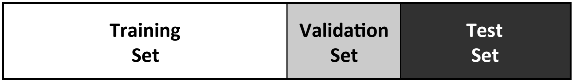
Fundamentals of Machine Learning

Chapter 8: Evaluation

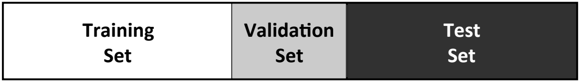
Sections 8.4, 8.5

Designing Evaluation Experiments

Sometimes we use a "validation set" to avoid overfitting during modelling, such as for pruning a decision tree.



(a) A 50:20:30 split



(b) A 40:20:40 split

Figure: **Hold-out sampling** can divide the full data into training, validation, and test sets.

Fundamental Rule: The data used to evaluate a model must be different from the data used to train it.

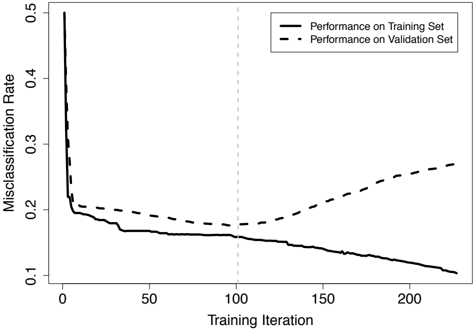


Figure: Using a validation set to avoid overfitting in iterative machine learning algorithms.

(Example: least squares iterations for training a logistic regression model)

10-fold Cross Validation (CV)

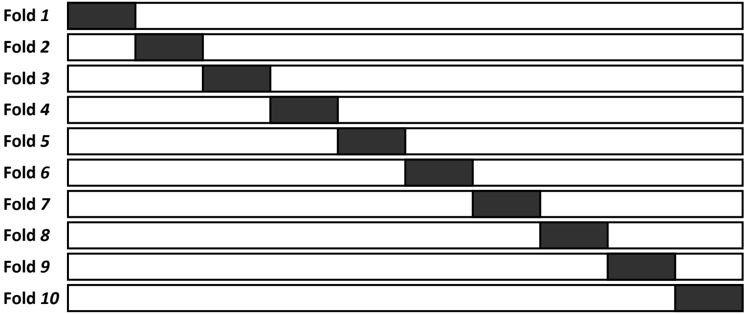


Figure: The division of data during the ***k*-fold cross validation** process. Black rectangles indicate test data, and white spaces indicate training data.

Why k-fold CV: (1) We might not have enough data for hold-out sampling. (2) Reduce effects of a "lucky" split: we happen to put difficult instances in the training data and the easy ones in the test data.

Example: Predict orientation of x-rays

Total of 500 instances, with 100 in each fold

Fold	Confusion Matrix				Class Accuracy
			Prediction		
			'lateral'	*'frontal'*	
1	Target	*'lateral'*	43	9	81%
		'frontal'	10	38	
			Prediction		
			'lateral'	*'frontal'*	
2	Target	*'lateral'*	46	9	88%
		'frontal'	3	42	
			Prediction		
			'lateral'	*'frontal'*	
3	Target	*'lateral'*	51	10	82%
		'frontal'	8	31	
			Prediction		
			'lateral'	*'frontal'*	
4	Target	*'lateral'*	51	8	85%
		'frontal'	7	34	
			Prediction		
			'lateral'	*'frontal'*	
5	Target	*'lateral'*	46	9	84%
		'frontal'	7	38	
			Prediction		
			'lateral'	*'frontal'*	
Overall	Target	*'lateral'*	237	45	84%
		'frontal'	35	183	

Pay attention to the sums

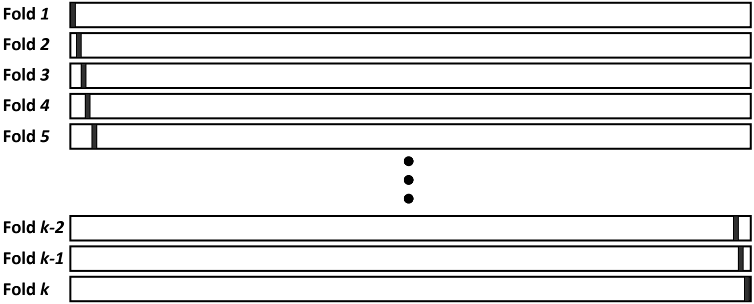


Figure: The division of data during the **leave-one-out cross validation** process. Black rectangles indicate instances in the test set, and white spaces indicate training data.

