Predicting DVD purchase

In this report we'll estimate several different models on the `dvd` data. Specifically, we'll use Logistic Regression, Neural networks, and Decision trees to model the probability of purchasing a DVD after exposure to an "instant coupon" online. The models can all be estimated in Radiant.

Data description

The data contain information on a sample of 20,000 customers who received an "instant coupon." The value of the coupon was varied between $1 and $5 and randomly assigned to the selected customers.

Our interest is in estimating the effect of the coupon on purchase of a newly released DVD. We will also investigate the role of two additional variables: `purch` is a measure of frequency of purchase and `last` is a measure of the recency of the last purchase by a customer. These measures are often used in practice to predict response rates.

Customers who received the coupon and purchased the DVD are identified in the data by the variable `buy`. Because the variable we want to predict is binary (`buy` = `yes` if the customer purchased the DVD and `buy` = `no` if she did not), logistic regression is appropriate.

Variables

- buy = `yes` if the customer purchased the DVD and `no` if she did not
- coupon = value of an "instant coupon" in dollars. the value varies between $1 and $5
- purch = number of purchases by the customer in the past year
- last = days since the last purchase by the customer

Source

The dataset `dvd.rda` is available for download from GitHub.

The `dvd` dataset is simulated using characteristics from a real dataset but has less noise than the original. For the analysis we'll assume that the cost of getting the coupon offer to a (potential) customer is $1 and that the net revenue of a purchase is $8.

To keep the `dvd` data loaded in Radiant through the *Data > Manage* tab clean and ensure the analysis is reproducible we'll start by making a working copy in the *Data > View* tab. We'll call the new dataset `dvd_wrk`.

To compare model performance we need a training and a validation (or test) dataset. In Radiant we can create a `training` variable in the *Data > Transform* tab that is 1 if an observation is assigned to the training sample and 0 otherwise. Because the `dvd` dataset is very *clean* we'll use a training sample of only 20% of the data. This will increase the chance that we'll overfit the training data, particularly for the Neural Network model and (un-pruned) Decision Trees.

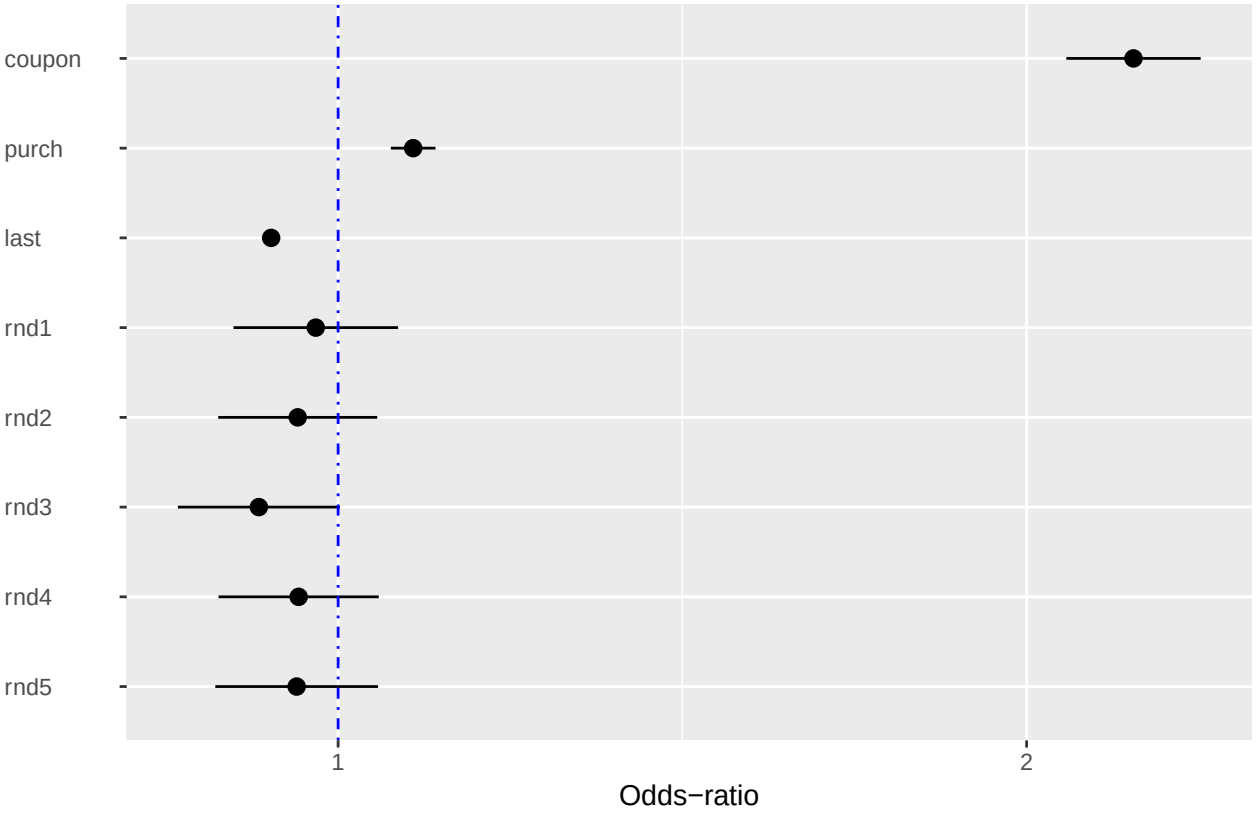
To make things even a bit more interesting lets add a few random generated variables as well.

