Grant Case Study

Kuhn and Johnson, Ch. 20

Brooke Anderson

April 6, 2016

Purpose of model

2011 Kaggle competition, sponsered by the University of Melbourne, with the aim:

> *Predict whether a grant is funded based on certain characteristics of the grant.*

- Training data: 8,708 grant applications, 2004–2008
- Testing data: 2,176 grant applications, 2009–2010

Predictive variables

249 features, including:

- size of the grant (in 17 categories)
- research area (RFCD code; 738 values; $>$ 1 value possible)
- grant sponser
- "socio-economic objective" (purpose of the grant)
- submission date
- features of the investigator

In the data, some of these had non-sensical values for some observations.

Evaluation

This competition was judged on Area Under the Curve (AUC).

Interesting fact about AUC from Kaggle:

> *"AUC was first used by the American army after the attack on Pearl Harbour, to detect Japanese aircraft from radar signals."*

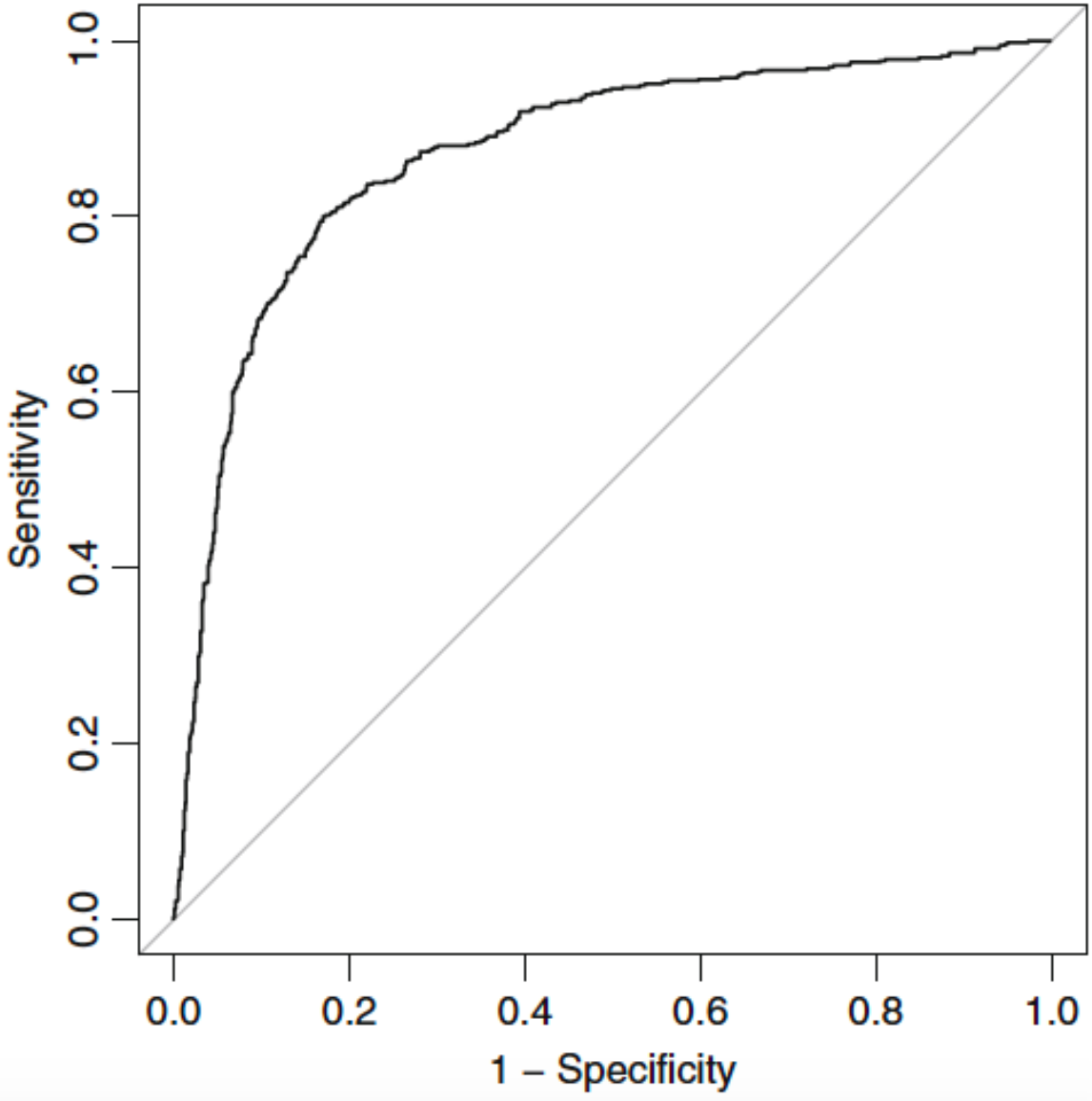
- "random guess" model: AUC $= 0.5$
- perfect model: AUC $= 1$

