




MODEL EVALUATION

CS576 MACHINE LEARNING



Dr. Jin Soung Yoo, Professor
Department of Computer Science
Purdue University Fort Wayne

Reference

- J. D. Kelleher et al., Machine Learning for Predictive Data Analytics, 2nd Ed., Ch 9
- Mitchell, Machine Learning, Ch 5
- Alpaydın, Introduction to Machine Learning, 4th Ed., Ch 20.6-20.11

Outline

- Basic Concepts of Model Evaluation
- Approaches for Model Evaluation
- Performance Metrics
- Techniques for Model Comparison

Basic Concepts of Model Evaluation

- Model evaluation is a fundamental aspect of many learning methodologies.
- The **goal of model evaluation** is to measure and compare the performance of various models, **determining which one *best* performs** in the prediction task that they were designed for.
- **What do we mean by the *best* (good) models?**
 - Defining the term "best" in an objective manner can be hard.
 - In machine learning, a foundational principle suggests that robust models should perform well on new, unseen data.
 - There are various approaches to measure a model's effectiveness.
- **How to evaluate the machine learning models built?**
 - Precise evaluation of their performance is essential.

Basic Concepts of Model Evaluation

- How to obtain a **reliable estimate of performance**?
 - The training error rate may under-estimate the true generalization error rate.
 - The most important part of the design of an evaluation experiment is **ensuring that the data used to evaluate the model is not the same as the data used to train the model**
- We need a **test data set** that is different from the training set.
- When data is limited, what is the best way to use this data to both learn a model and estimate its accuracy ?