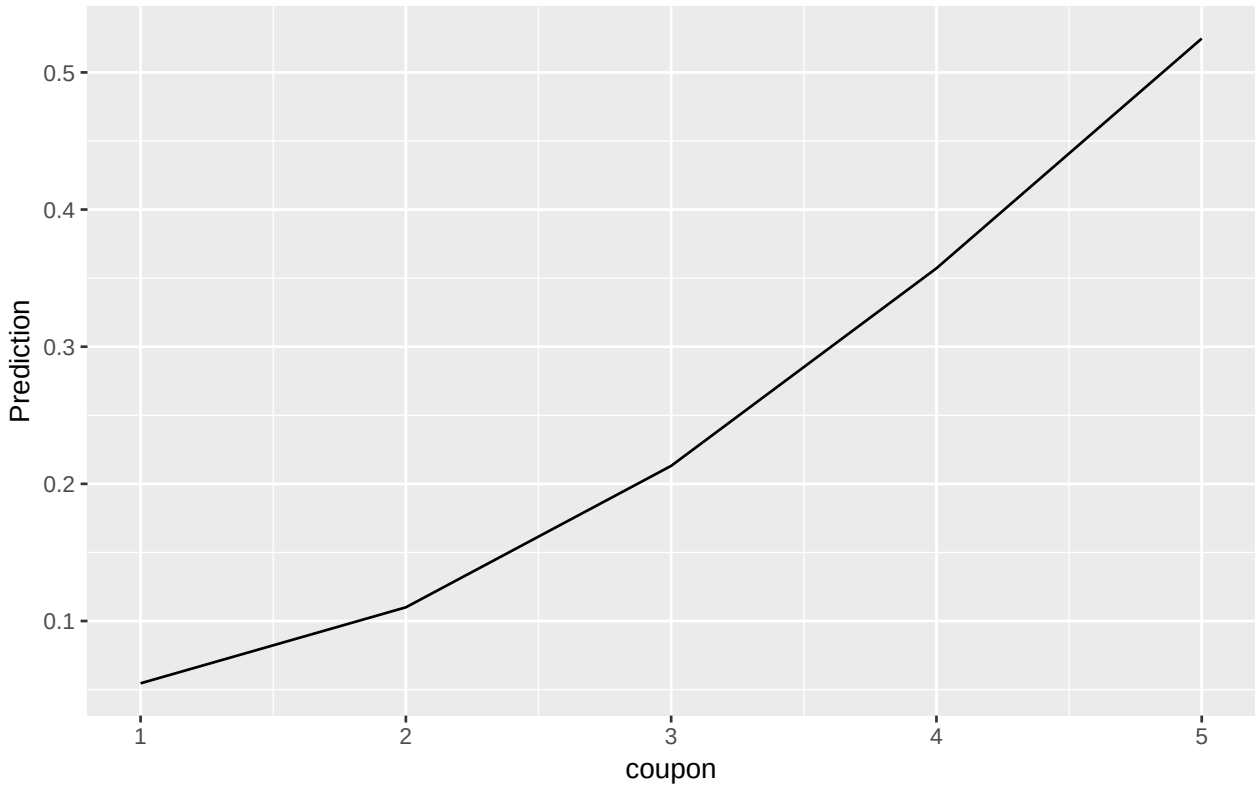
Estimate a Logistic Regression

In a first step lets estimate a Logistic Regression with all available explanatory variables and generate a prediction of the response probability of buying a DVD (i.e., `predict_logit`).

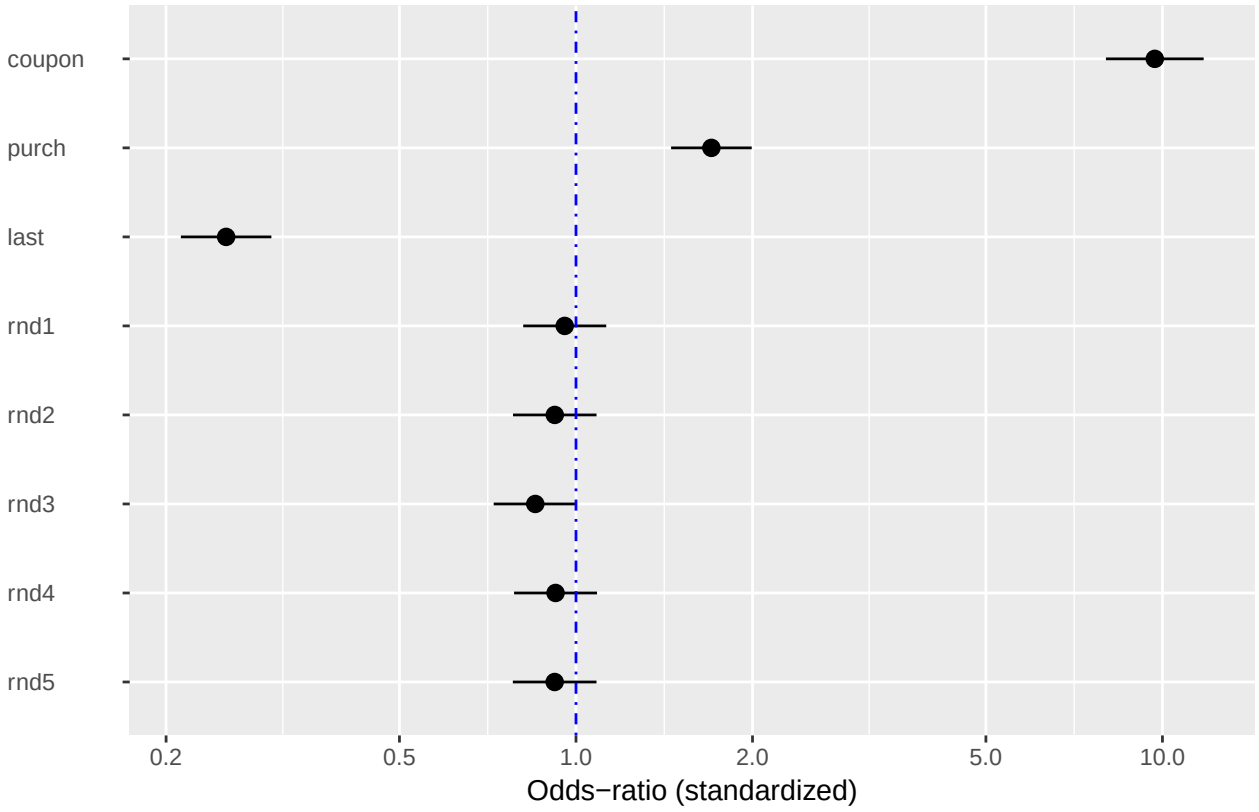
```
Logistic regression (GLM)
Data                 : dvd_wrk
Filter               : training == 1
```

```
Response variable    : buy
Level                : yes in buy
Explanatory variables: coupon, purch, last, rnd1, rnd2, rnd3, rnd4, rnd5
Null hyp.: there is no effect of x on buy
Alt. hyp.: there is an effect of x on buy

               OR coefficient std.error z.value p.value
 (Intercept)            -3.156     0.145 -21.703  < .001 ***
 coupon      2.227       0.801     0.034  23.211  < .001 ***
 purch       1.078       0.076     0.011   6.587  < .001 ***
 last        0.935      -0.067     0.004 -15.150  < .001 ***
 rnd1        0.978      -0.023     0.042  -0.532   0.595
 rnd2        0.960      -0.041     0.041  -0.996   0.319
 rnd3        0.923      -0.080     0.042  -1.918   0.055 .
 rnd4        0.961      -0.040     0.041  -0.966   0.334
 rnd5        0.959      -0.042     0.042  -0.999   0.318

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Pseudo R-squared: 0.215
Log-likelihood: -1781.448, AIC: 3580.896, BIC: 3637.542
Chi-squared: 974.905 df(8), p.value < .001
Nr obs: 4,000
```





