and scored shots, etc.

References

[1] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(10):1533–1545, Oct 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014.

[3] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[6] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709, 2017.

[7] Ying Zhang, Mohammad Pezeshki, Philemon Brakel, Saizheng Zhang, César Laurent, Yoshua Bengio, and Aaron C. Courville. Towards end-to-end speech recognition with deep convolutional neural networks. *CoRR*, abs/1701.02720, 2017.