Basic Process for Model Evaluation

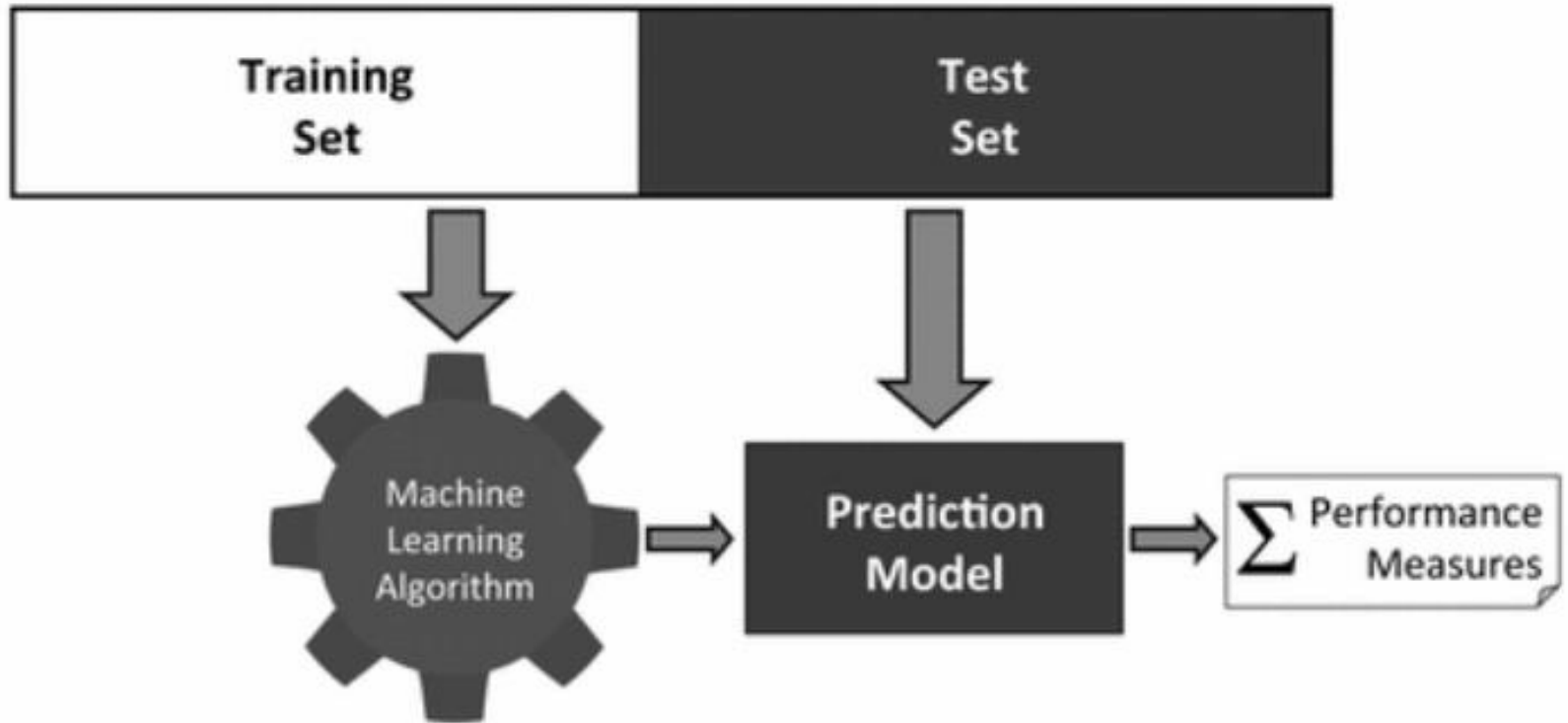


Figure. The basic process of building and evaluating a model

Basic Process of Model Evaluation

- 1. Prepare a test data set** created in the data preparation phase.
 - It is important that the test set should be *never used in the training process*
- 2. Present the test examples to a trained model** to examine how well the model can generalize beyond the examples used to training it.
- 3. Compare the predictions that the model made to the predictions we expected.**
 - A performance measure is used to capture, numerically, how well the prediction made by the model matches those that were expected.
 - Simple performance measures are *error* and *accuracy*.

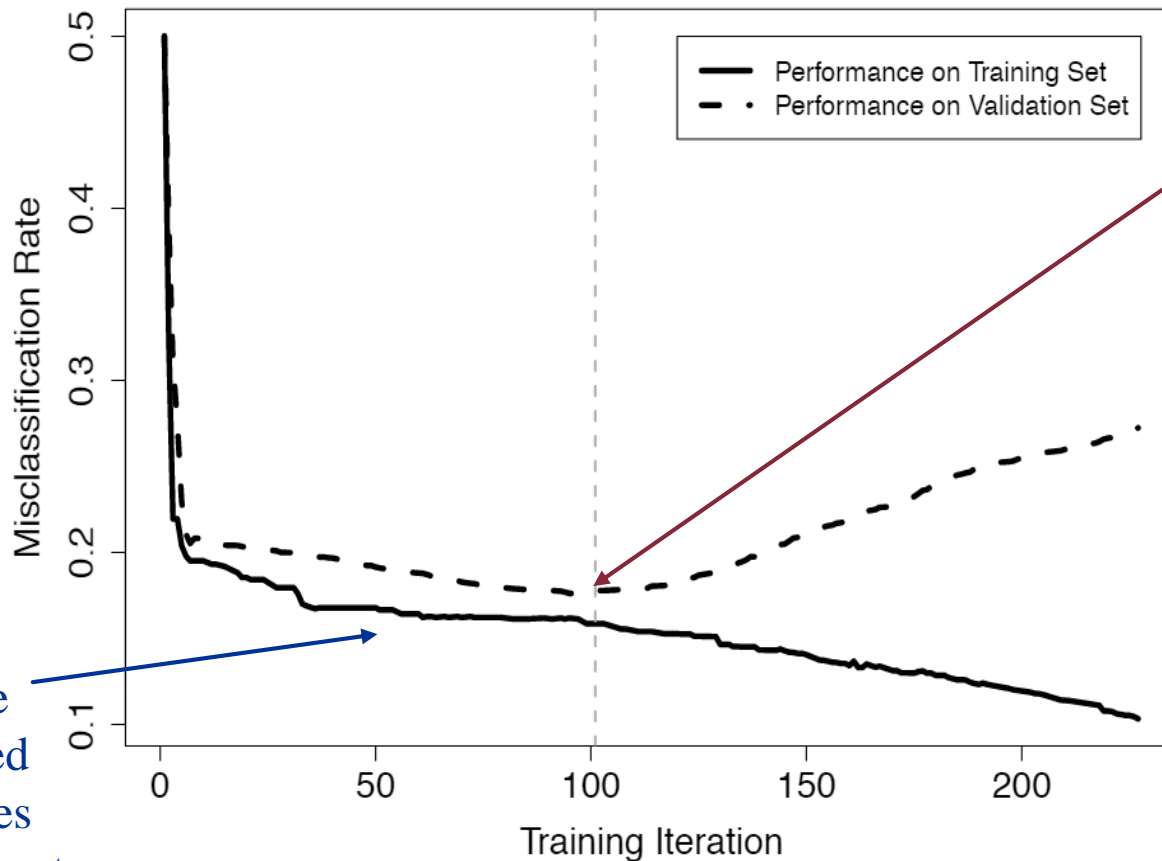
Basic Concepts of Model Evaluation

- Sometimes we also need a *validation set* which are different a training set.
- In machine learning, a **validation set** is a subset of the data that is used **to assess the performance of a model during the training phase, but it's not used for actual training.**
- The **primary purpose of the validation set** is to provide an unbiased evaluation of a model fit on the training dataset while tuning model hyperparameters (like the depth of a decision tree or the learning rate in neural networks).
- While both validation and test sets are essential for model evaluation, **the validation set is for model tuning and iterative improvement**, whereas the test set is for final performance assessment.