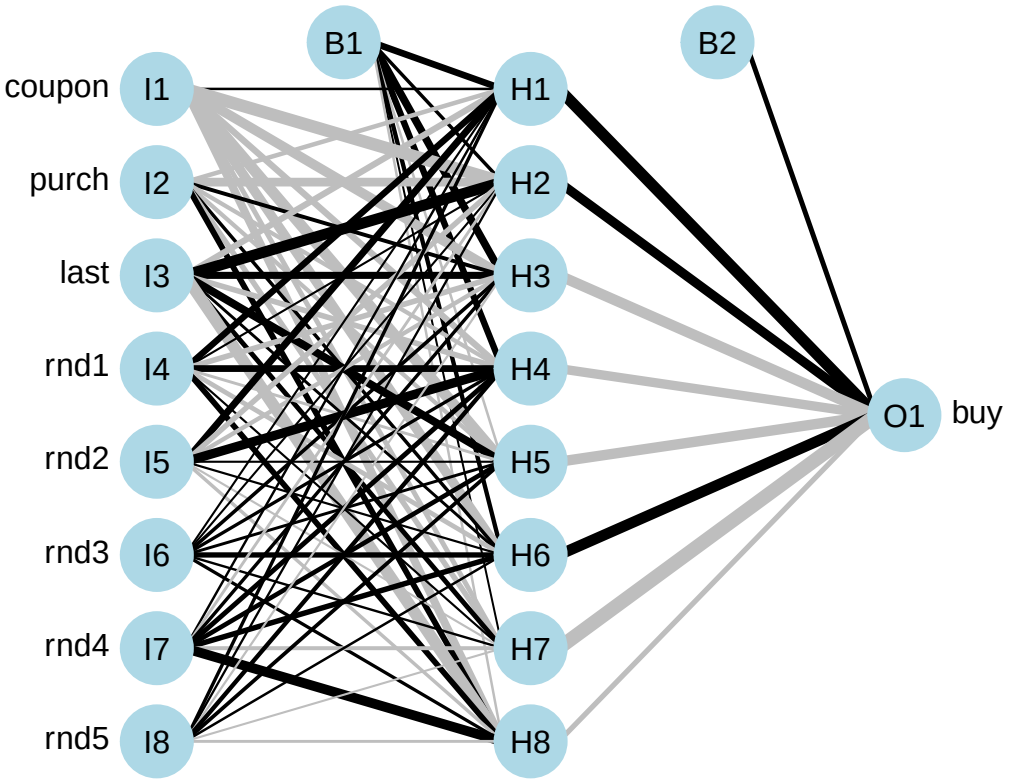
It is not straightforward to evaluate the relative importance of the variables in the model based on the plot above because the variables are on different scales. Lets re-estimate to get standardized odds-ratios.



Estimate a Neural Network

Next, we'll estimate a Neural Network with 8 nodes in the hidden layer and parameter decay of 0.5. The optimal values for these parameters can be established using a grid-search in, e.g., the `caret` package in R. The network plot below illustrates how inputs are connected to the nodes in the hidden layer and to the output node. Predicted probabilities are stored in `predict_ann`.

```
Artificial Neural Network (ANN)
Activation function  : Logistic (classification)
Data                 : dvd_wrk
Filter               : training == 1
Response variable    : buy
Level                : yes in buy
Explanatory variables: coupon, purch, last, rnd1, rnd2, rnd3, rnd4, rnd5
Network size         : 8
Parameter decay      : 0.05
Seed                 : 1234
Network              : 8-8-1 with 81 weights
Nr obs               : 4,000
Weights              :
 b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1 i7->h1 i8->h1
  2.09   0.27  -1.14  -1.94   1.97   2.23   0.08   0.23   0.66
 b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2
  0.68  -4.67  -3.01   4.03   0.09  -1.55   0.07  -0.45   0.16
 b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3
  2.55  -3.44   0.89   2.27  -1.61  -1.08   0.76   0.66  -0.14
 b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4
  1.98  -1.97  -1.03  -1.61   2.13   3.26   0.63   1.33   1.09
 b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5
 -0.28  -4.07  -1.04   2.65  -0.45   0.09   0.47   1.22   0.83
 b->h6 i1->h6 i2->h6 i3->h6 i4->h6 i5->h6 i6->h6 i7->h6 i8->h6
  1.08  -3.38   0.69   0.45  -1.25   0.14   1.50   1.57   0.26
 b->h7 i1->h7 i2->h7 i3->h7 i4->h7 i5->h7 i6->h7 i7->h7 i8->h7
  0.04  -1.32  -1.57   1.31   0.11  -0.11   0.21  -0.91  -0.08
 b->h8 i1->h8 i2->h8 i3->h8 i4->h8 i5->h8 i6->h8 i7->h8 i8->h8
 -0.25  -2.73   1.73  -4.36   1.69  -0.67   0.62   3.48  -0.43
 b->o h1->o h2->o h3->o h4->o h5->o h6->o h7->o h8->o
 1.36  4.20  3.30 -3.87 -3.28 -3.73  3.98 -5.34 -1.47
```



