Evaluation

Chapter 8

Péter Molnár[1]

J. Mack Robinson College of Business
Georgia State University

MSA8150 – Spring 2016

[1] Original slides from John Kelleher, Brian Mac Namee, and Aoife D'Arcy

- The most important part of the design of an evaluation experiment for a predictive model is ensuring that the data used to evaluate the model is not the same as the data used to train the model.
- The purpose of evaluation is threefold:
  1. to determine which model is the most suitable for a task
  2. to estimate how the model will perform
  3. to convince users that the model will meet their needs

Misclassification Rate on Hold-out Test Set

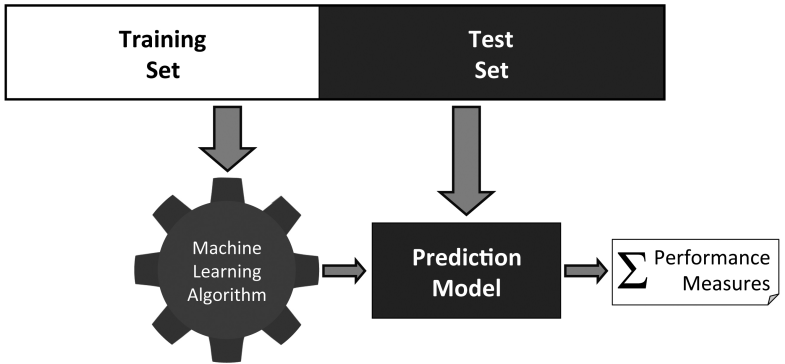


Figure: The process of building and evaluating a model using a **hold-out test set**.

Table: A sample test set with model predictions.

ID	Target	Pred.	Outcome
1	spam	ham	FN
2	spam	ham	FN
3	ham	ham	TN
4	spam	spam	TP
5	ham	ham	TN
6	spam	spam	TP
7	ham	ham	TN
8	spam	spam	TP
9	spam	spam	TP
10	spam	spam	TP

ID	Target	Pred.	Outcome
11	ham	ham	TN
12	spam	ham	FN
13	ham	ham	TN
14	ham	ham	TN
15	ham	ham	TN
16	ham	ham	TN
17	ham	spam	FP
18	spam	spam	TP
19	ham	ham	TN
20	ham	spam	FP

$$\text{misclassification rate} = \frac{\text{number incorrect predictions}}{\text{total predictions}} \tag{1}$$

$$\text{misclassification rate} = \frac{(2+3)}{(6+9+2+3)} = 0.25$$

- For binary prediction problems there are 4 possible outcomes:
  1. True Positive (TP)
  2. True Negative (TN)
  3. False Positive (FP)
  4. False Negative (FN)

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 [6].

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

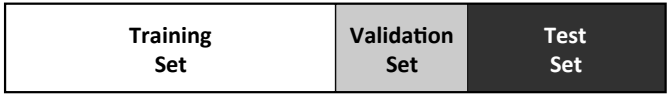
$$\text{misclassification accuracy} = \frac{(FP + FN)}{(TP + TN + FP + FN)} \tag{2}$$

$$\text{misclassification accuracy} = \frac{(2 + 3)}{(6 + 9 + 2 + 3)} = 0.25$$

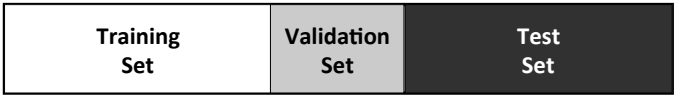
$$\text{classification accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{3}$$

$$\text{classification accuracy} = \frac{(6 + 9)}{(6 + 9 + 2 + 3)} = 0.75$$

Designing Evaluation Experiments



(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.

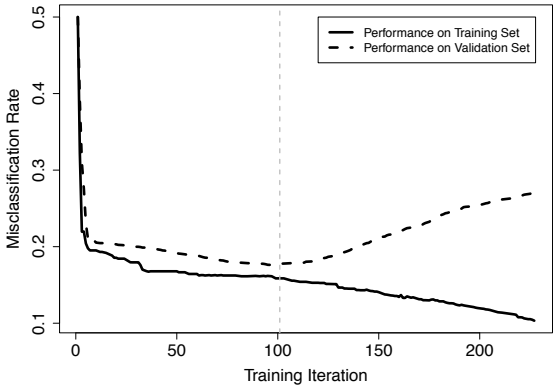


Figure: Using a validation set to avoid overfitting in iterative machine learning algorithms.

