EXPERIMENT REPORT

Student Name	Roger Yu
Project Name	MDSI ADSI Assignment 1 Part C
Date	2020-02-21
Deliverables	yu_roger-10906675-week3_voting_classifier.ipynb 10906675_voting_classifier.joblib https://github.com/roger-yu-ds/assignment_1/tree/roger

---

1. EXPERIMENT BACKGROUND	
Provide information about the problem/project such as the scope, the overall objective, expectations. Lay down the goal of this experiment and what are the insights, answers you want to gain or level of performance you are expecting to reach.	
1.a. Business Objective	NBA draft picking. The teams would like to pick players with a high probability of having a career greater than 5 years.
1.b. Hypothesis	Predicting more negative classes Reducing the `scale_pos_weight` from the default of 1 would make the model predict more negative classes and improve the validation/test AUC scores. Further reduction of overfitting • Early stopping is used in all experiments • Reduction in the number of features Adversarial validation Previous experiments indicated that the distributions of the training and test sets are different, this was hypothesised because the test AUC proved to be much lower than the validation AUC. If the training set and test sets were indeed different to a significant extent, then a classification algorithm would be able to tell them apart, i.e. if the observation came from the training set or from the test set.

	# Stack Ensemble: Voting Classifier Including the predictions/probabilities from several algorithms and conducting a soft/hard voting could overcome some of the deficiencies/biases of any one particular algorithm and result in a better performance over all.
1.c. Experiment Objective	1. Improve the validation AUC beyond 0.70 2. Reduce overfitting as indicated by the difference between the training and validation AUC, i.e. 0.13 3. Determine if there are significant differences in the training and test sets 4. Visualise the calibration curve

---

2. EXPERIMENT DETAILS
Elaborate on the approach taken for this experiment. List the different steps/techniques used and explain the rationale for choosing them.

2.a. Data Preparation

Original classification

No notable preparation steps in addition to those in the previous reports.

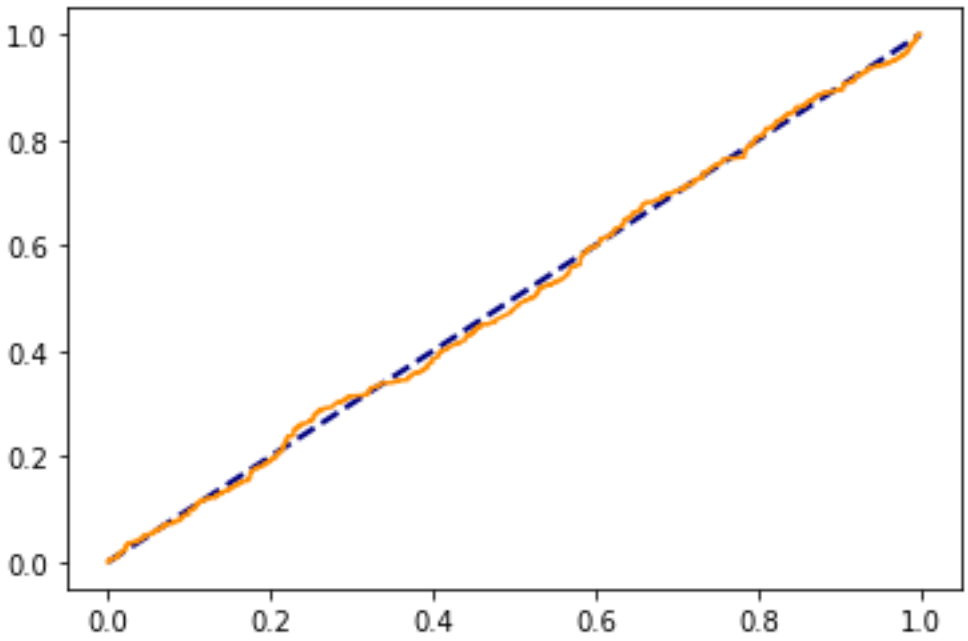
Adversarial validation

1. Columns are dropped.

Column	Training Set	Test Set
Id Old	drop	drop
Id	drop	drop
TARGET_5Yrs	drop	na

2. A new column “dataset” is added, with values of “train” and “test”, for the training and test sets respectively. This column is the target column of the adversarial validation
3. The data is sampled without replacement to produce an adversarial training set and an adversarial testing set; the sizes of the sets are the same as that of the original training and test sets.