Estimate a Decision Tree

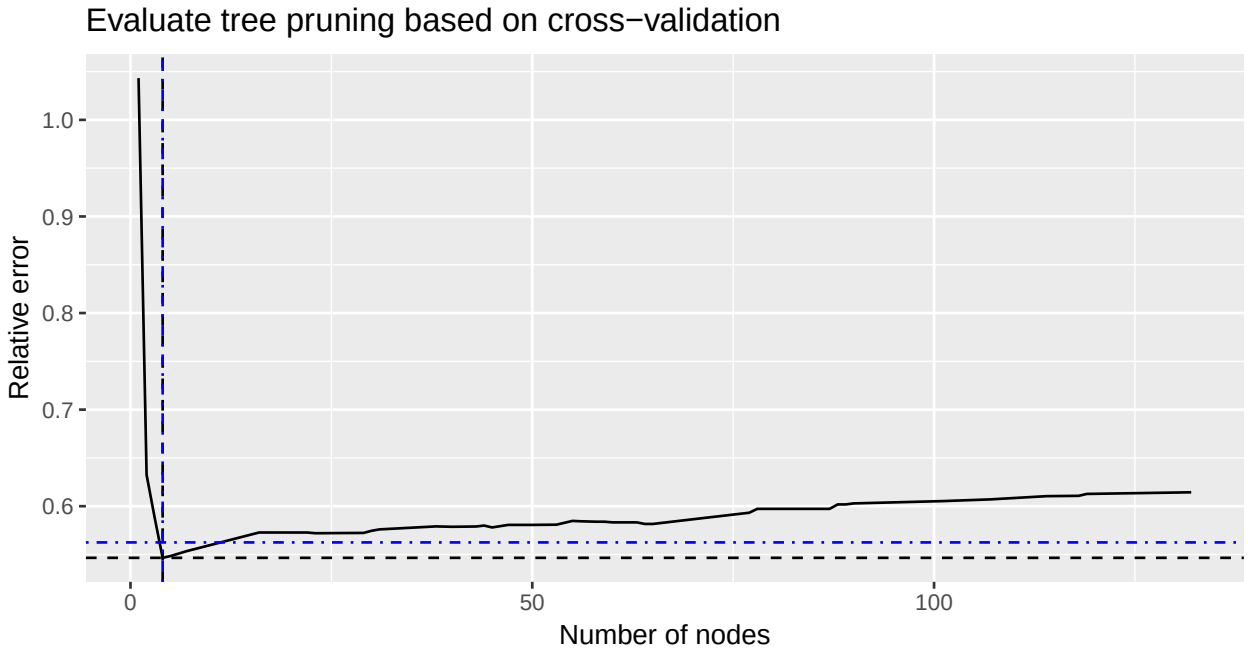
Finally, we'll estimate a Decision Tree. The optimal size of the tree can be established using cross-validation. We'll generate predicted probabilities for an un-pruned tree (i.e., `predict_crtree_large`) and for a pruned tree (i.e., `predict_crtree`). The pruned tree has four nodes.

```
Classification tree
Data                  : dvd_wrk
Filter                : training == 1
Response variable     : buy
Level                 : yes in buy
Explanatory variables: coupon, purch, last, rnd1, rnd2, rnd3, rnd4, rnd5
Complexity parameter : 0.001
Priors                : 0.5
Nr obs                : 4,000
```

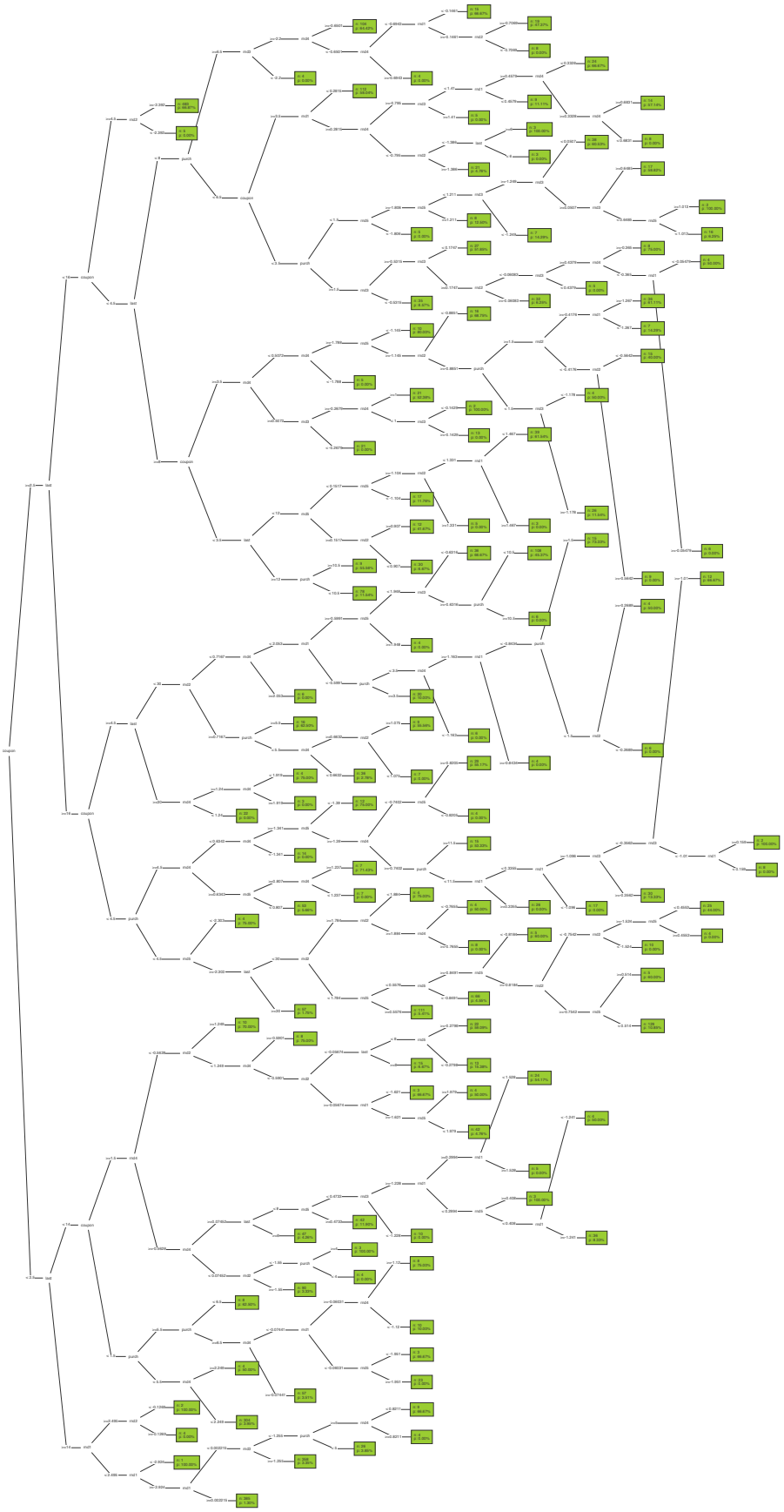
```
Classification and regression trees
Data                  : dvd_wrk
Filter                : training == 1
Response variable     : buy
Level(s)              : yes in buy
Explanatory variables: coupon, purch, last, rnd1, rnd2, rnd3, rnd4, rnd5
Prediction dataset    : dvd_wrk
Rows shown            : 10 of 20,000
```