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

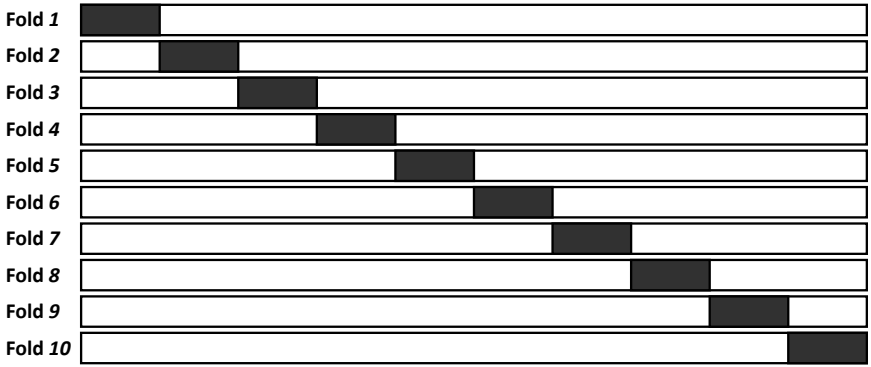


Figure: The division of data during the ***k*-fold cross validation** process. Black rectangles indicate test data, and white spaces indicate training data.

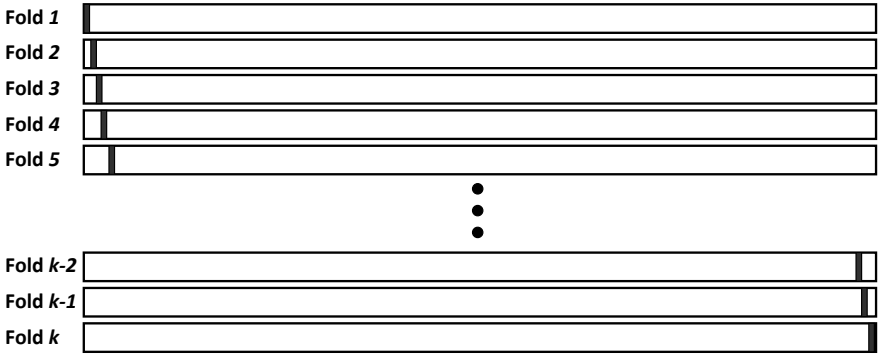


Figure: The division of data during the **leave-one-out cross validation** process. Black rectangles indicate instances in the test set, and white spaces indicate training data.

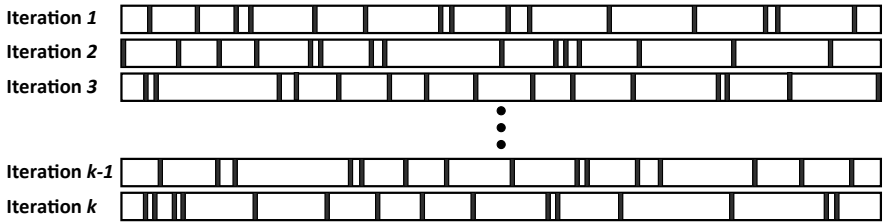


Figure: The division of data during the $\epsilon 0$ bootstrap process. Black rectangles indicate test data, and white spaces indicate training data.

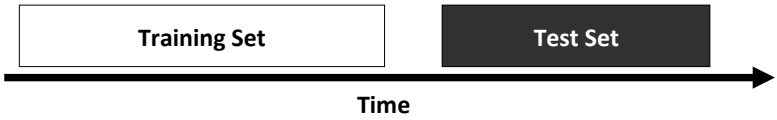


Figure: The **out-of-time sampling** process.