Competition results

Winning score: AUC $= 0.968$

Evaluation

Area under the curve:



Evaluation

Get class probabilities:

```
ctrl <- trainControl(summaryFunction = twoClassSummary,
                     classProbs = TRUE)
```

Tuning in caret with AUC:

```
lrFull <- train(training[,fullSet],
                y = training$Class,
                method = "glm",
                metric = "ROC",
                trControl = ctrl)
```

Also, see the pROC package, with roc and auc functions.

Feature engineering

Challenges:

- Suboptimal coding: Often there are separate columns for each investigator involved with a grant, for example.
- Lots of missing data

Feature engineering

New features (final: $>$ 1,500):

- Made variables for month and day of week of the grant submission
- Counted number of investigators on the grant by role (PI, etc.)
- Created 0 / 1 variables for each sponser code and grant category. For variables that could have more than one value, they set this to "1" for any existing (non-zero) value ("grouped" versus "indepedent" categories)
- For investigator characteristics (native language, degree, birth year, etc.), aggregated to counts across roles

Example:

> *"One variable counts the number of chief investigators from Australia while another counts the total number of successful grants from all delegated researchers on the grant."*

Pre-processing

Challenges	"Fix"
Missing data	Created binary "missing" predictors
Highly correlated predictors	High-correlation filter (> 0.99)
Near-zero variance predictors	Depended on model type

They used the `trim.matrix` function (`subselect` package) to handle extreme collinearity. Also, check out `findLinearCombos` in `caret`.

Handling missing data

For categorical variables, they created a new category called "missing" and used that.

> *"For example, 912 grants had missing category codes. A binary predictor for missing grant categories was created to capture this information."*

Near-zero variance predictors

For some predictors, all but a few observations were "0". Some types of models are very sensitive to sparse and unbalanced predictors.

Therefore, they created two datasets, and picked which one to use based on whether the model type is sensitive to near-zero variance predictors:

- "Full dataset": All predictors (after filtering out highly correlated predictors) (1,070)
- "Reduced dataset": Removed the near-zero variance predictors (252)

Unsupervised feature selection: Removing variables without checking if they're associated with the outcome.

Full and reduced datasets

They created vectors called `fullSet` and `reducedSet` identifying the columns of predictors in each set. They used this when they fit models:

```
lrFull <- train(training[,fullSet],
                y = training$Class,
                method = "glm",
                metric = "ROC",
                trControl = ctrl)
```

For some models, they would also center and scale the predictors.