LOOCV: Extreme form of k-fold CV where k = number of training instances.
We use this method when the amount of data available is too small for k-fold CV.

Performance Measures: Categorical Targets

Remember:

Table: The structure of a confusion matrix.

		Prediction	
		positive	**negative**
Target	**positive**	*TP*	*FN*
	negative	*FP*	*TN*

Table: A confusion matrix for the set of predictions shown in Table 1 [7].

		Prediction	
		'spam'	*'ham'*
Target	*'spam'*	6	3
	'ham'	2	9

$$\mathrm{TPR} = \frac{TP}{(TP + FN)} \tag{1}$$

$$\mathrm{TNR} = \frac{TN}{(TN + FP)} \tag{2}$$

$$\mathrm{FPR} = \frac{FP}{(TN + FP)} \tag{3}$$

$$\mathrm{FNR} = \frac{FN}{(TP + FN)} \tag{4}$$

TPR: True Positive Rate
FNR: False Negative Rate

$$
\begin{aligned}
\text{TPR} &= \tfrac{6}{(6+3)} &&= 0.667 \\
\text{TNR} &= \tfrac{9}{(9+2)} &&= 0.818 \\
\text{FPR} &= \tfrac{2}{(9+2)} &&= 0.182 \\
\text{FNR} &= \tfrac{3}{(6+3)} &&= 0.333
\end{aligned}
$$

TPR + FNR = 1
TNR + FPR = 1

$$\text{precision} = \frac{TP}{(TP + FP)} \tag{5}$$

$$\text{recall} = \frac{TP}{(TP + FN)} \tag{6}$$

Precision: How “precise” are the results? That is, how many of the positives found by the classifier are truly positive?

Recall: How many “recalls”? That is, how many of the true positives can the classifier find, that is “recall”?

Example: Breast cancer dataset
Positive class: cancer
Negative class: healthy

Precision is what percent of the cancer predictions are truly cancers.

Recall is what percent of the cancers did the classifier correctly labeled as cancer.

$$\text{precision} = \frac{6}{(6+2)} = 0.75$$

$$\text{recall} = \frac{6}{(6+3)} = 0.667$$

$$\text{F}_1\text{-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{7}$$

F1 measure: harmonic mean of precision and recall

In general, harmonic mean tends toward the smaller values in a list of numbers and therefore it can be less sensitive to large outliers than the arithmetic mean, which tends toward higher values.

$$\text{F}_1\text{-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{7}$$

$$\begin{aligned}\text{F}_1\text{-measure} &= 2 \times \frac{\left(\frac{6}{(6+2)} \times \frac{6}{(6+3)}\right)}{\left(\frac{6}{(6+2)} + \frac{6}{(6+3)}\right)} \\ &= 0.706\end{aligned}$$

Table: A confusion matrix for a k-NN model trained on a churn prediction problem.

Precision = 1/1

		Prediction	
		'non-churn'	*'churn'*
Target	*'non-churn'*	90	0
	'churn'	9	1

Recall = 1/10

Accuracy: 91%

Table: A confusion matrix for a naive Bayes model trained on a churn prediction problem.