Categorical Targets

$$\text{TPR} = \frac{TP}{(TP + FN)} \tag{4}$$

$$\text{TNR} = \frac{TN}{(TN + FP)} \tag{5}$$

$$\text{FPR} = \frac{FP}{(TN + FP)} \tag{6}$$

$$\text{FNR} = \frac{FN}{(TP + FN)} \tag{7}$$

Confusion Matrix-based Performance Measures

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

$$\text{precision} = \frac{TP}{(TP + FP)} \tag{8}$$

$$\text{recall} = \frac{TP}{(TP + FN)} \tag{9}$$

Precision, Recall and F_1 Measure

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

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

$$\text{F}_1\text{-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{10}$$

$$\text{F}_1\text{-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \tag{10}$$

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

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

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

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

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

Average Class Accuracy

$$\text{average class accuracy} = \frac{1}{|\mathit{levels}(t)|} \sum_{l \in \mathit{levels}(t)} \text{recall}_l \tag{11}$$

$$\text{average class accuracy}_{\text{HM}} = \frac{1}{\frac{1}{|\mathit{levels}(t)|}\sum_{l \in \mathit{levels}(t)} \frac{1}{\text{recall}_l}} \tag{12}$$

Average Class Accuracy

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

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

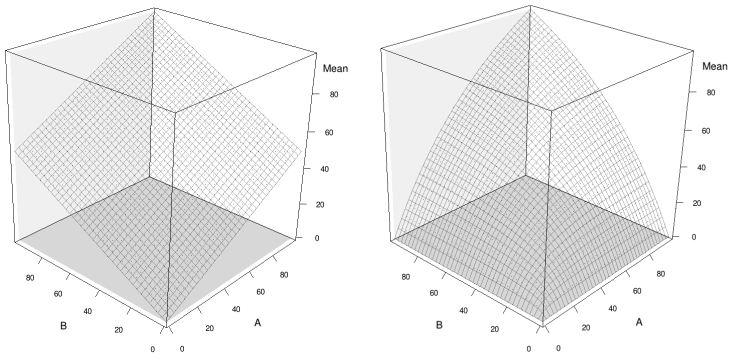


Figure: Surfaces generated by calculating (a) the **arithmetic mean** and (b) the **harmonic mean** of all combinations of features A and B that range from 0 to 100.

		True condition			
	Total population	Condition positive	Condition negative	Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive** (Type I error)	Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$	False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$
	Predicted condition negative	**False negative** (Type II error)	**True negative**	False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$	Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$
	Accuracy (ACC) $= \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	True positive rate (TPR), Sensitivity, Recall $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$
		False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) $= \frac{\text{FNR}}{\text{TNR}}$	

- 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 7 [34] and the **confusion matrix** in Table 8(a) [35]; (b) overall profit for the decision tree model using the profit matrix in Table 7 [34] and the **confusion matrix** in Table 8(b) [35].

(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

$$threshold(score, 0.5) = \begin{cases} positive & \text{if } score \geq 0.5 \\ negative & otherwise \end{cases} \tag{13}$$

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

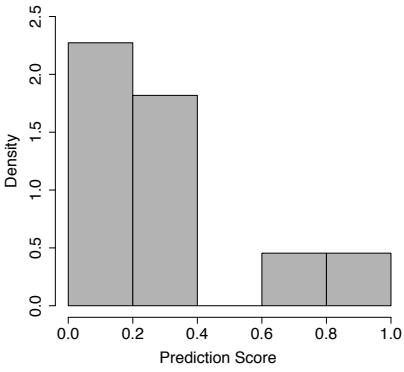
ID	Target	Pred-iction	Score	Out-come
7	ham	ham	0.001	TN
11	ham	ham	0.003	TN
15	ham	ham	0.059	TN
13	ham	ham	0.064	TN
19	ham	ham	0.094	TN
12	spam	ham	0.160	FN
2	spam	ham	0.184	FN
3	ham	ham	0.226	TN
16	ham	ham	0.246	TN
1	spam	ham	0.293	FN

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

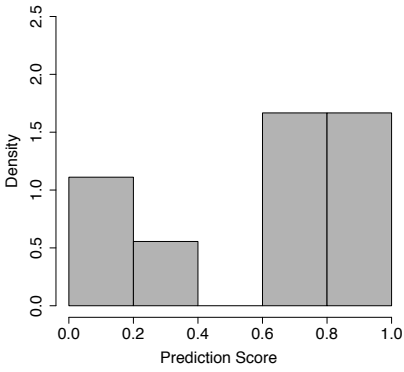
- We have ordered the examples by score so the threshold is apparent in the predictions.
- Note that, in general, instances that actually should get a prediction of *'ham'* generally have a low score, and those that should get a prediction of *'spam'* generally get a high score.

- 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). Far from reality, though.