Exploratory data analysis

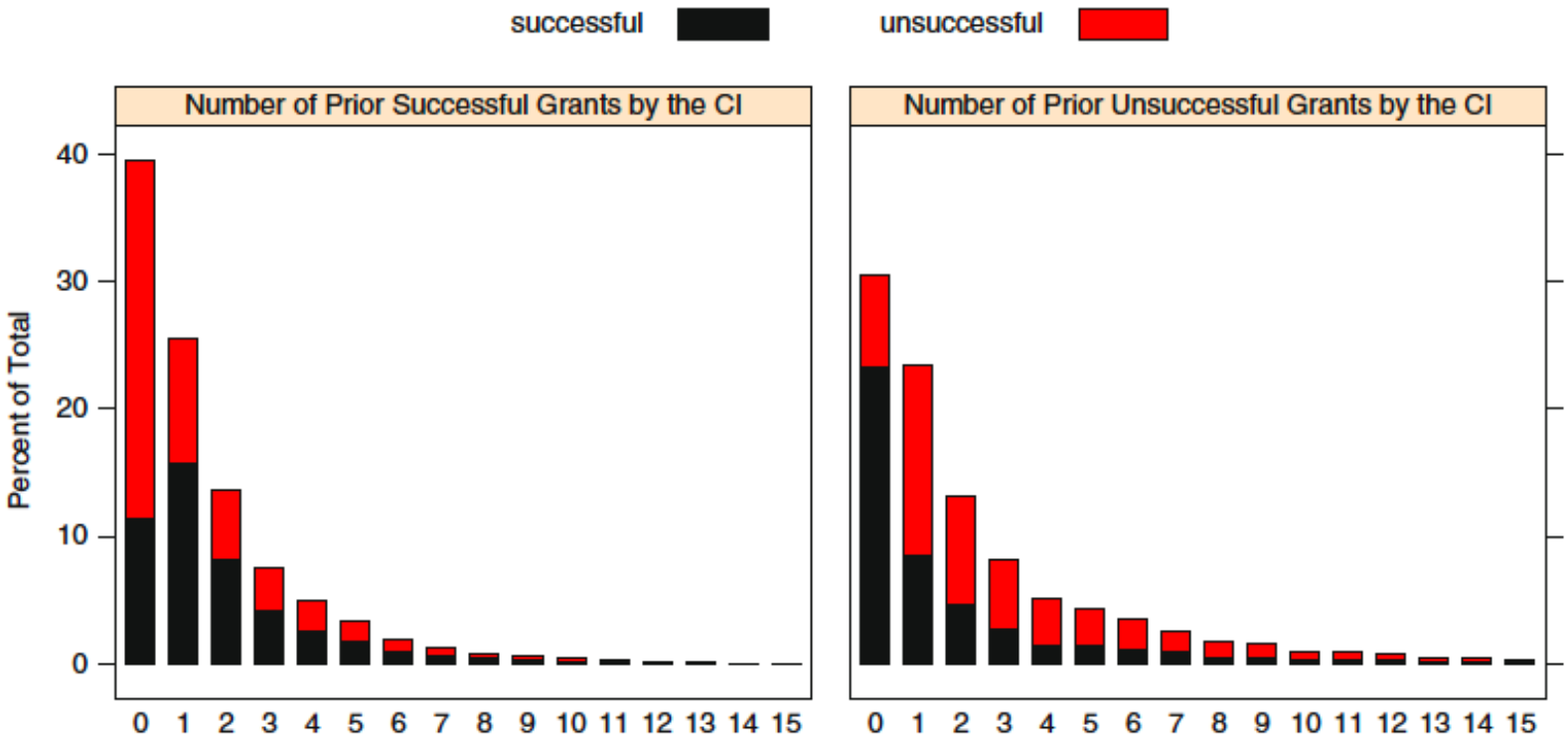
Based on univariate analysis, clearly associated predictors:

- # of prior successful grant applications by PI (count)
- # of prior unsuccessful grant applications by PI (count)
- January submission date (binary)
- Sponsor unknown (binary)
- Highest grant value category (binary)

Exploratory data analysis

	Grant success					
	Yes	No	N	Percent	Odds	Odds ratio
Contract value band						
A	1,501	818	2,319	64.7	1.835	2.84
Other bands	2,302	3,569	5,871	39.2	0.645	
Sponsor						
Unknown	732	158	890	82.2	4.633	6.38
Known	3,071	4,229	7,300	42.1	0.726	
Month						
January	480	45	525	91.4	10.667	13.93
Other months	3,323	4,342	7,665	43.4	0.765	

Exploratory data analysis



Exploratory data analysis

Grant success declined over time:

Year	Success (%)
2005	45.0%
2006	51.7%
2007	47.2%
2008	36.6%

Splitting into training and testing data

Because of the time trend in the outcome, you'll build a better model by being clever about dividing the training and testing sets.

> *"An alternative strategy [to a random test sample] would be to create models using the data before 2008, but tune them based on how well they fit the 2008 data."*

Problems:

- May overfit to 2008 and not generalize to later years
- No way to measure uncertainty (e.g., for tuning)

Splitting into training and testing data

They compromised by:

- Built and tuned models using pre-2008 data
- Tested models by checking them for a random sample of ~2,000 grants from 2008
- Build a final model using all pre-2008 data + the random sample from 2008
- Reserve a small, random holdout sample from 2008 ("2008 holdout set")
- Didn't use the "holdout" until they had fully tuned and re-built their candidate models

The holdout was used "to ensure that no gross methodology errors occur from repeatedly evaluating the 2008 data during model tuning".