coupon	purch	last	rnd1	rnd2	rnd3	rnd4	rnd5	Prediction
5	2	5	-0.974	-0.700	-0.562	0.311	0.713	0.669
5	2	33	-0.100	-1.641	0.171	-0.151	-0.202	0.000
4	11	11	-0.111	-0.813	-0.538	0.953	1.023	0.000

```
3     5   25  1.192  0.626 -0.758  0.279 -0.113      0.000
1     1   15 -1.656 -0.835 -0.397  0.023  0.206      0.034
5    10   27 -1.046 -0.252 -0.154 -0.034  0.993      0.100
2     1   11 -1.740  1.249  0.627 -1.363  0.325      0.700
4     6   25  0.513  0.539 -0.749 -0.048 -0.144      0.000
3     9    3 -0.446 -0.565 -1.633  1.925 -0.808      0.644
5     2   27 -1.839  1.001 -0.385 -0.702 -1.950      0.028
```



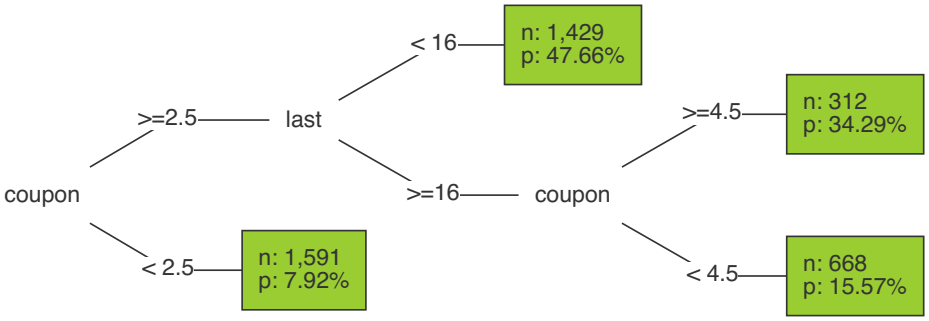
Minimum error achieved with 4 nodes



```
Classification tree
Data                  : dvd_wrk
Filter                : training == 1
Response variable     : buy
Level                 : yes in buy
Explanatory variables: coupon, purch, last, rnd1, rnd2, rnd3, rnd4, rnd5
Complexity parameter : 0.001
Maximum nr. nodes     : 4 out of 132
Priors                : 0.5
Nr obs                : 4,000
```

```
Classification and regression trees
Data                  : dvd_wrk
Filter                : training == 1
Response variable     : buy
Level(s)              : yes in buy
Explanatory variables: coupon, purch, last, rnd1, rnd2, rnd3, rnd4, rnd5
Prediction dataset    : dvd_wrk
Rows shown            : 10 of 20,000
```

coupon	purch	last	rnd1	rnd2	rnd3	rnd4	rnd5	Prediction
5	2	5	-0.974	-0.700	-0.562	0.311	0.713	0.477
5	2	33	-0.100	-1.641	0.171	-0.151	-0.202	0.343
4	11	11	-0.111	-0.813	-0.538	0.953	1.023	0.477
3	5	25	1.192	0.626	-0.758	0.279	-0.113	0.156
1	1	15	-1.656	-0.835	-0.397	0.023	0.206	0.079
5	10	27	-1.046	-0.252	-0.154	-0.034	0.993	0.343
2	1	11	-1.740	1.249	0.627	-1.363	0.325	0.079
4	6	25	0.513	0.539	-0.749	-0.048	-0.144	0.156
3	9	3	-0.446	-0.565	-1.633	1.925	-0.808	0.477
5	2	27	-1.839	1.001	-0.385	-0.702	-1.950	0.343



