README for the performance measurement part

Group 2

December, 2nd, 2019

Abstract

This Chapter is describes the evaluation of the best sampling and model combination, which is computed in the script 'PerformanceMeasurement.R'.

The goal of our performance measurement

After training a model it is important to check how well it performes. Since we not only trained one model but 120 in total (6 algos * 5 samplings * 4 positions), it is even more important to compare all the different models to see which one is the best. For our business case, this means that we want to see which combination of method and sampling we should take to predict which position, or if we even have a model, which does it better when we leave them together.

How to compare the method/sampling combinations for all positions

After training the models with 10-fold cross-validation, we applied them on all the (unsampled) data from 2007 to 2013 to obtain the true positives, true negatives, false positives and false negatives on the training set. At the end of every model-script, we save this information separately, to bring it all together in the script 'PerformanceMeasurement.R'. At the same time (and in the same scripts) we also compute the testing fit on the 2014 testing data.

The first step, is to bring them all into one dataframe, to use them easier. Then we make sure, that we used the same, unsampled data by computing the sums of TP, TN, FP, FN, for every method/sampling/position combination. In the CheckTibble we now see, that for every position the whole column contains always the same number, which means this is the case.

Then we calculate the Accuracy, Precision, Recall and the F1 score:

$$\text{Accuracy} = \frac{\text{Correct Classifications}}{\text{All Classifications}} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision (Positice Perdictive Value)} = \frac{TP}{TP + FP}$$

$$\text{Recall (Sensitivity)} = \frac{TP}{TP + FN}$$

$$\text{F1 score (harmonic mean of precision and recall)} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The result is a table showing the Accuracy, Precision, Recall and F1 scores for all the 120 model/sampling/position combination. We decided to us the F1 score as the model estimator, because is more sensitive to the inequality of availability of drafted vs. undrafted players in the target value. Therefore we just visualize the F1 score and the accuracy of all the models for the testing set. This table is quite big, but it is still interesting to have a look at it, to see how well which combination performes.

Table 1: F1 score by Model/Position and Sampling on 2014 unsampled testing data

Method	Sampling	QB_F1	WR_F1	RB_F1	Together_F1
ClassificationTree	no_sampling	0.5161	0.1702	0.4000	0.2299
ClassificationTree	oversampling	0.3913	0.3621	0.3774	0.3280
ClassificationTree	undersampling	0.3175	0.3443	0.3385	0.3294
ClassificationTree	Rose_both	0.3462	0.3281	0.2667	0.2814
ClassificationTree	Smote	0.3830	0.3559	0.2759	0.3399
randomForest	no_sampling	0.3704	0.2979	0.4516	0.3619
randomForest	oversampling	0.3750	0.3881	0.3902	0.4122
randomForest	undersampling	0.3279	0.3393	0.3380	0.3443
randomForest	Rose_both	0.4255	0.4043	0.3600	0.3776
randomForest	Smote	0.4211	0.3765	0.3922	0.3860
ANN	no_sampling	0.6154	0.3860	0.5294	0.4779
ANN	oversampling	0.2857	0.2683	0.2376	0.2573
ANN	undersampling	0.2933	0.2635	0.2376	0.2683
ANN	Rose_both	0.3056	0.2500	0.2330	0.2611
ANN	Smote	0.2941	0.2316	0.2286	0.2472
KNN	no_sampling	0.2727	0.3684	0.2500	0.3636
KNN	oversampling	0.3390	0.2617	0.2222	0.3289
KNN	undersampling	0.3226	0.3878	0.3288	0.3504
KNN	Rose_both	0.3175	0.2222	0.2169	0.2868
KNN	Smote	0.3636	0.2752	0.2857	0.3333
NaiveBayes	no_sampling	0.4348	0.2812	0.1739	0.1194
NaiveBayes	oversampling	0.3279	0.2833	0.2456	0.2941
NaiveBayes	undersampling	0.3279	0.2667	0.3265	0.2737
NaiveBayes	Rose_both	0.3571	0.2432	0.1379	0.2900
NaiveBayes	Smote	0.3571	0.2906	0.2667	0.3457
LogisticRegression	no_sampling	0.2875	0.3852	0.4675	0.4873
LogisticRegression	oversampling	0.3266	0.1649	0.4525	0.5862
LogisticRegression	undersampling	0.2734	0.4777	0.4320	0.5870
LogisticRegression	Rose_both	0.3393	0.5081	0.4083	0.6002
LogisticRegression	Smote	0.3393	0.5081	0.4083	0.6002