Splitting into training and testing data

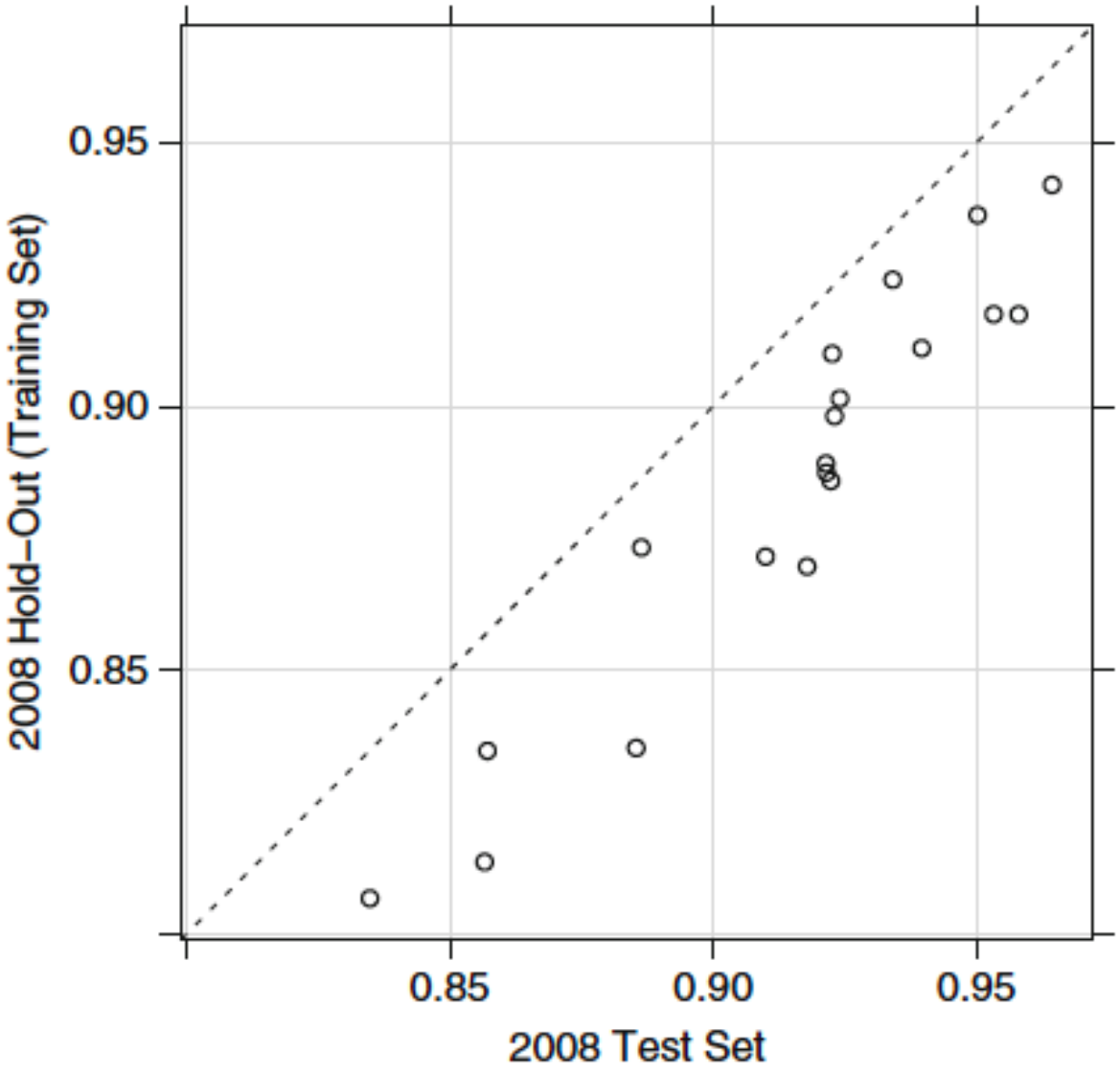
	Model tuning		Final model	
	Training	Holdout	Training	Holdout
Pre-2008 ($n = 6,633$)	×		×	
2008 ($n = 1,557$)		×	×	
2008 ($n = 518$)				×