(a) spam



(b) ham

Figure: Prediction score distributions for the (a) *'spam'* and (b) *'ham'* target levels based on the data in Table 10 [39].

- 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 10 [39] using (a) a prediction score threshold of 0.75 and (b) a prediction score threshold of 0.25.

(a) Threshold: 0.75

		Prediction	
		'spam'	***'ham'***
Target	***'spam'***	4	4
	'ham'	2	10

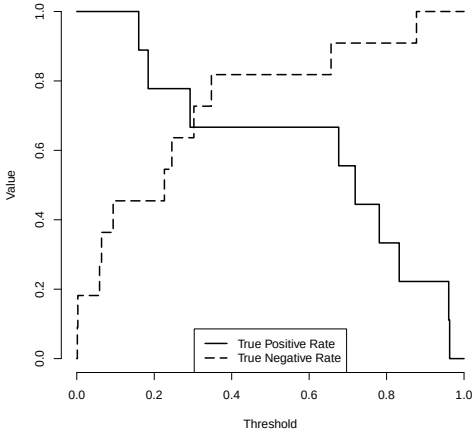
(b) Threshold: 0.25

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

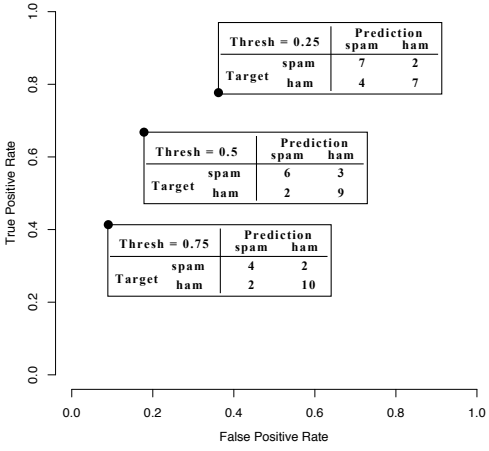
ID	Target	Score	Pred. (0.10)	Pred. (0.25)	Pred. (0.50)	Pred. (0.75)	Pred. (0.90)
7	ham	0.001	ham	ham	ham	ham	ham
11	ham	0.003	ham	ham	ham	ham	ham
15	ham	0.059	ham	ham	ham	ham	ham
13	ham	0.064	ham	ham	ham	ham	ham
19	ham	0.094	ham	ham	ham	ham	ham
12	spam	0.160	spam	ham	ham	ham	ham
2	spam	0.184	spam	ham	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.

Receiver Operating Characteristic Curves



(a)



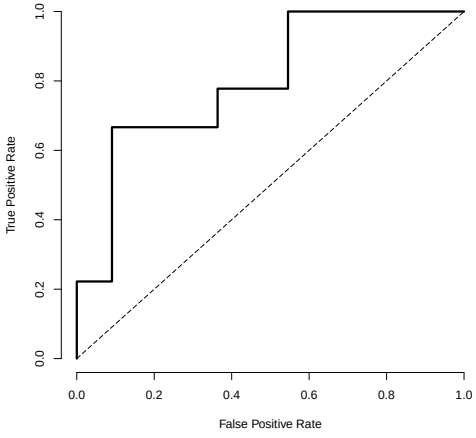
Thresh = 0.25		Prediction spam	Prediction ham
Target	spam	7	2
	ham	4	7

Thresh = 0.5		Prediction spam	Prediction ham
Target	spam	6	3
	ham	2	9

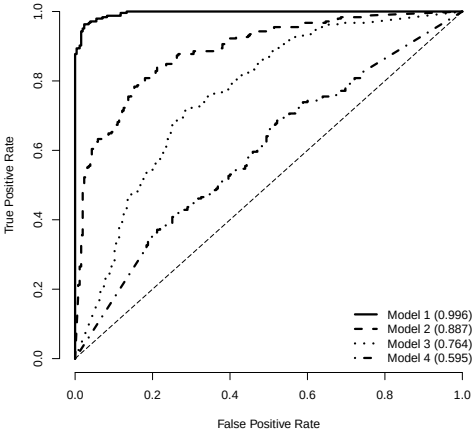
Thresh = 0.75		Prediction spam	Prediction ham
Target	spam	4	2
	ham	2	10

(b)

Receiver Operating Characteristic Curves



(a)