Table 2: Accuracy by Model/Position and Sampling on 2014 unsampled testing data

Method	Sampling	QB_Acc	WR_Acc	RB_Acc	Together_Acc
ClassificationTree	no_sampling	0.8707	0.8911	0.9082	0.9000
ClassificationTree	oversampling	0.7586	0.7933	0.8316	0.7493
ClassificationTree	undersampling	0.6293	0.7765	0.7806	0.7448
ClassificationTree	Rose_both	0.7069	0.7598	0.7194	0.7179
ClassificationTree	Smote	0.7500	0.7877	0.6786	0.7507
randomForest	no_sampling	0.8534	0.9078	0.9133	0.9000
randomForest	oversampling	0.8276	0.8855	0.8724	0.8851
randomForest	undersampling	0.6466	0.7933	0.7602	0.7612
randomForest	Rose_both	0.7672	0.8436	0.8367	0.8179
randomForest	Smote	0.8103	0.8520	0.8418	0.8433
ANN	no_sampling	0.9138	0.9022	0.9184	0.9119
ANN	oversampling	0.5690	0.6648	0.6071	0.6209
ANN	undersampling	0.5431	0.6564	0.6071	0.6418

Method	Sampling	QB_Acc	WR_Acc	RB_Acc	Together_Acc
ANN	Rose_both	0.5690	0.6480	0.5969	0.6284
ANN	Smote	0.5862	0.5922	0.5867	0.6000
KNN	no_sampling	0.8621	0.9330	0.9388	0.9269
KNN	oversampling	0.6638	0.7793	0.6786	0.7746
KNN	undersampling	0.6379	0.8324	0.7500	0.7731
KNN	Rose_both	0.6293	0.7067	0.6684	0.7104
KNN	Smote	0.6983	0.7793	0.7194	0.7672
NaiveBayes	no_sampling	0.7759	0.8715	0.9031	0.9119
NaiveBayes	oversampling	0.6466	0.7598	0.7806	0.7851
NaiveBayes	undersampling	0.6466	0.7542	0.8316	0.7940
NaiveBayes	Rose_both	0.6897	0.6872	0.8724	0.7881
NaiveBayes	Smote	0.6897	0.7682	0.6633	0.8418
LogisticRegression	no_sampling	0.8007	0.8761	0.8704	0.8875
LogisticRegression	oversampling	0.7866	0.7834	0.7660	0.8358
LogisticRegression	undersampling	0.8272	0.7344	0.7391	0.8345
LogisticRegression	Rose_both	0.7802	0.8319	0.6330	0.8405
LogisticRegression	Smote	0.7802	0.8319	0.6330	0.8405

We can see these two tendencies in the models:

- Mostly better performance with unsampled data, than with sampled data
- Mostly better performance when we split the positions manually

ThThe four following plots combine the information of both tibbles, showing the accuracy and the F1 score on every sampling method for every position on its own. The horizontal lines show the no information rates for the accuracy, which displays, how a model's performance would be when predicting only 0's.

RUBEN PLOTS

Our best models

Now let's have a look at the best model/sampling combination for every position:

Table 3: The best model/sampling combinations by position

Position	Method	Sampling	F1	Accuracy
QB	ANN	no_sampling	0.6154	0.9138
WR	LogisticRegression	Rose_both	0.5081	0.8319
RB	ANN	no_sampling	0.5294	0.9184
Together	LogisticRegression	Rose_both	0.6002	0.8405

As we see, measured at the F1-score, the atrificial neural networks perform best for QB's, RB's and all positions together. For the WR's it is the rose-both-sampled data trained random forest.

Discussion

Our models with accuracies up to 91.8% seem to be very good at the first sight. But we have to keep the reason for filtering out the players with <10 played games and sampling the data in mind. Here we applied the models to the unsampled but filtered data of 2014, which only contains 7.01% of drafted players. This means, that a model predicting "not drafted" for every player would still perform better, since it would still have an accuracy of 92.99% on the whole data set (='Together'). In other words, our models all perform unter the 'no information rate', which makes them not really good.

Interpretations of models are always quite difficult, but the following thoughts are pretty likely to be true, according to the TP/TN/FP/FN. Our models are okay at predicting the likelihood of players being drafted that didn't perform very well in college football being low. It probably also does not too bad in predicting great players to be drafted, that nearly must be picked (and probably are picked early in the draft). But there must be much room for improvement for all the players that still performed well in college, and might or might not be drafted. Again in other words, the models can predict the more or less obvious drafts and non-drafts but is not really better than random for the intresting cases.

We would like to close the circle to one of our fist lessons in machine learning, in which we were taught the following very high level formula for models (the right part). Y denotes the true outcome (in our case whether a player is drafted or not), f(X) is the true pattern that describes it, and ϵ is the noise, which appears to be random. With our models (the left part) we try to predict a $\hat{Y}$, which shall be as close as possible to the true Y.

$$\hat{f}(X) = \hat{Y} \approx Y = f(X) + \varepsilon$$

Looking at this inequation we can think of three possibilities, why our models make so many mistakes:

- Our models $\hat{f}_i(X)$ do not include enough variables and/or are not sophisticated enough
- The data is not good enough
- The NFL draft contains a pretty large ε

Variables that are certainly missing in the model, are the ones that are not quantifyable easily, such as game intelligence, strenght of the own team, strenght of the opponents, maturity of the player and negative factors such as criminality, drug consumation and other negative behaviours. In the past, it happened again and again, that players with great game statistics and a great game intelligence, which were expected to be drafted in the first round fell far behind or were not drafted at all, because pictures of them consuming marihuana were published.