Chapter 8: Model Evaluation

MATH2319

- **Golden Rule of Model Evaluation:** The data used to evaluate a model must be different from the data used to train it.

- 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

Standard Approach: Measuring Misclassification Rate on a Hold-out Test Set

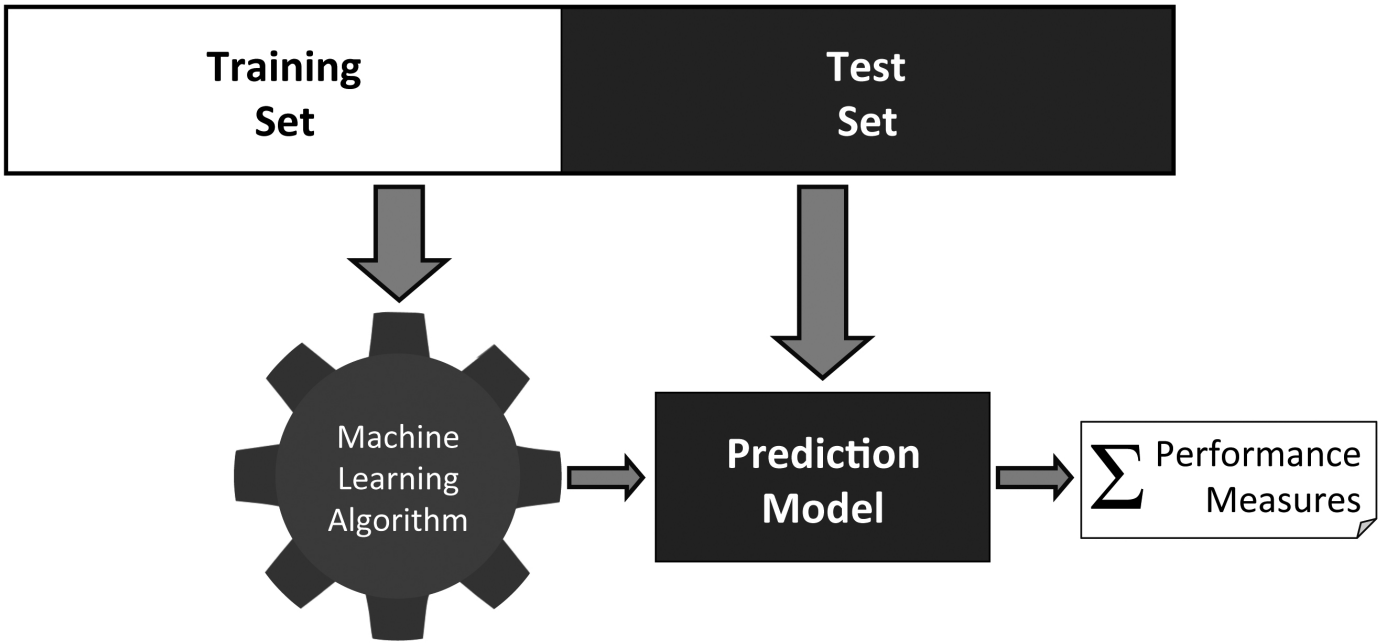


Figure: The process of building and evaluating a model using a **hold-out test set**.

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

- 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 above Table.

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

$$\text{error (misclassification) rate} = \frac{\text{number of incorrect predictions}}{\text{total predictions}}$$

$$\text{error rate} = \frac{(FP + FN)}{(TP + TN + FP + FN)}$$

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

$$\text{accuracy rate} = \frac{\text{number of correct predictions}}{\text{total predictions}}$$

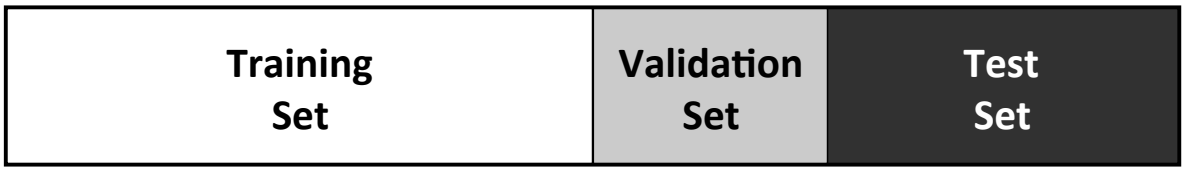
$$\text{accuracy rate} = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

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

Model Evaluation: Hold-out vs. K-Fold Cross Validation

Hold-out Sampling

- We can use a **validation set** for finding optimal hyper-parameter values for an algorithm, such as
  - number of neighbours in KNN
  - tree depth in Decision Trees
- The idea is that the algorithm **never** sees the **test set**, neither while training nor while hyper-parameter fine-tuning.