2.b. Feature Engineering	NA
2.c. Modelling	`scale_pos_weight` (10906675_xgb_es_spw.joblib) The XGB classifier parameters were the same as the best model from the previous experiment, from the model randomised_xgb.joblib The investigation starts with a coarse search space in the range of (0, 1) with a step size of 0.1, which resulted in an optimal weight of 0.5. After which a finer grid is searched with the range (0.4, 0.6) with a step size of 0.01, the results on the validation set are shown in the table below. ## Further reduction of overfitting A GridSearchCV was run over the range (1, 7) of numbers of PCA components; 8 components was the random search previous found to be optimal. The base estimator is the model in the previous section (10906675_xgb_es_spw.joblib). ## Adversarial validation An XGB Classifier was fit on the adversarial training set. The test AUC was close to 0.5, which indicates that there was significant difference between the training and the test sets. The chart below shows that the ROC curve (orange line) is quite close to the theoretical unskilled classifier (dashed line) that always predicts the mode.



Calibration Curve

The base estimator is the model in the previous section (10906675_xgb_es_spw.joblib). While the calibration curve on the training set looks reasonable, if a little

Stack Ensemble: Voting Classifier

All the saved estimators from my previous experiments were passed into the VotingClassifier for soft voting, i.e. the predictions are weighted by the classifier's importance and summed up, then the target label with the greatest sum of weighted probabilities wins the vote.

---

3. EXPERIMENT RESULTS

Analyse in detail the results achieved from this experiment from a technical and business perspective. Not only report performance metrics results but also any interpretation on model features, incorrect results, risks identified.

3.a. Technical Performance

`scale_pos_weight`

The finer grid search achieved a `scale_pos_weight` value of 0.55, which has slightly higher validation results, see table below.

scale_pos_weight	AUC	Class 0 f1-score	Class 1 f1-score
1	0.6989	0.01	0.91
0.55	0.6991	0.04	0.91
0.5	0.6963	0.04	0.91

Even with the `scale_pos_weight` value of 0.55, reduced by almost half, there are still very few predictions of the negative class, as shown by the low f1-score, and the confusion matrix of the validation set below.

	Predicted 0	Predicted 1
Actual 0	5	258
Actual 1	5	1332

This model (10906675_xgb_es_spw.joblib) resulted a test AUC of 0.69390.

Further reduction of overfitting

As expected, the fewer the components the less overfitting observed, with one component resulting in the smallest difference between the training and validation AUC, but also with the lowest validation AUC; the table below shows the

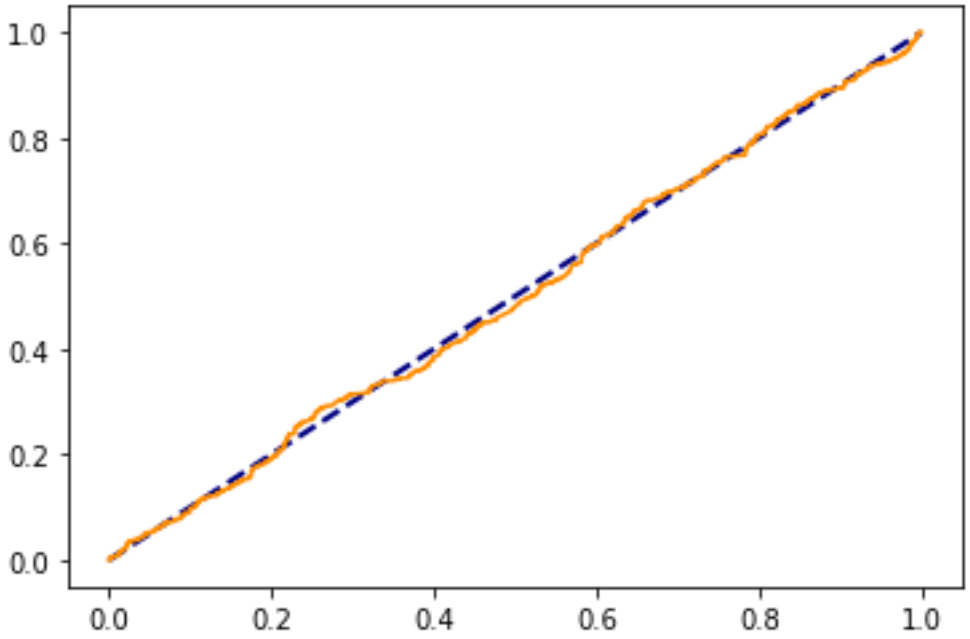
n_components	train_auc	val_auc	diff_auc	ratio_auc
1	0.685	0.685	0.001	0.001
2	0.706	0.681	0.025	0.035
3	0.706	0.682	0.024	0.035
4	0.719	0.691	0.029	0.040
5	0.708	0.678	0.031	0.043
6	0.744	0.683	0.061	0.082
7	0.770	0.694	0.076	0.098

The class 0 and 1 f1-scores are 0.08 and 0.91; a doubling of the class 0 f1-score compared to the previous model. The confusion matrix below shows an increase in the predictions of the negative class, although the false negative has also increased.