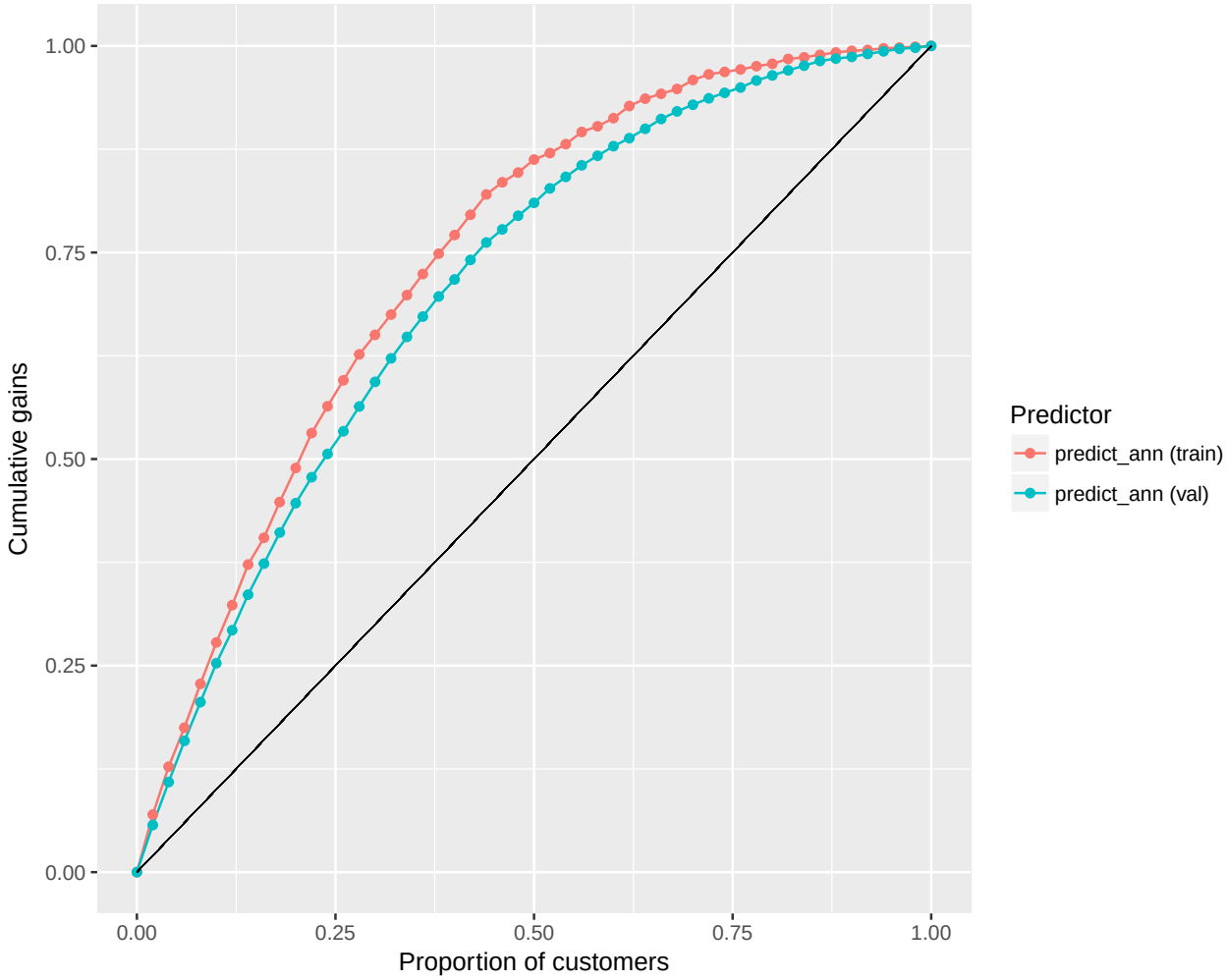
Evaluate model performance

We can evaluate model performance in the training and validation sample using *Model > Evaluate classification*.

First, lets see if there is any evidence of overfitting for the neural network with 8 nodes in the hidden layer. The gains chart shown below indicates that performance in the training dataset is better than in the validation dataset.

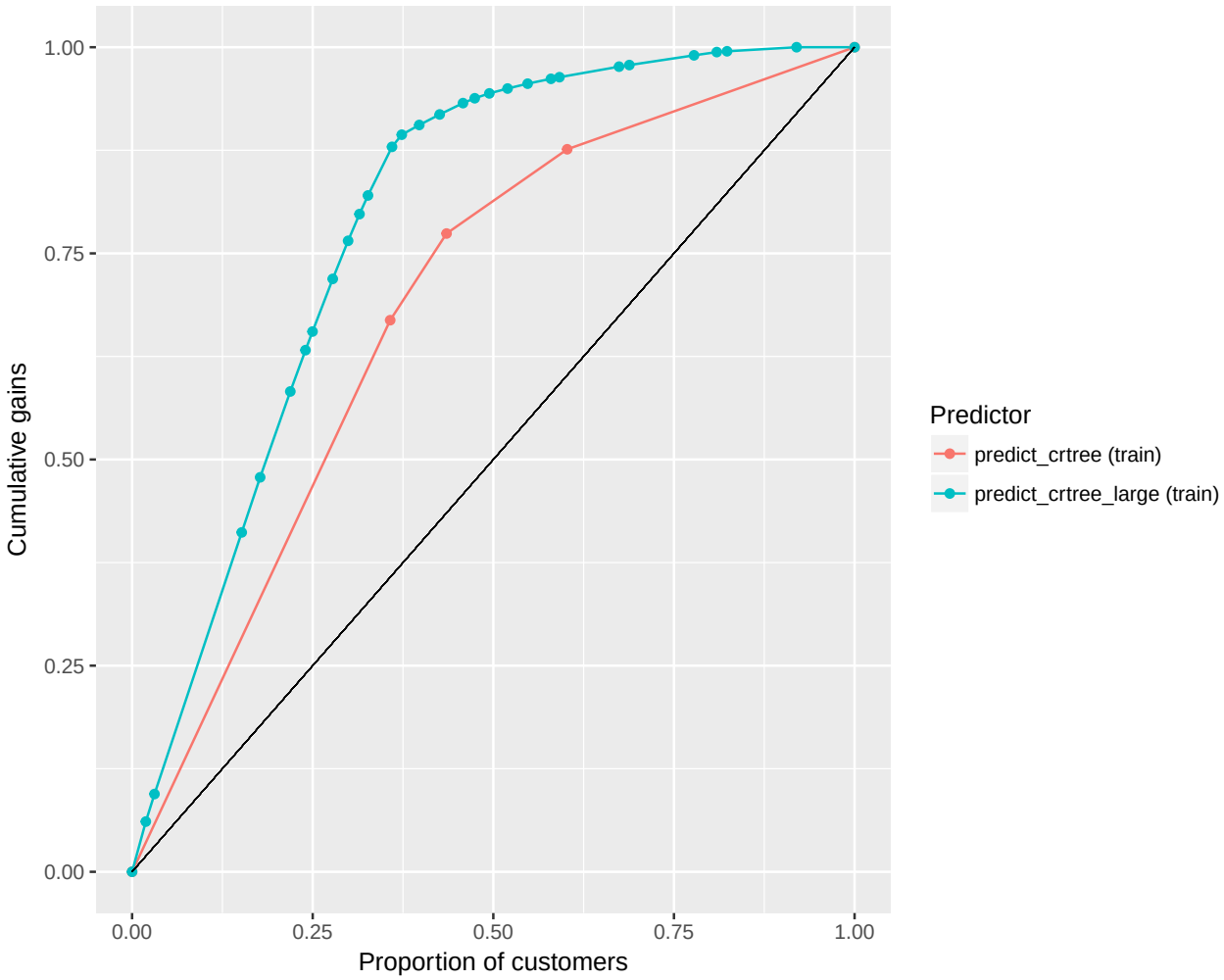
```
Evaluate predictions for binary response models
Data        : dvd_wrk
Filter      : training == 1
Results for : Both
```

```
Predictors  : predict_ann
Response    : buy
Level       : yes in buy
Bins        : 50
Cost:Margin : 1 : 8
```