(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.

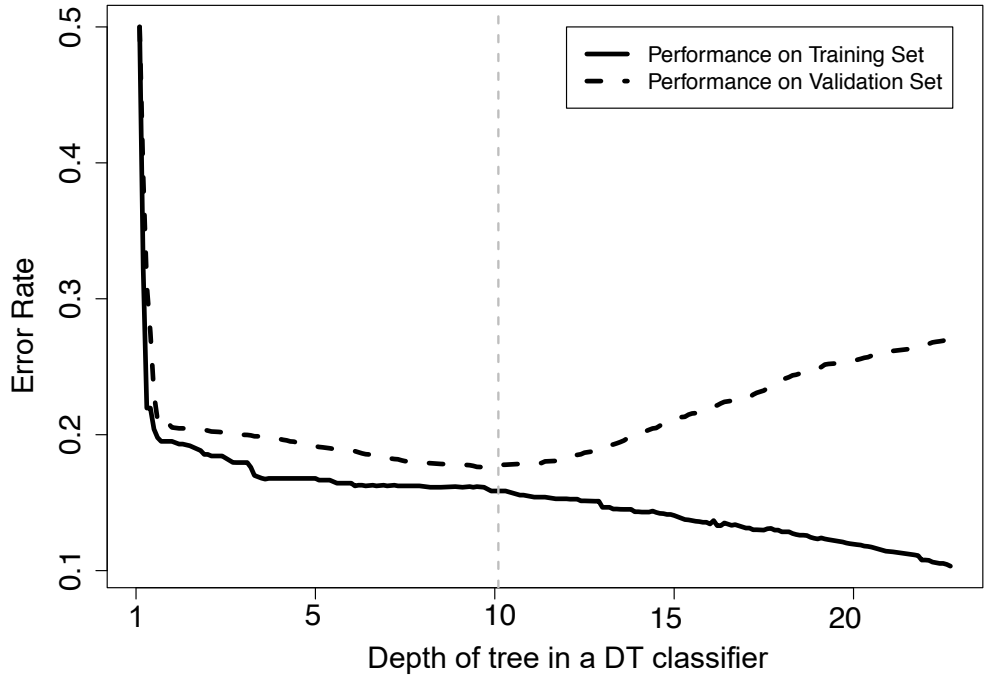


Figure: Example: Using a validation set to avoid overfitting in decision trees.