Precision = 8/28

		Prediction	
		'non-churn'	*'churn'*
Target	*'non-churn'*	70	20
	'churn'	2	8

Recall = 8/10

Accuracy: 78%

$$\text{average class accuracy} = \frac{1}{|\textit{levels}(t)|} \sum_{l \in \textit{levels}(t)} \text{recall}_l \quad (8)$$

(using arithmetic mean)

$$\text{average class accuracy}_{\text{HM}} = \frac{1}{\frac{1}{|\mathit{levels}(t)|} \sum_{l \in \mathit{levels}(t)} \frac{1}{\text{recall}_l}} \tag{9}$$

(using harmonic mean)

Average class accuracy using harmonic mean:

k-NN model:

$$\frac{1}{\frac{1}{2}\left(\frac{1}{1.0}+\frac{1}{0.1}\right)}=\frac{1}{5.5}=18.2\%$$

Naive Bayes model:

$$\frac{1}{\frac{1}{2}\left(\frac{1}{0.778}+\frac{1}{0.800}\right)}=\frac{1}{1.268}=78.873\%$$

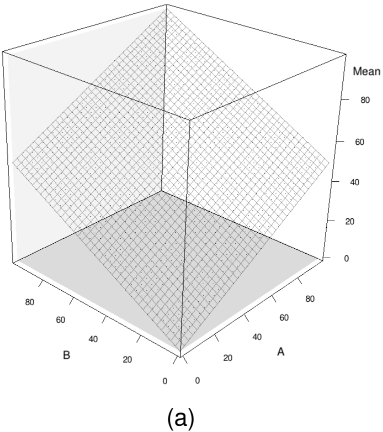


Figure: Surfaces generated by calculating (a) the **arithmetic mean** and (b) the **harmonic mean** of all combinations of features A and B that range from 0 to 100.

- It is not always correct to treat all outcomes equally
- In these cases, it is useful to take into account the cost of the different outcomes when evaluating models

Table: The structure of a **profit matrix**.

		Prediction	
		positive	**negative**
Target	**positive**	TP_{Profit}	FN_{Profit}
	negative	FP_{Profit}	TN_{Profit}

Table: The **profit matrix** for the pay-day loan credit scoring problem.

		Prediction	
		'good'	***'bad'***
Target	***'good'***	140	-140
	'bad'	-700	0

Table: (a) The confusion matrix for a k-NN model trained on the pay-day loan credit scoring problem (average class accuracy$_{HM}$ $= 83.824\%$); (b) the confusion matrix for a decision tree model trained on the pay-day loan credit scoring problem (average class accuracy$_{HM}$ $= 80.761\%$).

(a) k-NN model

		Prediction	
		'good'	*'bad'*
Target	***'good'***	57	3
	'bad'	10	30

(b) decision tree

		Prediction	
		'good'	*'bad'*
Target	***'good'***	43	17
	'bad'	3	37

Table: (a) Overall profit for the *k*-NN model using the profit matrix in Table 4 [25] and the **confusion matrix** in Table 5(a) [26]; (b) overall profit for the decision tree model using the profit matrix in Table 4 [25] and the **confusion matrix** in Table 5(b) [26].

(a) *k*-NN model

		Prediction	
		'good'	*'bad'*
Target	*'good'*	7 980	−420
	'bad'	−7 000	0
	Profit		**560**

(b) decision tree

		Prediction	
		'good'	*'bad'*
Target	*'good'*	6 020	−2 380
	'bad'	−2 100	0
	Profit		**1 540**

Performance Measures: Prediction Scores

- All our classification prediction models return a score which is then thresholded.

Example

$$threshold(score, 0.5) = \begin{cases} positive & \text{if } score \geq 0.5 \\ negative & otherwise \end{cases} \tag{10}$$

Table: A sample test set with model predictions and scores (threshold$=0.5$.