Purposes of Validation Set

- **Model Selection and Hyperparameter Tuning:** When you have different models or different sets of hyperparameters for a model, you use the validation set to compare their performances.
- **Preventing Overfitting:** A validation set helps identify Overfitting. If a model performs exceptionally well on training data but poorly on the validation data, it's a clear sign of overfitting.
- **Model Ensemble:** If you are creating an ensemble of models, a validation set can help determine the weights to assign to each model or to choose which models to include in the ensemble.
- **Early Stopping:** In iterative algorithms, especially deep learning, monitoring the performance on the validation set can help in halting the training process when the validation error starts increasing, even if the training error is still decreasing.
- **Choosing Features:** If you're deciding between different sets of features to include in your model, a validation set can help you determine which features help the model generalize best to unseen data.
- Others: **Regularization Tuning, Data Augmentation Techniques, Adaptive Learning, etc.**

Example: Using a Validate Set



The model becomes more and more tuned to the examples in the training set. The error rate gradually decreases.

The **overfitting** has begun to occur at a point where the error of the model on the validation begins to increase.

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

Validation Set vs Test Set

■ Purpose:

- **Validation Set:** Primarily used for model tuning, such as hyperparameter optimization and model selection.
- **Test Set:** Used to evaluate the final model's performance after it has been trained and tuned.

■ Usage Frequency:

- **Validation Set:** Can be used multiple times during the modeling process.
- **Test Set:** Ideally, it should be used only once, after all model tuning has been completed, to prevent "data leakage" or over-optimizing to the test set.

■ Model Selection and Overfitting:

- **Validation Set:** Helps in identifying if the model is overfitting to the training data..
- **Test Set:** Gives a final assessment of overfitting.

Validation Set vs Test Set (cont.)

■ Size:

- **Validation Set:** The size can vary based on the total dataset and the method used for validation.
 - E.g., In k-fold cross-validation, the validation set is $\frac{1}{k}$ of the training dataset.
- **Test Set:** Typically, a certain proportion of the total data (e.g., 20%) is kept aside as the test set, but this can vary based on the size and nature of the dataset.

■ Role in Iterative Process:

- **Validation Set:** Used iteratively.
- **Test Set:** Not used iteratively in the model development process.
- In many machine learning workflows, the data is split into three sets: training, validation, and test.
 - E.g., , an 80-10-10 or a 70-15-15 split might be employed

Outline

- Basic Concepts of Model Evaluation
- Approaches for Model Evaluation
- ☞ **Performance Metrics**
 - Accuracy and Error
 - Metrics for categorical targets
 - Prediction scores
 - Metrics for multinomial targets
 - Metrics for continuous targets
- Techniques for Model Comparison

Accuracy and Error

- Empirically evaluating the **accuracy** of a model is fundamental to Machine Learning.
- **Error** = $\frac{\text{Number of incorrect predictions}}{\text{total number of predictions}}$
- **Accuracy** = $1 - \text{Error}$

Types of Error

- Two notions of *error* (equivalently, *accuracy*)
 - *Sample error*: the error rate of the hypothesis (i.e., model) over the sample of data that is available.
 - *Training error* (*Resubstitution error*) committed on the training dataset
 - *Test error* committed on the test (validation) set
 - *True error* (*Generalization error*): the error rate of the hypothesis over the entire unknown distribution of examples

Possible Outcomes of a Model

- Many metrics of model performance is based on **the counts of test examples correctly and incorrectly predicted** by the model.
- For binary prediction problems there are 4 possible outcomes:
 - **True Positive (TP)**: The number of positive examples that were correctly predicted by the model
 - **True Negative (TN)**: the number of negative examples correctly predicted by the model.
 - **False Positive (FP)**: the number of negative examples incorrectly predicted as positive
 - **False Negative (FN)**: the number of positive examples that were incorrectly predicted as negative.

Confusion Matrix

- **Confusion matrix** : Summary of TP, TN, FP and FN

		PREDICTED CLASS	
		Positive(+)	Negative(-)
ACTUAL CLASS	Positive(+)	<i>TP</i>	<i>FN</i>
	Negative(-)	<i>FP</i>	<i>TN</i>

Target function $c: X \rightarrow \{+, -\}$

* If more than 2 classes, positive tuples are the tuples of the main class of interest and negative tuples are all other tuples.