	Predicted 0	Predicted 1
Actual 0	12	251
Actual 1	17	1320

Adversarial Validation

Using an XGB Classification algorithm, the AUC on the adversarial test set was close enough to 0.5 to conclude that there wasn't any difference in the data sets, the ROC curve below shows that the observed curve (orange) is just as skilful as an algorithm predicting the mode (dashed line).



Stack Ensemble: Voting Classifier

This model [10906675_voting_classifier.joblib] resulted in the highest test AUC of 0.69658.

The model is still extremely overfit

	AUC	Class 0 f1-score	Class 1 f1-score
training set	0.993	0.54	0.94
validation set	0.698	0.09	0.91

However, it seems that the VotingClassifier was able to correctly predict one more of the negative class, while predicting 5 fewer correct positive classes on the validation set.

	Predicted 0	Predicted 1
Actual 0	13	250
Actual 1	22	1315

3.b. Business Impact

The model is almost always going to predict that a player will have a career greater than 5 years. So I will concentrate on the consequences of the false positive predictions. At this level of performance, the model is falsely predicting 250/(1315+250) = 16% to be greater than 5 years.

To understand the impacts to business decisions, another layer of cost and benefit is needed. What is the cost of investing in players that won't continue past the 5-year mark? And what is the benefit that the players give to the team if they do pass the 5-year mark. If there is a net benefit, then the model can be used.

Impact on the false negatives is unlikely to have a big impact. The issue is passing on a potential Michel Jordan or Lebron James, in which case extremely high benefit is are missed and instead goes to a competitor team.

3.c. Encountered Issues	# Bias towards the positive class The XGB Classifier seems to be heavily weighted to the prediction of the positive class and while reducing the `pos_scale_weight` would mitigate this effect, the drop in the validation AUC is such that the optimal value is 0.55, which also doesn't increase the predictions of the negative class by a significant amount. This is due to the imbalanced data set; SMOTE and undersampling did not seem to have solved this problem. Perhaps an entirely different algorithm is required to overcome this bias. # Collaboration in models My VotingClassifier could have performed better if the constituents were not all XGB models. We did not stipulate at the beginning the format and conventions of saving models. So I was not able to leverage of other members' models in the VotingClassifier, even though these would probably have been much more beneficial than just my own set of XGB models. Non-XGB model predictions would be less correlated and would potentially have better performance in areas where the XGB is underperforming. The problem was that the models were not saved as base estimators: 1. Some were saved as notebooks/Python scripts 2. Some were saved as `GridSearchCV` objects 3. Some were `CalibratedClassifierCV` objects, which doesn't work with `VotingClassifier` for some reason which made it difficult to incorporate into the ensemble algorithm.

---

4. FUTURE EXPERIMENT	
Reflect on the experiment and highlight the key information/insights you gained from it that are valuable for the overall project objectives from a technical and business perspective.	
4.a. Key Learning	# Group work I don't think that there has been a truly collaborative way of working in these three weeks. While we have been sharing some functions in the src directory, it's unclear as to how much is actually leveraged, I know for sure that I only used one function created by a team member (`classification_reports`), albeit I have used it extremely frequently. Only in the 11th hour did I try `VotingClassifier` on all of my models, which were sklearn estimators saved as joblib objects. And I realised that this method would have benefit greatly if I also included others' models as well. Alas I did not have easily

	access to these models, because only then did I realise that other members were saving the models as notebooks or Python scripts. A high level key learning for me in the future would be to standardise some of the ways of working so that any future collaboration could be exploited immediately. ## Best practices From what I've seen in the code that others have committed, the code doesn't adhere to the principle of single responsibility, e.g. the `submit_predictions` function performs predictions and submission. The prediction interface did not include `**fit_params` hence was not able to handle more sophisticated predictions. ## Conventions Many of the points below relate to the VotingClassifier issue that I experienced, these are the "rules" that I would likely suggest to be enforced in future projects: 1. Ensuring that the saved models as the best base estimators, i.e. not `RandomizedSearchCV` objects because otherwise `VotingClassifier` will attempt to fit the grid again, which consumes a lot of resources unnecessarily. 2. Saving `CalibratedClassifierCV` objects separately, as `VotingClassifier` throws an error. There is potentially an accepted way to deal with this but I still haven't found a good solution yet. 3. Saving models as a particular type, e.g. joblib objects, rather than notebooks or scripts 4. Using the root directory as the working directory in all notebooks

4.b. Suggestions / Recommendations	# Group Work
	
	1. Agree on the ways of working at the outset
	2. Populate the README.md file with ways of working, conventions, and protocols
	3. Review commits/pull requests and comment on how to better structure the code
	4. Plan on who will tackle
	    a. which aspects of the problem, e.g. overfitting, EDA, feature engineering
	    b. which algorithms, e.g. XGB, SVM, so that we don't double up and end up with highly correlated results, which would be ineffective for ensembling