ID	Target	Pred-iction	Score	Out-come
7	ham	ham	0.001	TN
11	ham	ham	0.003	TN
15	ham	ham	0.059	TN
13	ham	ham	0.064	TN
19	ham	ham	0.094	TN
12	spam	ham	0.160	FN
2	spam	ham	0.184	FN
3	ham	ham	0.226	TN
16	ham	ham	0.246	TN
1	spam	ham	0.293	FN

ID	Target	Pred-iction	Score	Out-come
5	ham	ham	0.302	TN
14	ham	ham	0.348	TN
17	ham	spam	0.657	FP
8	spam	spam	0.676	TP
6	spam	spam	0.719	TP
10	spam	spam	0.781	TP
18	spam	spam	0.833	TP
20	ham	spam	0.877	FP
9	spam	spam	0.960	TP
4	spam	spam	0.963	TP

- We have ordered the examples by score so the threshold is apparent in the predictions.
- Note that, in general, instances that actually should get a prediction of *'ham'* generally have a low score, and those that should get a prediction of *'spam'* generally get a high score.

- There are a number of performance measures that use this ability of a model to rank instances that should get predictions of one target level higher than the other, to assess how well the model is performing.
- The basis of most of these approaches is measuring how well the distributions of scores produced by the model for different target levels are separated

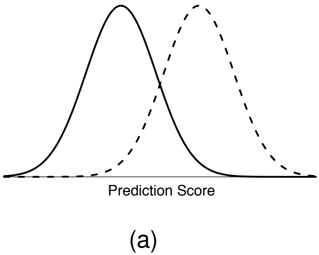
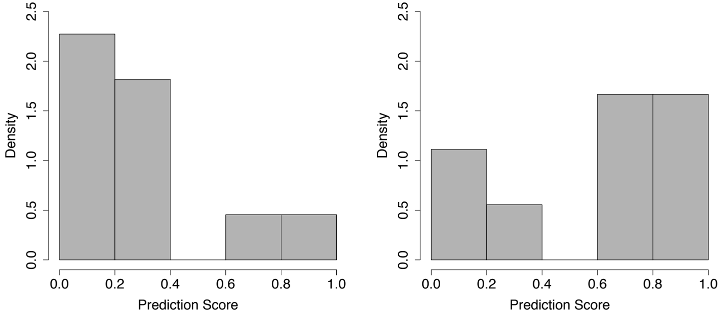


Figure: Prediction score distributions for two different prediction models. The distributions in (a) are much better separated than those in (b).



(a) spam (b) ham

Figure: Prediction score distributions for the (a) *'spam'* and (b) *'ham'* target levels based on the data in Table 7 [30].

- The receiver operating characteristic index (ROC index), which is based on the receiver operating characteristic curve (ROC curve), is a widely used performance measure that is calculated using prediction scores.
- TPR and TNR are intrinsically tied to the threshold used to convert prediction scores into target levels.
- This threshold can be changed, however, which leads to different predictions and a different confusion matrix.

Table: Confusion matrices for the set of predictions shown in Table 7 [30] using (a) a prediction score threshold of 0.75 and (b) a prediction score threshold of 0.25.

(a) Threshold: 0.75

		Prediction *'spam'*	**Prediction** *'ham'*
Target	***'spam'***	**5**	**5**
	'ham'	**1**	10