Splitting into training and testing data

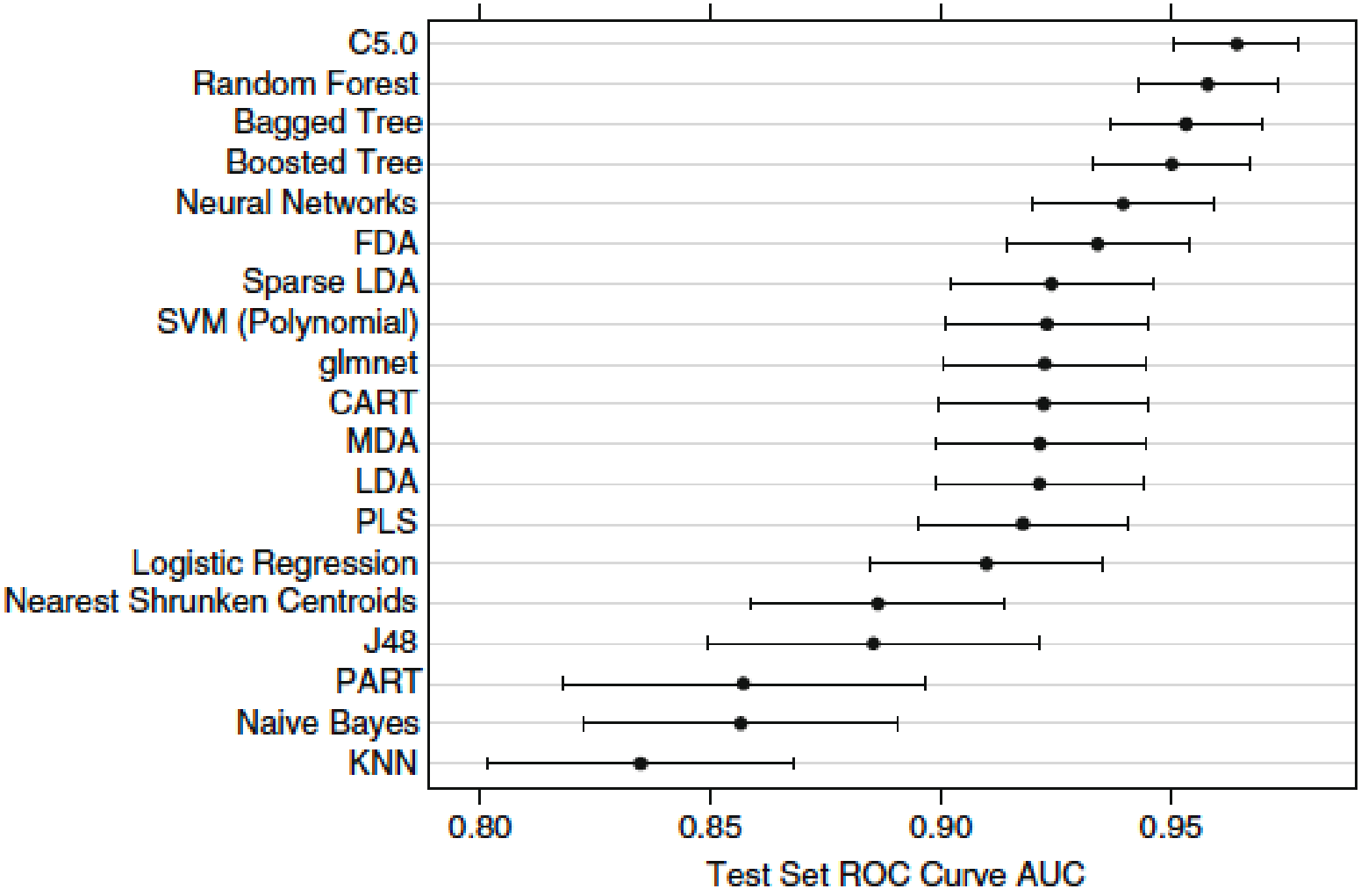
Code to make this split (note the use of the `index` argument):

```
ctrl <- trainControl(method = "LGOCV",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     index = list(TrainSet = pre2008))
```

Splitting into training and testing data



Model results



Model results

Model	Holdout set	Test set
C5.0	1	1
Boosted tree	2	4
Bagged tree	4	3
Random forest	5	2
FDA	3	6
Neural networks	6	5
Sparse LDA	8	7
glmnet	7	9
SVM (polynomial)	9	8
CART	12	10
LDA	10	12
MDA	11	11
Logistic regression	14	14
Nearest shrunken centroids	13	15
PLS	15	13
J48	16	16
PART	17	17
Naive Bayes	18	18
KNN	19	19

Model results

Important predictors:

- Past success / failure of chief investigator
- Informative missingness (e.g., for the sponsor and grant value)
- Time of year of grant submission

C4.5

- In Kuhn and Johnson, some explanation starts on p. 377
- Type of decision tree, but splitting criteria different from CART
- Modified version can group 2+ categories of categorical variables
- Like CART, needs pruning ("pessimistic pruning")
- R function: `J48` in `RWeka` package

C4.5

Information measure:

$$info = -[p \log_2 p + (1 - 0) \log_2 (1 - p)]$$

Gain for a split:

$$gain(split) = info(\text{prior to split}) - info(\text{after split})$$

Because $gain(split)$ is biased against picking predictors with many possible outcomes, instead the method uses *gain ratio*.

C5.0

- Like C4.5, but with boosting and unequal costs for different types of errors
- Source code made public in 2011
- Likely to generate smaller trees than C4.5 (in part because of final pruning procedure)
- Possible to boost (similar procedure to AdaBoost)
- `C50` package in R, code examples with boosting on p.410 of Kuhn and Johnson
- Use with `caret`, in `train` use 'method = "C5.0"