- **Error** =
$$\frac{(FP+FN)}{(TP+TN+FP+FN)}$$
- **Accuracy** =
$$\frac{(TP+TN)}{(TP+TN+FP+FN)}$$

Example

- The confusion matrix is

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

- Error (*Misclassification Rate*)

$$\frac{FP+FN}{TP+TN+FP+FN} = \frac{2+3}{6+9+2+3} = 0.25$$

- Accuracy (*Classification Accuracy*)

$$\frac{TP+TN}{TP+TN+FP+FN} = \frac{6+9}{6+9+2+3} = 0.75$$

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
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

Table: A sample test set with model predictions

Confusion Matrix-based Measures

ACTUAL CLASS	PREDICTED CLASS		
		positive	negative
	positive	<i>TP</i>	<i>FN</i>
	negative	<i>FP</i>	<i>TN</i>

- **True Positive Rate: $TPR = \frac{TP}{(TP+FN)}$**
 - **true positive recognition rate** (also known as **hit rate**)
- **True Negative Rate: $TNR = \frac{TN}{(TN+FP)}$**
- **False Positive Rate: $FPR = \frac{FP}{(TN+FP)}$**
 - Also known **false alarm rate**
 - $FPR = 1 - TNR$
- **False Negative Rate: $FNR = \frac{FN}{(TP+FN)}$**
 - $FNR = 1 - TPR$

Precision, Recall, Sensitivity, and Specificity

■ **Precision:** $\text{precision} = \frac{TP}{(TP+FP)}$

- (**Exactness**) The fraction of examples that actually are positive in the group of examples that the model has predicted as a positive class.

ACTUAL CLASS	PREDICTED CLASS		
		positive	negative
positive	positive	TP	FN
	negative	FP	TN

■ **Recall:** $\text{recall} = \frac{TP}{(TP+FN)} = TPR$

- (**Completeness**) The fraction of positive examples correctly predicted

■ **Sensitivity:** $\text{Sensitivity} = \frac{TP}{(TP+FN)} = TPR$

- True positive recognition rate

■ **Specificity:** $\text{Specificity} = \frac{TN}{(FP+TN)} = 1 - FPR = TNR$

- True negative recognition rate

F-Measure

- Building a model that maximizes both precision (for exactness) and recall (for completeness) is the key challenge of classification algorithms.
 - A model that declares every record to be the positive class will have a perfect recall, but very poor precision.
 - Conversely, a model that assigns a positive class to every test record that matches one of the positive records in the training set has very high precision, but low recall.
- **F-measure** (F_1 or **F-score**) : **harmonic mean** of precision and recall

$$\mathbf{F - mearue} = 2 * \frac{(\textit{precision} \times \textit{recall})}{(\textit{precision} + \textit{recall})}$$

- A high value of F-measure ensures that both precision and recall are reasonably high.

Problem of Accuracy with Imbalanced Class

- When one class may be rare, e.g., majority of the negative class and minority of the positive class (e.g., cancer='yes')
 - E.g., Consider a 2-class problem
 - Number of negative examples = 9990
 - Number of positive examples = 10
 - If a model predicts everything to be negative, **accuracy** is $9990/10000 = 99.9\%$
 - This is misleading because the model does not detect any positive examples (rare class example).
 - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc.)
- For the class imbalance problem, we need metrics to measure how well the learner (model) can recognize the positive examples (e.g., *sensitivity*) and also how well it can recognize the negative examples (e.g., *specificity*)

Average Class Accuracy

- *Average class accuracy* can be used for data with imbalanced class

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

where $levels(t)$ is the set of levels (class labels) that the target feature t and $recall_l$ is the recall achieved by a model for level l

- Or using **harmonic mean** (preferred)

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

- Example

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

$$Recall_{non-churn} = \frac{70}{90} = 0.778$$

$$Recall_{churn} = \frac{8}{10} = 0.800$$

$$\begin{aligned} \text{average class accuracy}_{HM} &= \frac{1}{\frac{1}{2} \left(\frac{1}{0.778} + \frac{1}{0.800} \right)} \\ &= 78.873\% \end{aligned}$$

Measuring Profit and Loss

- It is not always correct to treat all outcomes equally
 - E.g., In the churn prediction example, correctly classifying a customer as likely to churn is more worth than correctly classifying a customer as not likely to churn.
- In these cases, it is useful to **take into account the cost of the different outcomes** when evaluating models

- **Profit matrix**

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

- TP_{profit} represents the profit arising from a correct positive prediction
- FN_{profit} is the profit arising from an incorrect negative prediction
- Profit can refer to a positive or a negative value.
- The actual values in a profit matrix are determined through domain expertise. It might be hard to exactly quantify the profit associated TP, FN, FP and TN. The relative profit associated with each outcome is fine.

Example

- A loan company has built a **credit scoring model** to predict the likelihood that a borrower will default on a loan.