(b) Threshold: 0.25

		Prediction *'spam'*	**Prediction** *'ham'*
Target	***'spam'***	7	2
	'ham'	4	7

ID	Target	Score	Pred. (0.10)	Pred. (0.25)	Pred. (0.50)	Pred. (0.75)	Pred. (0.90)
7	ham	0.001	ham	ham	ham	ham	ham
11	ham	0.003	ham	ham	ham	ham	ham
15	ham	0.059	ham	ham	ham	ham	ham
13	ham	0.064	ham	ham	ham	ham	ham
19	ham	0.094	ham	ham	ham	ham	ham
12	spam	0.160	spam	ham	ham	ham	ham
2	spam	0.184	spam	ham	ham	ham	ham
3	ham	0.226	spam	ham	ham	ham	ham
16	ham	0.246	spam	ham	ham	ham	ham
1	spam	0.293	spam	spam	ham	ham	ham
5	ham	0.302	spam	spam	ham	ham	ham
14	ham	0.348	spam	spam	ham	ham	ham
17	ham	0.657	spam	spam	spam	ham	ham
8	spam	0.676	spam	spam	spam	ham	ham
6	spam	0.719	spam	spam	spam	ham	ham
10	spam	0.781	spam	spam	spam	spam	ham
18	spam	0.833	spam	spam	spam	spam	ham
20	ham	0.877	spam	spam	spam	spam	ham
9	spam	0.960	spam	spam	spam	spam	spam
4	spam	0.963	spam	spam	spam	spam	spam
Misclassification Rate			0.300	0.300	0.250	0.300	0.350
True Positive Rate (TPR)			1.000	0.778	0.667	0.444	0.222
True Negative rate (TNR)			0.455	0.636	0.818	0.909	1.000
False Positive Rate (FPR)			0.545	0.364	0.182	0.091	0.000
False Negative Rate (FNR)			0.000	0.222	0.333	0.556	0.778

- Note: as the threshold increases TPR decreases and TNR increases (and vice versa).
- Capturing this tradeoff is the basis of the ROC curve.

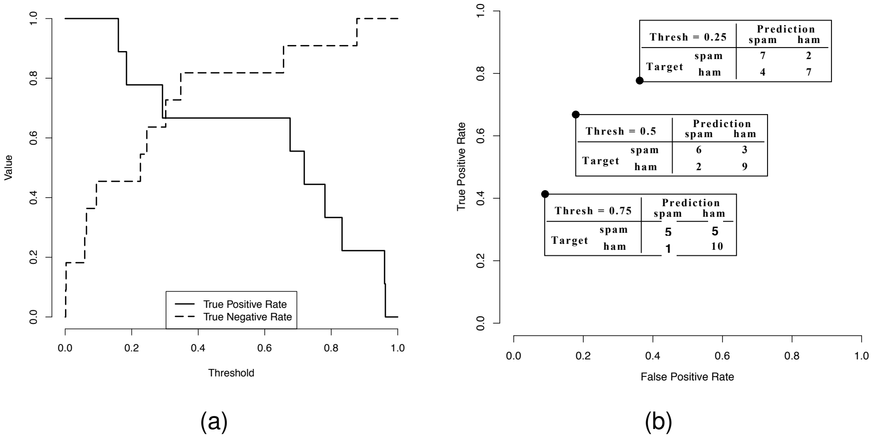


Figure: (a) The changing values of TPR and TNR for the test data shown in Table 36 [37] as the threshold is altered; (b) points in ROC space for thresholds of 0.25, 0.5, and 0.75.

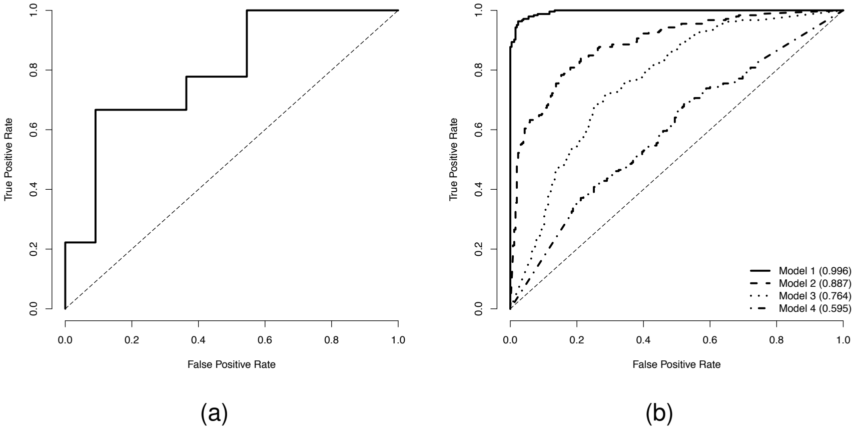


Figure: (a) A complete ROC curve for the email classification example; (b) a selection of ROC curves for different models trained on the same prediction task.