(b)

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

$$\text{ROC index} = \sum_{i=2}^{|\mathbf{T}|} \frac{(FPR(\mathbf{T}[i]) - FPR(\mathbf{T}[i-1])) \times (TPR(\mathbf{T}[i]) + TPR(\mathbf{T}[i-1]))}{2} \tag{14}$$

- The **Gini coefficient** is a linear rescaling of the ROC index

$$\text{Gini coefficient} = (2 \times \text{ROC index}) - 1 \tag{15}$$

- The **Kolmogorov-Smirnov statistic** (K-S statistic) is another performance measure that captures the separation between the distribution of prediction scores for the different target levels in a classification problem.

- To calculate the K-S statistic, we first determine the cumulative probability distributions of the prediction scores for the positive and negative target levels:

$$CP(positive, ps) = \frac{\text{num positive test instances with score} \leq ps}{\text{num positive test instances}} \tag{16}$$

$$CP(negative, ps) = \frac{\text{num negative test instances with score} \leq ps}{\text{num negative test instances}} \tag{17}$$

Kolmogorov-Smirnov Statistic

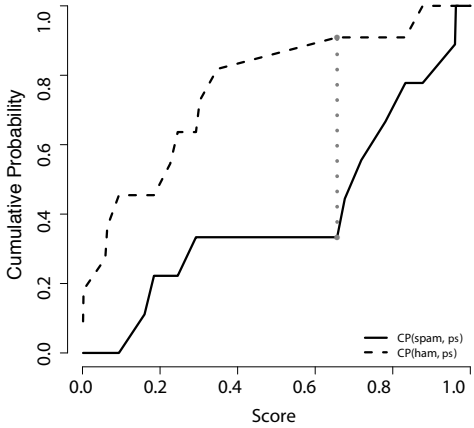


Figure: The K-S chart for the email classification predictions shown in Table 10 [39].

- The K-S statistic is calculated by determining the maximum difference between the cumulative probability distributions for the positive and negative target levels.

$$\text{K-S} = \max_{ps} \left(CP(positive, ps) - CP(negative, ps) \right) \tag{18}$$

ID	Prediction Score	Positive (*'spam'*) Cumulative Count	Negative (*'ham'*) Cumulative Count	Positive (*'spam'*) Cumulative Probability	Negative (*'ham'*) Cumulative Probability	Distance
7	0.001	0	1	0.000	0.091	0.091
11	0.003	0	2	0.000	0.182	0.182
15	0.059	0	3	0.000	0.273	0.273
13	0.064	0	4	0.000	0.364	0.364
19	0.094	0	5	0.000	0.455	0.455
12	0.160	1	5	0.111	0.455	0.343
2	0.184	2	5	0.222	0.455	0.232
3	0.226	2	6	0.222	0.545	0.323
16	0.246	2	7	0.222	0.636	0.414
1	0.293	3	7	0.333	0.636	0.303
5	0.302	3	8	0.333	0.727	0.394
14	0.348	3	9	0.333	0.818	0.485
17	**0.657**	**3**	**10**	**0.333**	**0.909**	**0.576***
8	0.676	4	10	0.444	0.909	0.465
6	0.719	5	10	0.556	0.909	0.354
10	0.781	6	10	0.667	0.909	0.242
18	0.833	7	10	0.778	0.909	0.131
20	0.877	7	11	0.778	1.000	0.222
9	0.960	8	11	0.889	1.000	0.111
4	0.963	9	11	1.000	1.000	0.000