		Prediction	
		<i>good</i>	<i>bad</i>
Target	<i>good</i>	140	-140
	<i>bad</i>	-700	0

Table The profit matrix for the payday loan credit scoring problem, where *good* means good borrowers, and *bad* means bad borrowers

- Incorrectly predicting the *good* level for a borrower who turn out to be *bad* (not repaid the loan in full) is a very costly mistake (\$700 loss).
- Comparison of overall profits of two models

(a) Model 1's confusion matrix

		Prediction	
		<i>good</i>	<i>bad</i>
Target	<i>good</i>	57	3
	<i>bad</i>	10	30

Model 1's overall profit

		Prediction	
		<i>good</i>	<i>bad</i>
Target	<i>good</i>	7,980	-420
	<i>bad</i>	-7,000	0
Profit		560	

(b) Model 2's confusion matrix

		Prediction	
		<i>good</i>	<i>bad</i>
Target	<i>good</i>	43	17
	<i>bad</i>	3	37

Model 2's overall profit

		Prediction	
		<i>good</i>	<i>bad</i>
Target	<i>good</i>	6,020	-2,380
	<i>bad</i>	-2,100	0
Profit		1,540	

- Model 2 results in a higher profit than Model 1

Outline

- Basic Concepts of Model Evaluation
- Approaches for Model Evaluation
- **Performance Metrics**
 - Accuracy and Error
 - Metrics for categorical targets
 - ☞ **Prediction scores**
 - Metrics for multinomial targets
 - Performance measures for continuous targets
- Techniques for Model Comparison

Prediction Score

- All classifiers do not simply produce a target feature level (class label) as its output.
- In many cases, a **prediction score** (usually between 0 and 1 for each class) is produced. This score is a measure of the model's confidence that a given instance belongs to a particular class.
- A **threshold process** is used to convert this score into one of the levels of the target feature.
- In a binary classification, a **threshold of 0.5** is used to convert this score into a categorical prediction as :

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

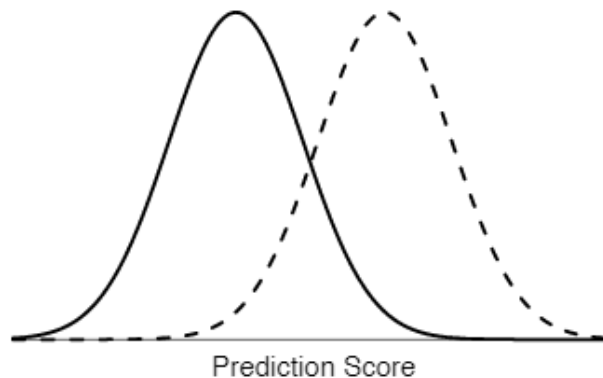
Example

- **Convert scores to class levels**
 - The test examples are sorted by score.
 - Apply the threshold (0.5) process.
- In general, examples that actually should get a prediction of negative ('ham') generally have a low score, and those that should get a prediction of positive ('spam') generally get a high score.

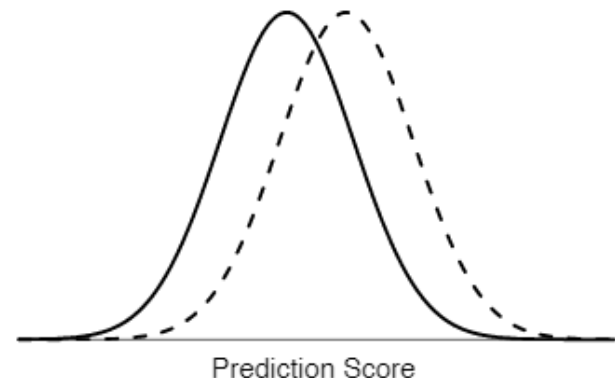
ID	Target	Prediction	Score	Outcome
7	ham	ham	0.001	TN
11	ham	ham	0.003	TN
15	ham	ham	0.059	TN
13	ham	ham	0.064	TN
19	ham	ham	0.094	TN
12	spam	ham	0.160	FN
2	spam	ham	0.184	FN
3	ham	ham	0.226	TN
16	ham	ham	0.246	TN
1	spam	ham	0.293	FN
5	ham	ham	0.302	TN
14	ham	ham	0.348	TN
17	ham	spam	0.657	FP
8	spam	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

Model Performance by Prediction Scores

- To assess how well the model is performing, you can measure how well the **distributions of scores** produced by the model for different target levels **are separated**
- If the distributions follow normal distributions, the separation can be examined with the means and standard deviations.