k-Fold Cross Validation

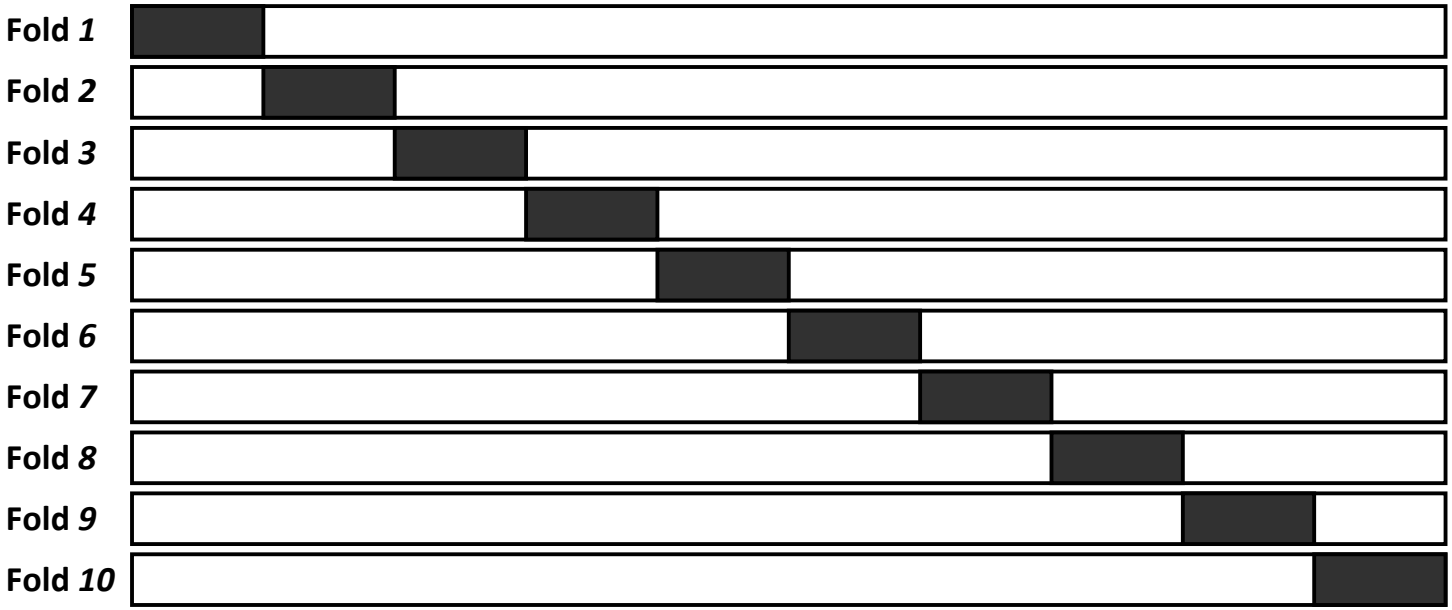


Figure: The division of data during the ***10*-fold cross validation (CV)** process. Black rectangles indicate test data, and white spaces indicate training data.

- Similar to using a validation set, we can also use k-fold CV for the following tasks:
  - finding optimal hyper-parameter values for an algorithm.
  - model performance evaluation.
- Why k-fold CV:
  - We might not have enough data for hold-out sampling (train/ validation/ test sets).
  - Reduce effects of a “lucky” split: we might be putting difficult instances in the training set and the easy ones in the test set.
- Common values for k are 5 and 10.
- If execution time is an issue, some people will use $k = 3$.

5-Fold CV Example:

- Predict orientation of x-rays: lateral or frontal (binary classification).
- Total of 500 instances with 100 in each fold.
- Mean CV accuracy is computed as the average accuracy across the 5 folds.

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	

Performance Measures: Categorical Targets

Confusion Matrix-based Performance Measures

Remember:

Table: The structure of a confusion matrix.

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

- TPR: True Positive Rate
- TNR: True Negative Rate

$$\text{TPR} = \frac{TP}{TP + FN}$$
$$\text{FNR} = 1 \text{ - TPR}$$
$$\text{TNR} = \frac{TN}{TN + FP}$$
$$\text{FPR} = 1 \text{ - TNR}$$

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

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

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

- TPR + FNR = 1
- TNR + FPR = 1

Precision, Recall and F1-Measure

- **Precision:** How "precise" are the results? That is, how many of the positives found by the classifier are truly positive?
- **Recall (i.e., TPR):** 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 labelled as cancer.

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

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

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

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

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

$$\text{recall} = \frac{TP}{(TP + FN)} = \frac{6}{(6 + 3)} = 0.67$$

- The positive class is “spam”.
- Precision is calculated across the **column** of the positive class.
- Recall is calculated across the **row** of the positive class.

- **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.

$$\text{F1-measure} = \frac{1}{\frac{1}{2}\left(\frac{1}{\textit{precision}} + \frac{1}{\textit{recall}}\right)}$$

$$\text{F1-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})}$$

$$\begin{aligned}\text{F1-measure} &= 2 \times \frac{(0.75 \times 0.67)}{(0.75 + 0.67)} \\ &= 0.706\end{aligned}$$

Table: A confusion matrix for a k-NN model trained on a churn prediction problem. The positive class is "churn".

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

- Precision: 1/1 (computed using the second column)
- Recall: 1/10 (computed using the second row)
- Accuracy: 91/100

Table: A confusion matrix for a naive Bayes model trained on a churn prediction problem. The positive class is “churn”.

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

- Precision: 8/28 (computed using the second column)
- Recall: 8/10 (computed using the second row)
- Accuracy: 78/100

Average Class Accuracy - Arithmetic:

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

Average Class Accuracy - Harmonic:

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

Average Class Accuracy - Harmonic for the KNN Model:

$$\frac{1}{\frac{1}{2}\left(\frac{1}{90/90}+\frac{1}{1/10}\right)}=\frac{1}{5.5}=18.2\%$$

Average Class Accuracy - Harmonic for the naive Bayes Model:

$$\frac{1}{\frac{1}{2}\left(\frac{1}{70/90}+\frac{1}{8/10}\right)}=\frac{1}{1.268}=78.873\%$$

Measuring Profit and Loss

- 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 (obtained by element-wise multiplication of the profit and confusion matrices); (b) overall profit for the decision tree model.