Area under the ROC curve is called "AUC" and it is a fundamental performance metric for binary classification problems. Higher AUC is better.

- We can also calculate a single performance measure from an ROC curve
- The ROC Index measures the area underneath an ROC curve.

ROC index =

$$\sum_{i=2}^{|\mathbf{T}|} \frac{(FPR(\mathbf{T}[i]) - FPR(\mathbf{T}[i-1])) \times (TPR(\mathbf{T}[i]) + TPR(\mathbf{T}[i-1]))}{2} \quad (11)$$

Interpretation of AUC: Say AUC is 97%. This means that if we present two observations to the classifier, one positive one negative, the classifier will find make the correct decision 97% of the time.

Most important property of AUC is that it is robust to the "class imbalance problem" where one class (usually the negative class) dominates the other class, such as internet users who click on a particular ad.

Performance Measures: Multinomial Targets

Table: The structure of a confusion matrix for a multinomial prediction problem with l target levels.

		Prediction					Recall
		level1	***level2***	***level3***	$\cdots$	***levell***	
	level1	-	-	-		-	-
	level2	-	-	-		-	-
Target	***level3***	-	-	-		-	-
	$\vdots$				$\ddots$		$\vdots$
	levell	-	-	-		-	-
	Precision	-	-	-	$\cdots$	-	

$$\text{precision(l)} = \frac{TP(l)}{TP(l) + FP(l)} \tag{20}$$

$$\text{recall(l)} = \frac{TP(l)}{TP(l) + FN(l)} \tag{21}$$

Table: A sample test set with model predictions for a bacterial species identification problem.

ID	Target	Prediction	ID	Target	Prediction
1	durionis	fructosus	16	ficulneus	ficulneus
2	ficulneus	fructosus	17	ficulneus	ficulneus
3	fructosus	fructosus	18	fructosus	fructosus
4	ficulneus	ficulneus	19	durionis	durionis
5	durionis	durionis	20	fructosus	fructosus
6	pseudo.	pseudo.	21	fructosus	fructosus
7	durionis	fructosus	22	durionis	durionis
8	ficulneus	ficulneus	23	fructosus	fructosus
9	pseudo.	pseudo.	24	pseudo.	fructosus
10	pseudo.	fructosus	25	durionis	durionis
11	fructosus	fructosus	26	pseudo.	pseudo.
12	ficulneus	ficulneus	27	fructosus	fructosus
13	durionis	durionis	28	ficulneus	ficulneus
14	fructosus	fructosus	29	fructosus	fructosus
15	fructosus	ficulneus	30	fructosus	fructosus

Table: A confusion matrix for a model trained on the bacterial species identification problem.

		Prediction				Recall
		'durionis'	***'ficulneus'***	***'fructosus'***	***'pseudo.'***	
Target	***'durionis'***	5	0	2	0	0.714
	'ficulneus'	0	6	1	0	0.857
	'fructosus'	0	1	10	0	0.909
	'pseudo.'	0	0	2	3	0.600
	Precision	1.000	0.857	0.667	1.000	

- The average class $\text{accuracy}_{\text{HM}}$ for this problem is:

$$\frac{1}{\frac{1}{4}\left(\frac{1}{0.714}+\frac{1}{0.857}+\frac{1}{0.909}+\frac{1}{0.600}\right)}=\frac{1}{1.333}=75.000\%$$

Performance Measures: Continuous Targets

$$\text{sum of squared errors} = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2 \quad (22)$$

ti is the true target feature value for the i-th instance, M(di) is the model's prediction.

$$\text{mean squared error} = \frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n} \quad (23)$$

$$\text{root mean squared error} = \sqrt{\frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n}} \quad (24)$$

A nice feature of RMSE is that its value is in the same unit as the target value, e.g., meters.