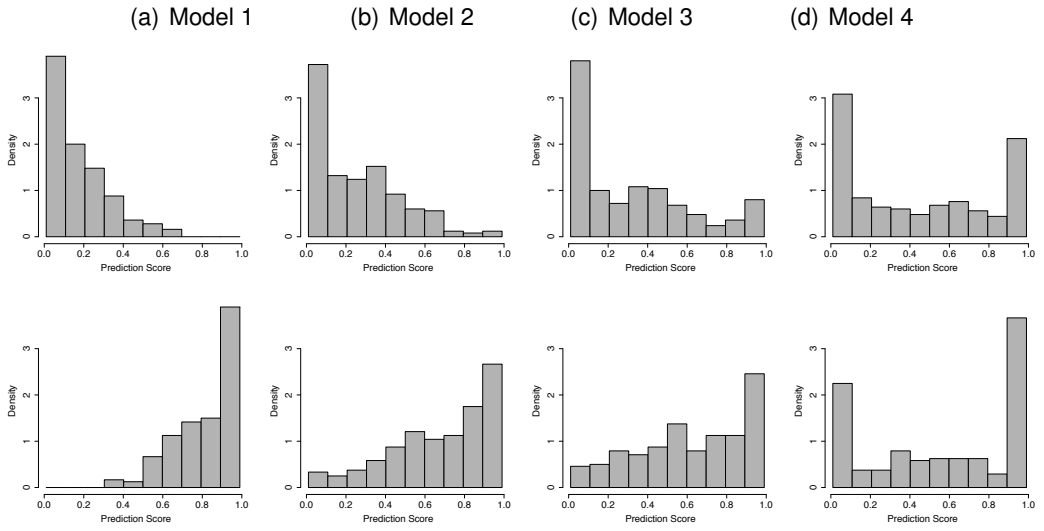


Figure: A series of charts for different model performance on email classification test set. From top to bottom: *'ham'* scores and *'spam'* scores predicted by the model.

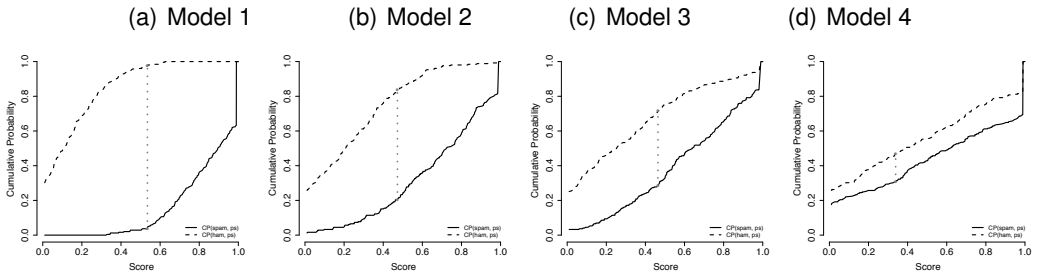


Figure: Corresponding K-S charts. Solid line CP of *'spam'*, dashed line CP of *'ham'*

Table: The test set with model predictions and scores from Table 10 [39] extended to include deciles.

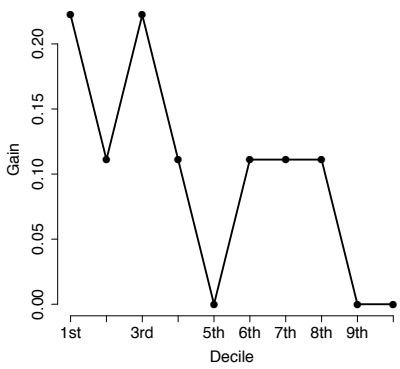
Decile	ID	Target	Prediction	Score	Outcome
1^{st}	9	spam	spam	0.960	TP
	4	spam	spam	0.963	TP
2^{nd}	18	spam	spam	0.833	TP
	20	ham	spam	0.877	FP
3^{rd}	6	spam	spam	0.719	TP
	10	spam	spam	0.781	TP
4^{th}	17	ham	spam	0.657	FP
	8	spam	spam	0.676	TP
5^{th}	5	ham	ham	0.302	TN
	14	ham	ham	0.348	TN
6^{th}	16	ham	ham	0.246	TN
	1	spam	ham	0.293	FN
7^{th}	2	spam	ham	0.184	FN
	3	ham	ham	0.226	TN
8^{th}	19	ham	ham	0.094	TN
	12	spam	ham	0.160	FN
9^{th}	15	ham	ham	0.059	TN
	13	ham	ham	0.064	TN
10^{th}	7	ham	ham	0.001	TN
	11	ham	ham	0.003	TN

$$\text{Gain}(dec) = \frac{\text{num positive test instances in decile } dec}{\text{num positive test instances}} \tag{19}$$

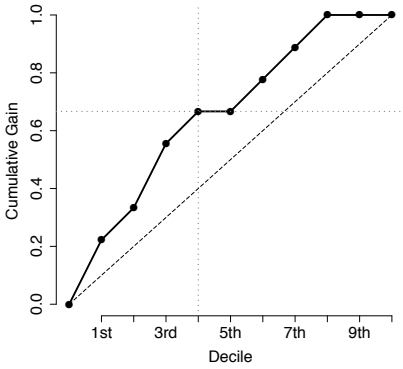
Table: Tabulating the workings required to calculate **gain**, **cumulative gain**, **lift**, and **cumulative lift** for the data given in Table 10 [39].