(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**

- k-NN has higher accuracy performance, yet the decision tree yields higher profits due to the asymmetric profit structure in this particular problem.

Performance Measures: Prediction Scores

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

Example

$$threshold(score, 0.5) = \begin{cases} positive & \text{if } score \geq 0.5 \\ negative & otherwise \end{cases} \tag{1}$$

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

ID	Target	Prediction	Score	Outcome
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

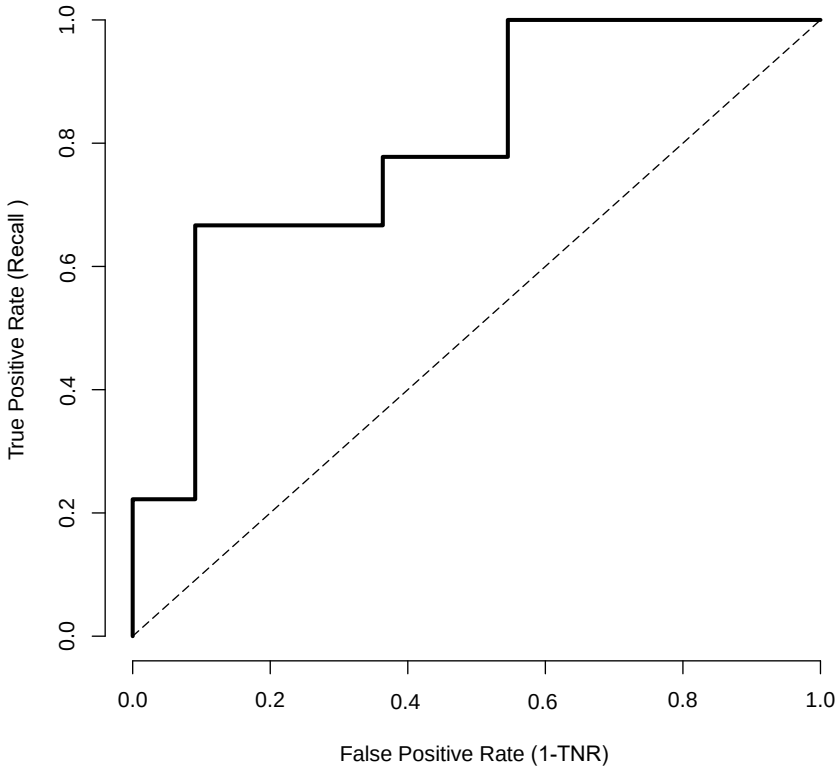
ID	Target	Prediction	Score	Outcome
5	ham	ham	0.302	TN
14	ham	ham	0.348	TN
17	ham	spam	0.657	FP
8	spam	spam	0.676	TP
6	spam	spam	0.719	TP
10	spam	spam	0.781	TP
18	spam	spam	0.833	TP
20	ham	spam	0.877	FP
9	spam	spam	0.960	TP
4	spam	spam	0.963	TP

- Above the threshold, the prediction becomes the positive class.
- In this table, we have ordered the examples by score so the threshold is apparent in the predictions.
- In general, true *'ham'* instances have a low score, and true *'spam'* instances have a high score.