$$\text{mean absolute error} = \frac{\sum_{i=1}^{n} abs(t_i - \mathbb{M}(\mathbf{d}_i))}{n} \quad (25)$$

		Linear Regression		k-NN	
ID	Target	Prediction	Error	Prediction	Error
1	10.502	10.730	0.228	12.240	1.738
2	18.990	17.578	-1.412	21.000	2.010
3	20.000	21.760	1.760	16.973	-3.027
4	6.883	7.001	0.118	7.543	0.660
5	5.351	5.244	-0.107	8.383	3.032
6	11.120	10.842	-0.278	10.228	-0.892
7	11.420	10.913	-0.507	12.921	1.500
8	4.836	7.401	2.565	7.588	2.752
9	8.177	8.227	0.050	9.277	1.100
10	19.009	16.667	-2.341	21.000	1.991
11	13.282	14.424	1.142	15.496	2.214
12	8.689	9.874	1.185	5.724	-2.965
13	18.050	19.503	1.453	16.449	-1.601
14	5.388	7.020	1.632	6.640	1.252
15	10.646	10.358	-0.288	5.840	-4.805
16	19.612	16.219	-3.393	18.965	-0.646
17	10.576	10.680	0.104	8.941	-1.634
18	12.934	14.337	1.403	12.484	-0.451
19	10.492	10.366	-0.126	13.021	2.529
20	13.439	14.035	0.596	10.920	-2.519
21	9.849	9.821	-0.029	9.920	0.071
22	18.045	16.639	-1.406	18.526	0.482
23	6.413	7.225	0.813	7.719	1.307
24	9.522	9.565	0.043	8.934	-0.588
25	12.083	13.048	0.965	11.241	-0.842
26	10.104	10.085	-0.020	10.010	-0.095
27	8.924	9.048	0.124	8.157	-0.767
28	10.636	10.876	0.239	13.409	2.773
29	5.457	4.080	-1.376	9.684	4.228
30	3.538	7.090	3.551	5.553	2.014
	MSE		**1.905**		**4.394**
	RMSE		**1.380**		**2.096**
	MAE		**0.975**		**1.750**
	R^2		**0.889**		**0.776**

R-squared measure is domain-independent.
It compares performance against an imaginary model
that always predicts the "average value", denoted by t-bar:

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}} \tag{26}$$

$$\text{total sum of squares} = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2 \tag{27}$$

Interpretation: R-squared times 100 is the percentage of variation in the target feature that is explained by the descriptive features in the model.

Evaluating Models after Deployment

To monitor the on-going performance of a model, we need a signal that indicates that something has changed. There are three sources from which we can extract such a signal:

1. The performance of the model measured using appropriate performance measures
2. The distributions of the outputs of a model
3. The distributions of the descriptive features in query instances presented to the model

- The simplest way to get a signal that concept drift has occurred is to repeatedly evaluate models with the same performance measures used to evaluate them before deployment.
- We can calculate performance measures for a deployed model and compare these to the performance achieved in evaluations before the model was deployed.
- If the performance changes significantly, this is a strong indication that **concept drift** has occurred and that the model has gone stale.

Moral of the story: Models tend to "wear out" over time and they will need to be re-trained.

- Although monitoring changes in the performance of a model is the easiest way to tell whether it has gone stale, this method makes the rather large assumption that the correct target feature value for a query instance will be made available shortly after the query has been presented to a deployed model.

Example: In credit loan scoring, whether a customer is "good" is understood only after years of on-time payments.

- An alternative to using changing model performance is to use changes in the distribution of model outputs as a signal for concept drift.

$$\text{stability index} = \sum_{l \in levels(t)} \left(\left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} - \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \times log_e \left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} / \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \right) \tag{28}$$

In general,

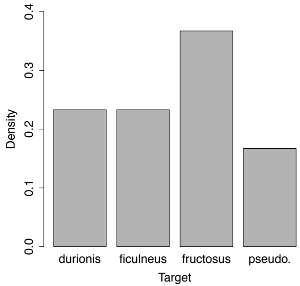
- stability index < 0.1, then the distribution of the newly collected test set is broadly similar to the distribution in the original test set.
- stability index is between 0.1 and 0.25, then some change has occurred and further investigation may be useful.
- stability index > 0.25 suggests that a significant change has occurred and corrective action is required.