Decile	Positive (*'spam'*) Count	Negative (*'ham'*) Count	Gain	Cum. Gain	Lift	Cum. Lift
1^{st}	2	0	0.222	0.222	2.222	2.222
2^{nd}	1	1	0.111	0.333	1.111	1.667
3^{rd}	2	0	0.222	0.556	2.222	1.852
4^{th}	1	1	0.111	0.667	1.111	1.667
5^{th}	0	2	0.000	0.667	0.000	1.333
6^{th}	1	1	0.111	0.778	1.111	1.296
7^{th}	1	1	0.111	0.889	1.111	1.270
8^{th}	1	1	0.111	1.000	1.111	1.250
9^{th}	0	2	0.000	1.000	0.000	1.111
10^{th}	0	2	0.000	1.000	0.000	1.000

$$\text{Cumulative gain}(dec) = \frac{\text{num positive test instances in all deciles up to } dec}{\text{num positive test instances}} \tag{20}$$



(a)



(b)

Figure: The (a) **gain** and (b) **cumulative gain** at each decile for the email predictions given in Table 10 [39].

$$\text{Lift}(dec) = \frac{\text{\% of positive test instances in decile } dec}{\text{\% of positive test instances}} \tag{21}$$

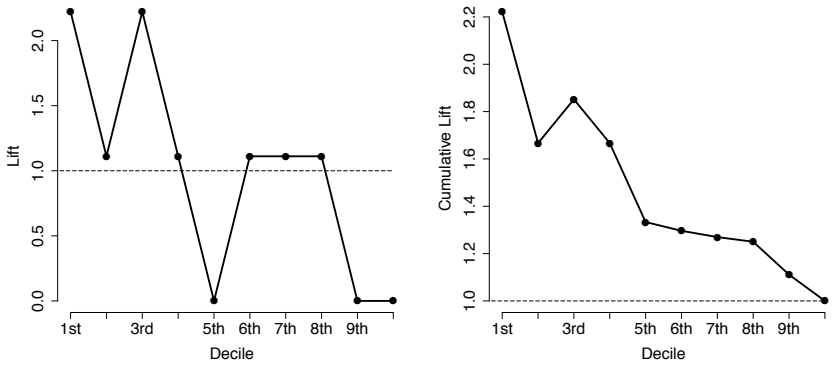


Figure: The (a) **lift** and (b) **cumulative lift** at each decile for the email predictions given in Table 10 [39].

$$\text{Cumulative lift}(dec) = \frac{\%\text{ of positive instances in all deciles up to } dec}{\%\text{ of positive test instances}} \tag{22}$$

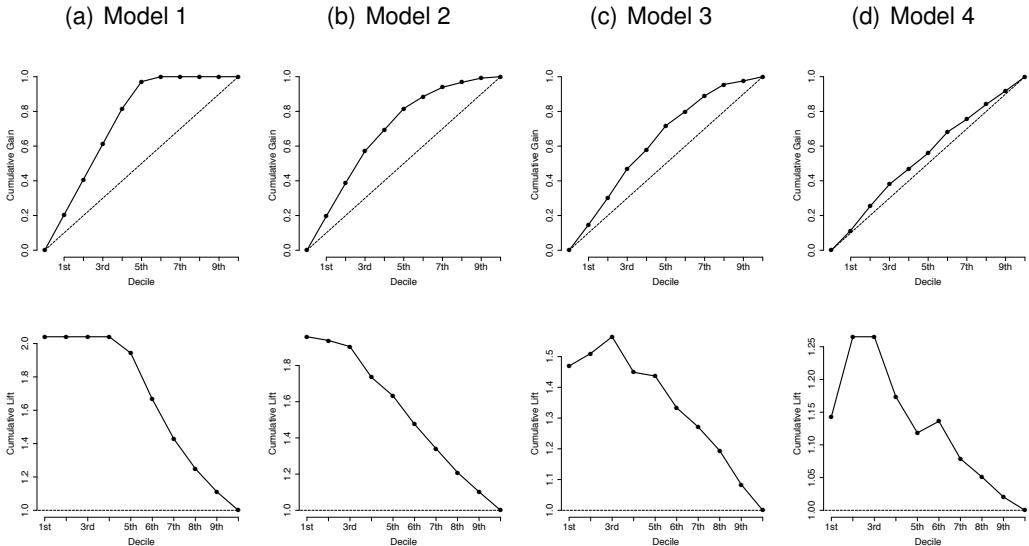


Figure: Cumulative gain, and **cumulative lift** charts for email classification test set.