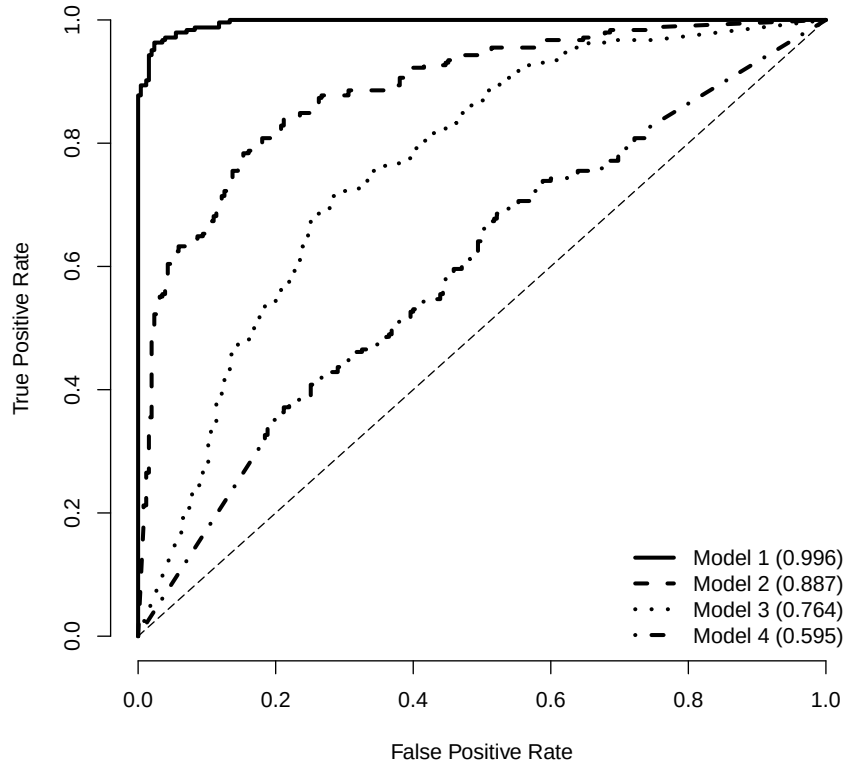
Receiver Operating Characteristic (ROC) Curves

- The receiver operating characteristic index (ROC index) is a widely used performance measure that is calculated using prediction scores.
- TPR (true positive rate, a.k.a. Recall) and FPR (false positive rate) depend on the threshold used to convert prediction scores into target levels.
- This threshold can be varied, which leads to different predictions and different confusion matrices.

- As the threshold is decreased, TPR (Recall) increases (we get more true positives) as well as FPR (we get more false positives).
- The ROC curve captures this relationship between TPR (Recall) and FPR: it shows true positive rates vs. false positive rates as the threshold is gradually decreased from 1 to 0.
- In an ROC curve,
  - the $y-$axis is **TPR (Recall)**
  - the $x-$axis is **FPR**
- Thus, the curve always starts at $(0.0, 0.0)$ (threshold = 1, no positive predictions) and ends at $(1.0, 1.0)$ (threshold = 0, all predictions are positive).



(a)



(b)

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.

AUC Scores

- We can also calculate a single performance measure from an ROC curve:
- The AUC metric denotes the “area underneath an ROC curve”.
- The higher the AUC, the better the performance.
- An AUC score can be computed only for binary classification problems!
- AUC also depends on which class you define as “positive”.

AUC Score Interpretation

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

$$\text{precision(l)} = \frac{TP(l)}{TP(l) + FP(l)} \tag{2}$$

$$\text{recall(l)} = \frac{TP(l)}{TP(l) + FN(l)} \tag{3}$$

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:

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

Performance Measures: Continuous Targets

Measures of Error: SSE and MSE

- t_i is the true target value for the $i-$th instance.
- $\mathbb{M}(\mathbf{d}_i)$ is the model's prediction.

$$\text{sum of squared errors} = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2 \tag{4}$$

$$\text{mean squared error} = \frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n} \tag{5}$$

Measures of Error: RMSE and MAE

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

$$\text{root mean squared error (RMSE)} = \sqrt{\frac{\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2}{n}} \tag{6}$$

$$\text{mean absolute error} = \frac{\sum_{i=1}^{n} abs(t_i - \mathbb{M}(\mathbf{d}_i))}{n} \tag{7}$$

Measures of Error: R^2

- The R^2 measure is domain-independent.
- It compares performance against an imaginary model that always predicts the "average value" of all the target features in the entire training set, denoted by $\bar{t}$.
- **Interpretation:** R^2 times 100 is the percentage of variation in the target feature that is explained by the descriptive features in the model.

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}} \tag{8}$$

$$\text{sum of squared errors} = \frac{1}{2}\sum_{i=1}^{n}(t_i - \mathbb{M}(\mathbf{d}_i))^2 \tag{9}$$

$$\text{total sum of squares} = \frac{1}{2}\sum_{i=1}^{n}\left(t_i - \bar{t}\right)^2 \tag{10}$$

Table: Blood thinning drug dosage prediction problem.

		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
	MSE		**1.703**		**4.064**
	RMSE		**1.305**		**2.016**
	MAE		**0.958**		**1.704**
	R^2		**0.921**		**0.812**