(a)



(b)

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

TPR, TNR, and Prediction Score Threshold

- TPR and TNR are intrinsically tied to the threshold used to convert prediction scores into target levels.
- The prediction score threshold can be changed, however, which leads to different predictions and a different confusion matrix.
- Example

(a) Threshold: 0.75

		Prediction	
		<i>spam</i>	<i>ham</i>
Target	<i>spam</i>	4	4
	<i>ham</i>	2	10

TPR=0.5, TNR=0.833

(b) Threshold: 0.25

		Prediction	
		<i>spam</i>	<i>ham</i>
Target	<i>spam</i>	7	2
	<i>ham</i>	4	7

TPR=0.777, TNR=0.636

Trade-offs in TPR and TNR

- For every possible value of the threshold, in the range $[0,1]$, there are corresponding TPR and TNR values.
- As the threshold increases TPR decreases and TNR increases (and vice versa).
- But misclassification rate doesn't change that much as the threshold changes due to the trade-offs between false positives and false negatives

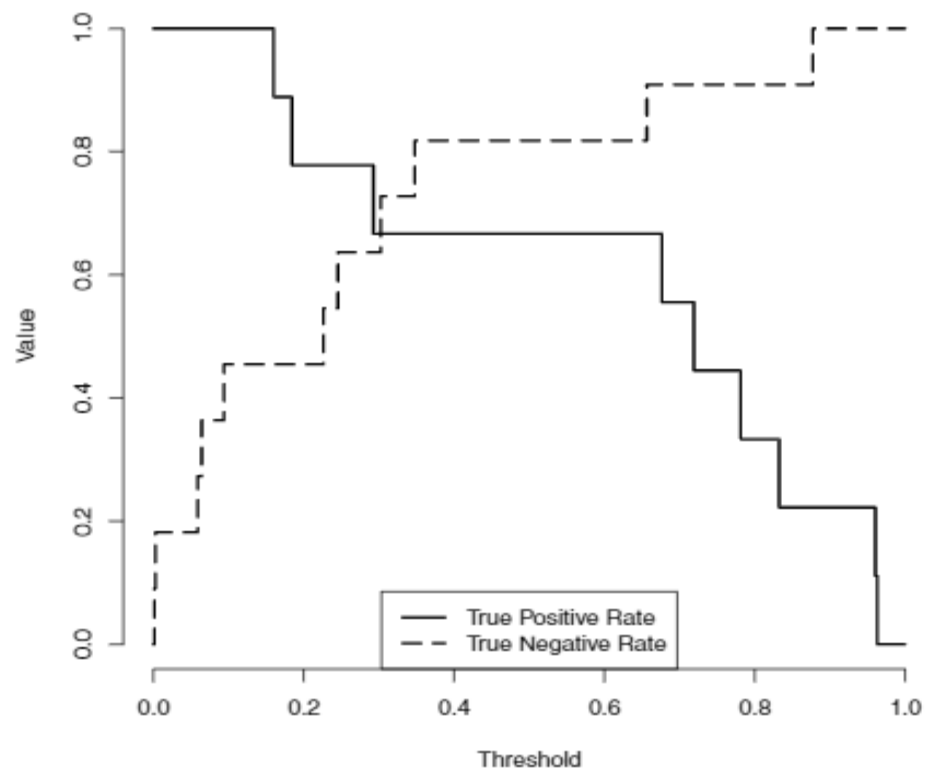


Figure The changing values of TPR and TNR for the test spam/ham data as the threshold is altered

Predictions based on Different Threshold Values

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

ROC Curve

- Capturing a trade-off between accuracy for predictions of positive target levels and accuracy for predictions of negative target levels is the basis of the ROC curve.
- **Receiver Operating Characteristic curve (ROC curve)**
 - TPR in y axis, FPR in x axis
 - A given threshold, TPR and FPR values are presented by a point in **ROC space**.
 - **ROC curve** is drawn by plotting a point for every feasible threshold value.