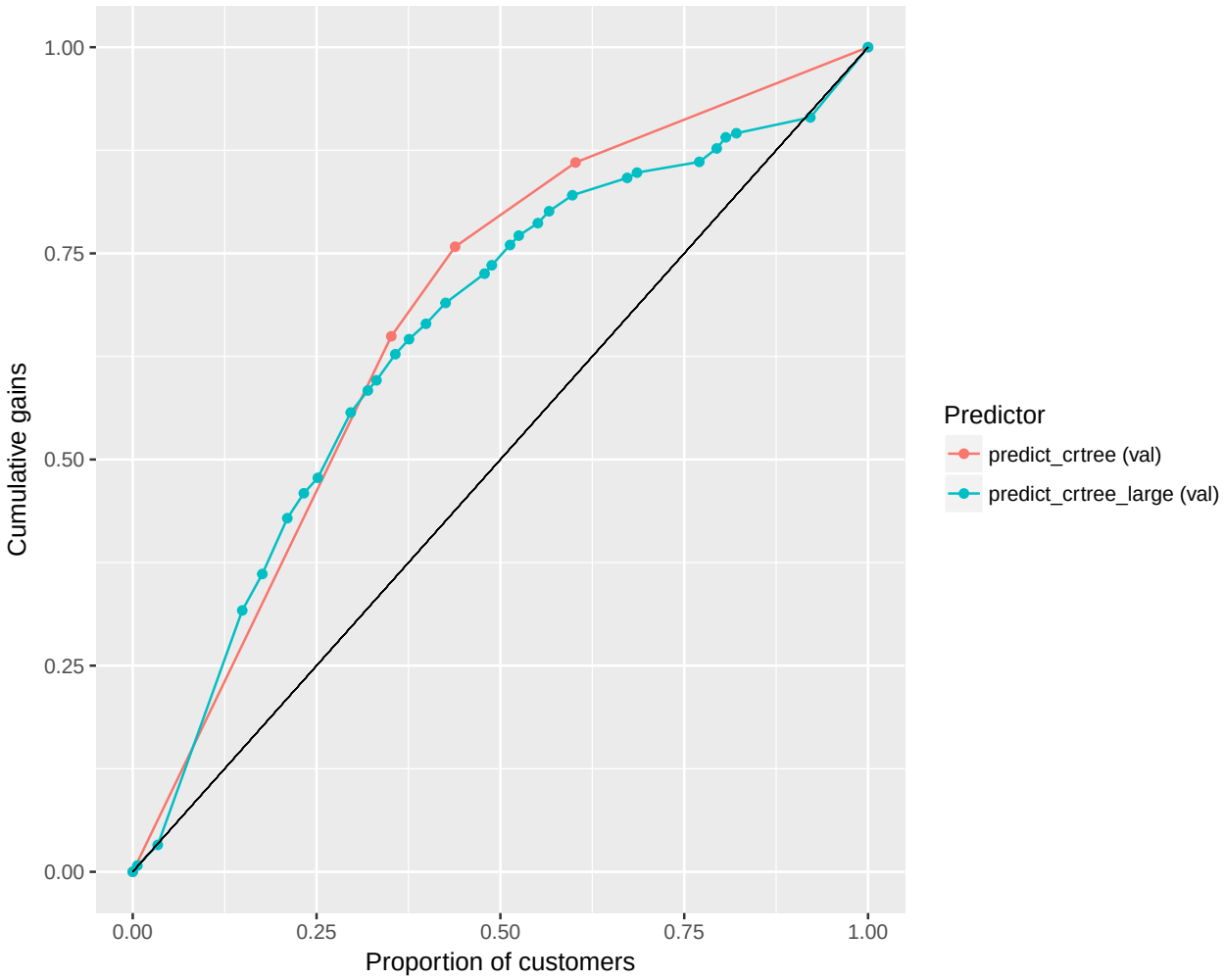
Next, lets evaluate the Decision Tree predictions in training and validation. The first gains chart shows that the large decision tree fits the training data much better compared to the pruned decision tree.

```
Evaluate predictions for binary response models
Data        : dvd_wrk
Filter      : training == 1
Results for : Training
Predictors  : predict_crtree, predict_crtree_large
Response    : buy
Level       : yes in buy
Bins        : 50
Cost:Margin : 1 : 8
```



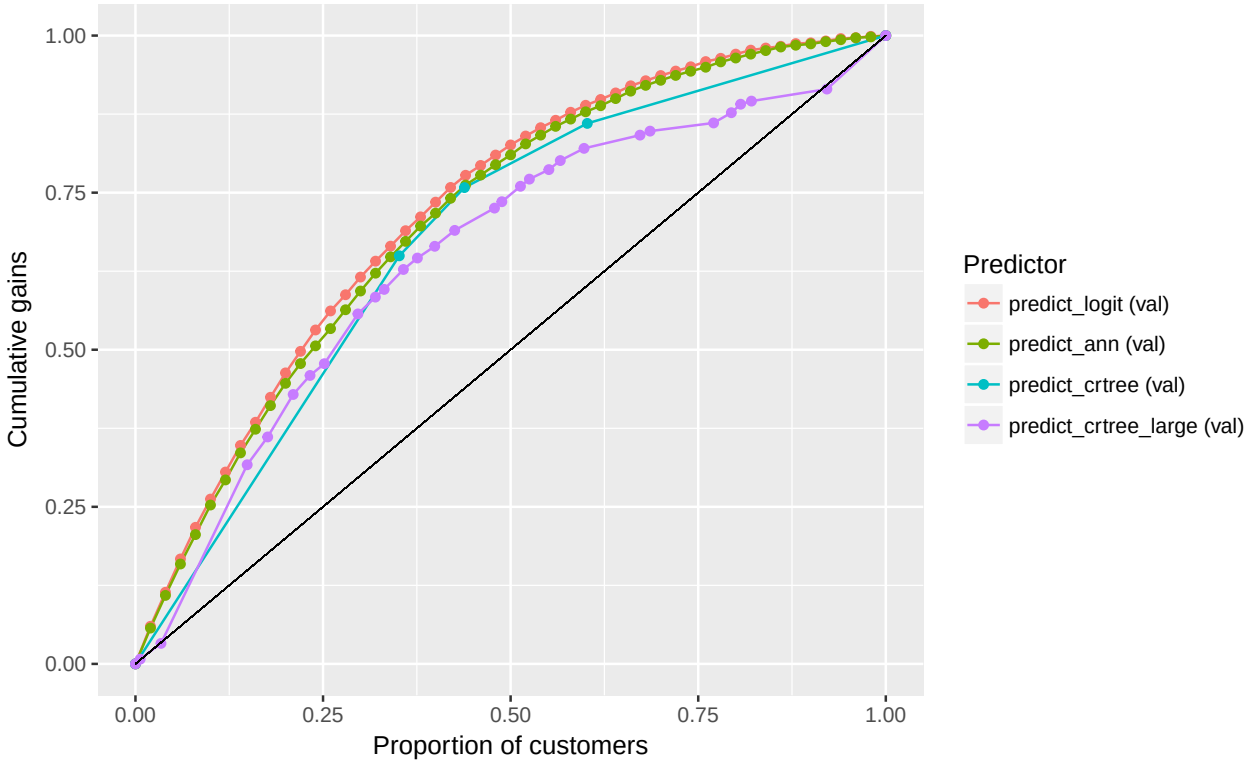
However, when we look at the validation dataset the conclusion is reversed. Now the pruned decision tree fits (slightly) better than the the large (un-pruned) decision tree.