Table: Calculating the **stability index** for the bacterial species identification problem given new test data for two periods after model deployment. The frequency and percentage of each target level are shown for the original test set and for two samples collected after deployment. The column marked SI_t shows the different parts of the stability index sum based on Equation (28)[72].

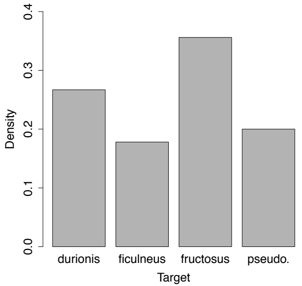
	Original		New Sample 1			New Sample 2		
Target	Count	%	Count	%	SI_t	Count	%	SI_t
'durionis'	7	0.233	12	0.267	0.004	12	0.200	0.005
'ficulneus'	7	0.233	8	0.178	0.015	9	0.150	0.037
'fructosus'	11	0.367	16	0.356	0.000	14	0.233	0.060
'pseudo.'	5	0.167	9	0.200	0.006	25	0.417	0.229
Sum	30		45		**0.026**	60		**0.331**

Stability index calculations for New Sample 1:

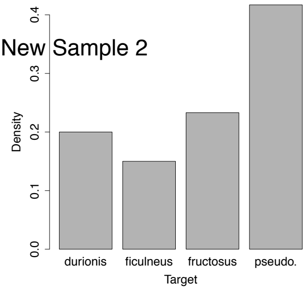
$$
\begin{aligned}
\text{stability index} \quad = \quad & \left(\frac{7}{30} - \frac{12}{45}\right) \times log_e\left(\frac{7}{30} / \frac{12}{45}\right) \\
& + \left(\frac{7}{30} - \frac{8}{45}\right) \times log_e\left(\frac{7}{30} / \frac{8}{45}\right) \\
& + \left(\frac{11}{30} - \frac{16}{45}\right) \times log_e\left(\frac{11}{30} / \frac{16}{45}\right) \\
& + \left(\frac{5}{30} - \frac{9}{45}\right) \times log_e\left(\frac{5}{30} / \frac{9}{45}\right) \\
= \quad & 0.026
\end{aligned}
$$



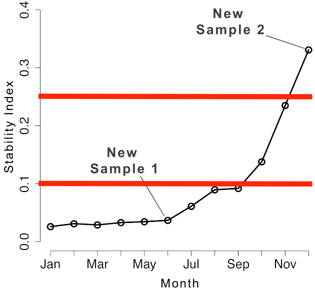
(a) Original



(b) New Sample 1



(c) New Sample 2



- We use control groups not to evaluate the predictive power of the models themselves, but rather to evaluate how good they are at helping with the business problem when they are deployed.

Doing a good job with the predictions is not enough.
The real question is:
Do these predictions translate to Dollars?

Table: The number of customers who left the mobile phone network operator each week during the comparative experiment from both the control group (random selection) and the treatment group (model selection).

Week	Control Group (Random Selection)	Treatment Group (Model Selection)
1	21	23
2	18	15
3	28	18
4	19	20
5	18	15
6	17	17
7	23	18
8	24	20
9	19	18
10	20	19
11	18	13
12	21	16
Mean	20.500	17.667
Std. Dev.	3.177	2.708

- These figures show that, on average, fewer customers churn when the churn prediction model is used to select which customers to call.

Here, 1000 random customers were selected for the Control Group for each week from a pool of 400,000 customers.
Likewise, the Treatment Group contains 1000 customers with the highest churn risk scores for each week.
Each group of customers were contacted by phone by the customer service centre and churners were recorded.