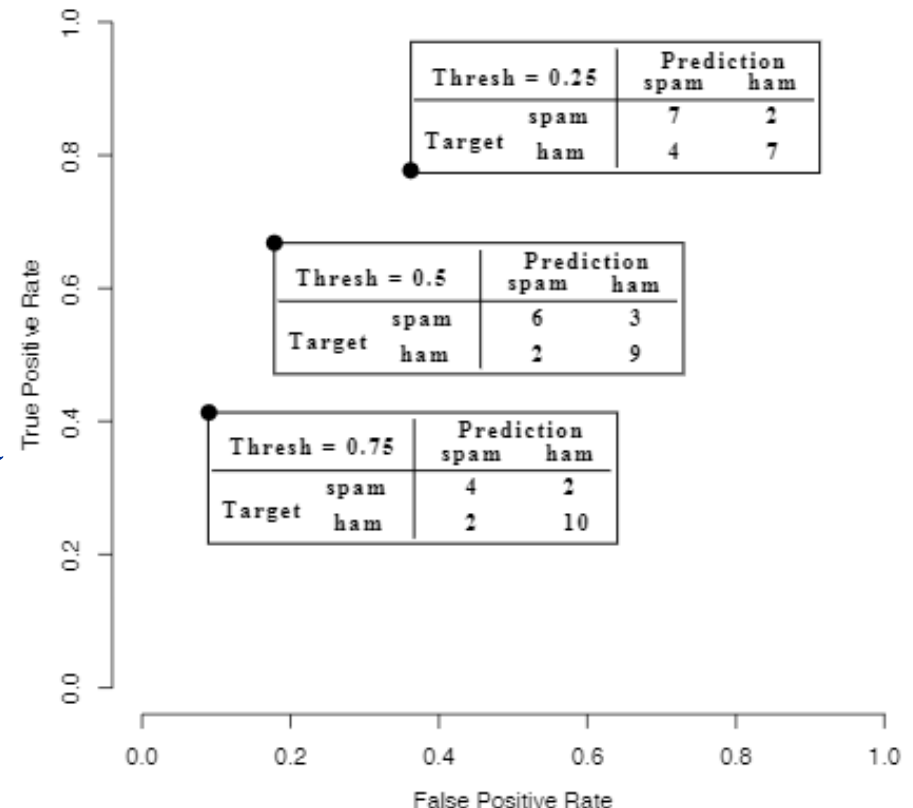
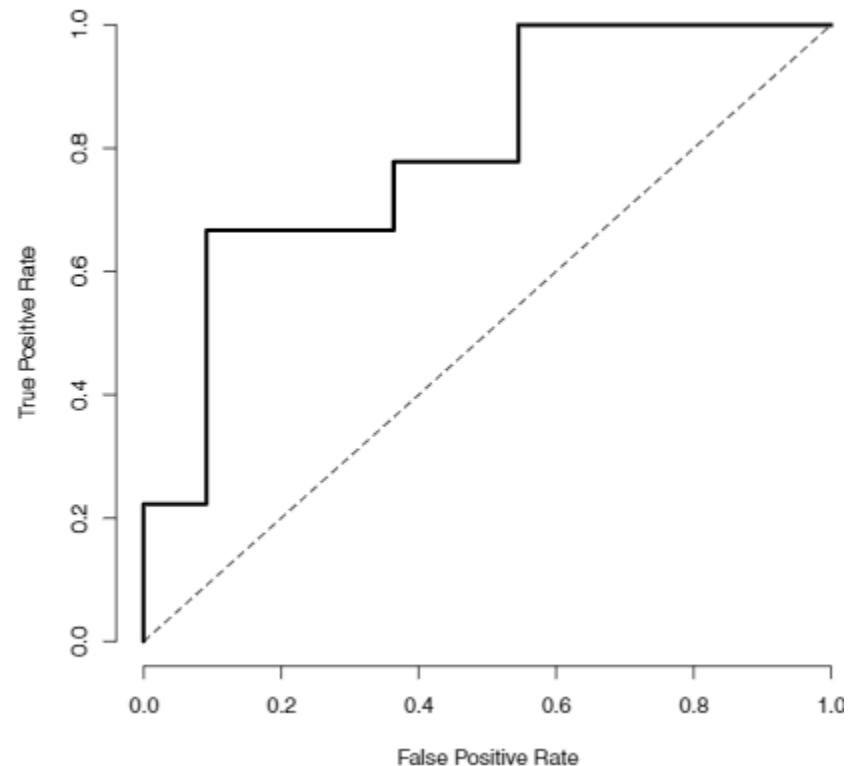


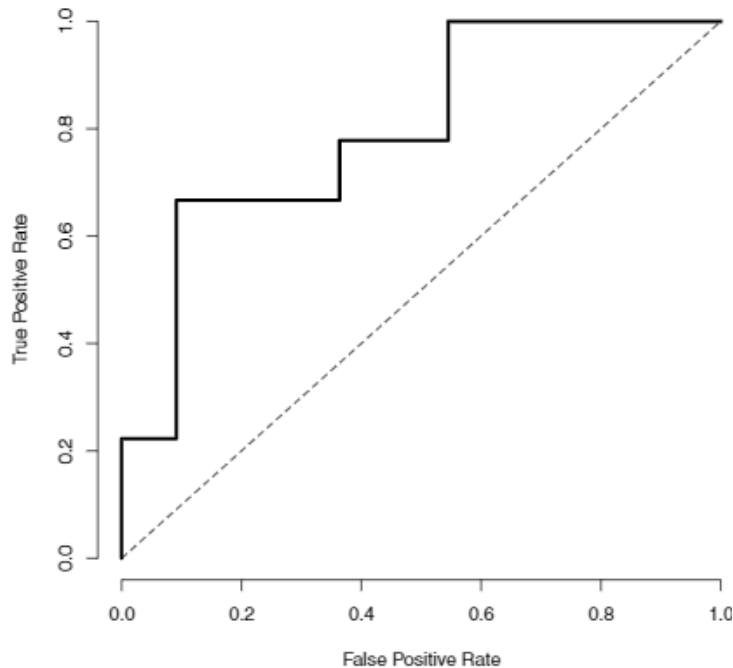
Figure Points in ROC space for thresholds of 0.25, 0.5, and 0.75.