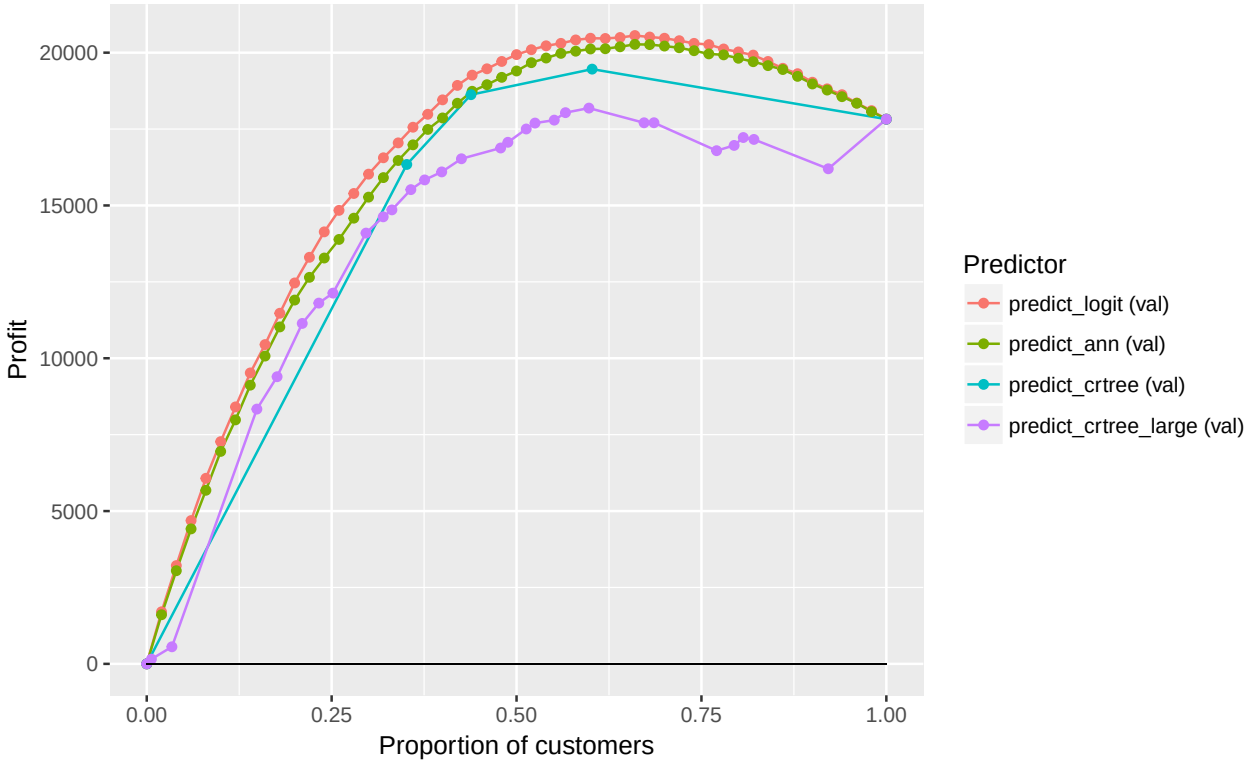
```
Evaluate predictions for binary response models
Data        : dvd_wrk
Filter      : training == 1
Results for : Validation
Predictors  : predict_crtree, predict_crtree_large
Response    : buy
Level       : yes in buy
Bins        : 50
Cost:Margin : 1 : 8
```



Finally, lets compare different models in the validation sample using a gains chart and a profit curve. Both plots suggest that, at least for the `dvd` data, the Logistic Regression performs best.

```
Evaluate predictions for binary response models
Data        : dvd_wrk
Filter      : training == 1
Results for : Validation
Predictors  : predict_logit, predict_ann, predict_crtree, predict_crtree_large
Response    : buy
Level       : yes in buy
Bins        : 50
Cost:Margin : 1 : 8
```





Results from the *Model* > *Evaluate classification* > *Confusion* tab confirm that the Logisitic Regression has the best fit in the validation sample (AUC).

```
Confusion matrix
Data       : dvd_wrk
```

```
Filter     : training == 1
Results for: Validation
Predictors : predict_logit, predict_ann, predict_crtree_large, predict_crtree
Response   : buy
Level      : yes in buy
Cost:Margin: 1 : 8
```

Type	Predictor	TP	FP	TN	FN	total	TPR	TNR	precision
Validation	predict_logit	3805	6148	5624	423	16000	0.900	0.478	0.382
Validation	predict_ann	3727	5945	5827	501	16000	0.882	0.495	0.385
Validation	predict_crtree_large	2750	3362	8410	1478	16000	0.650	0.714	0.450
Validation	predict_crtree	3637	5998	5774	591	16000	0.860	0.490	0.377

Type	Predictor	accuracy	kappa	profit	index	ROME	contact	AUC
Validation	predict_logit	0.589	0.263	20,487	1.000	2.058	0.622	0.801
Validation	predict_ann	0.597	0.267	20,144	0.983	2.083	0.605	0.786
Validation	predict_crtree_large	0.698	0.319	15,888	0.776	2.599	0.382	0.707
Validation	predict_crtree	0.588	0.249	19,461	0.950	2.020	0.602	0.742