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

$$\text{precision(l)} = \frac{TP(l)}{TP(l) + FP(l)} \tag{23}$$

$$\text{recall(l)} = \frac{TP(l)}{TP(l) + FN(l)} \tag{24}$$

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 accuracy$_{\text{HM}}$ for this problem is:

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

Continuous Targets

$$\text{sum of squared errors} = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2 \tag{25}$$

$$\text{mean squared error} = \frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n} \tag{26}$$

$$\text{root mean squared error} = \sqrt{\frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n}} \tag{27}$$

$$\text{mean absolute error} = \frac{\sum_{i=1}^{n} abs(t_i - \mathbb{M}(\mathbf{d}_i))}{n} \tag{28}$$

		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
⋮	⋮	⋮	⋮	⋮	⋮
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**

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}} \tag{29}$$

$$\text{total sum of squares} = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2 \tag{30}$$

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.

- 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.

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

$$\text{stability index} = \sum_{l \in \mathit{levels}(t)} \left(\left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} - \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \times \mathit{log}_e \left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} / \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \right) \tag{31}$$

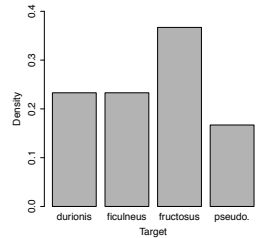
In general,

- 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.

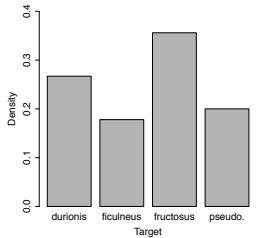
Table: Calculating the **stability index** for the bacterial species identification problem given new test data for two periods after model deployment. The frequency and percentage of each target level are shown for the original test set and for two samples collected after deployment. The column marked SI_t shows the different parts of the stability index sum based on Equation (31)[82].

	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**

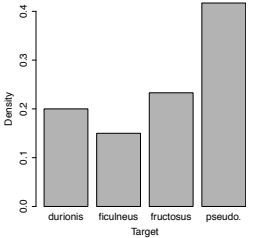
$$
\begin{aligned}
\text{stability index} &= \left(\frac{7}{30} - \frac{12}{45}\right) \times log_e\left(\frac{7}{30} / \frac{12}{45}\right) \\
&\quad + \left(\frac{7}{30} - \frac{8}{45}\right) \times log_e\left(\frac{7}{30} / \frac{8}{45}\right) \\
&\quad + \left(\frac{11}{30} - \frac{16}{45}\right)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



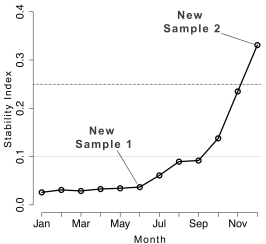
(a) Original



(b) New Sample 1



(c) New Sample 2



(d) Monitoring Over Time

Figure: The distributions of predictions made by a model trained for the bacterial species identification problem for (a) the original evaluation test set and for (b) and (c) two periods of time after model deployment; (d) shows how the stability index should be tracked over time to monitor for concept drift.

- In the same way we can compare the distributions of model outputs between the time that the model was built and after deployment, we can also make the same type of comparison for the distributions of the descriptive features used by the model.
- We can use any appropriate measure that captures the difference between two different distributions for this, including the stability index, the χ^2 **statistic**, and the K-S statistic.

- There is, however, a challenge here, as usually, there are a large number of descriptive features for which measures need to be calculated and tracked.
- Furthermore, it is unlikely that a change in the distribution of just one descriptive feature in a multi-feature model will have a large impact on model performance.
- For this reason, unless a model uses a very small number of descriptive features (generally fewer than 10), we do not recommend this approach.

- 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.

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.

Summary