ROC Curve

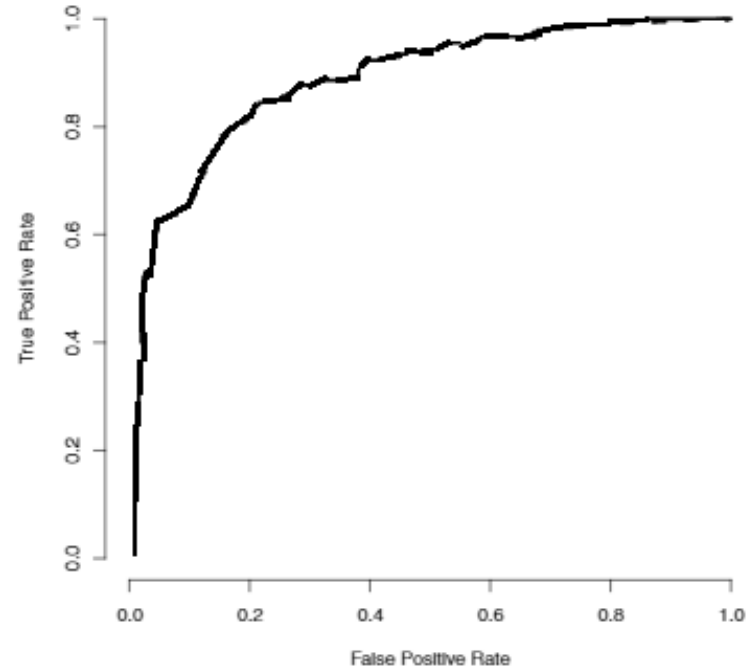
- The **diagonal** of ROC space from (0,0) to (1,1) is a reference line representing the expected performance of a model that makes random predictions.
- The ROC curve for a trained model **is expected to be above** the random reference line.
- The strength of a model increases, the ROC curve **moves farther away from the random line toward the top left-hand corner** of ROC space – toward a TPR of 1.0 and FPR of 0.0



Example: ROC Curves



(a)

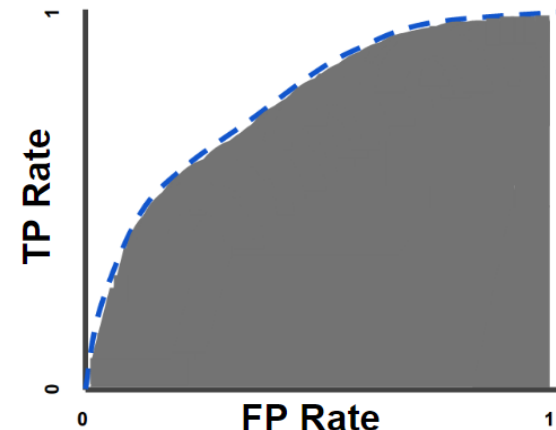


(b)

Figure Two ROC curves. The ROC curve in (b) is generated with many more instances than (a), which is why the curve (b) is so much smoother than the curve in (a)

ROC Index (AUC)

- The ROC curve gives a single numeric performance measure with which models can be assessed.
- The **ROC index** or **Area Under the Curve (AUC)**, which is based ROC curve
 - measures **the area underneath an ROC curve**,
 - is the value in the range [0,1]



ROC index =

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

where \mathbf{T} is a set of thresholds, $|\mathbf{T}|$ is the number of thresholds tested. $TPR(\mathbf{T}[i])$ and $FPR(\mathbf{T}[i])$ are the true positive and false positive rates at threshold i respectively.

AUC (ROC Index)

- AUC is a widely used performance measure that is calculated using prediction scores.
- AUC is **quite robust in the presence of imbalanced data**
- **AUC can be interpreted probabilistically as the probability that a model will assign a higher rank to a randomly selected positive instance than to a randomly selected negative instance.**
- Although an application-specific decision, a good rule of thumb is that **an AUC value of 0.7 indicates a *strong* model**, while a value below 0.6 indicates a *weak* model.
- An AUC of 0.5 means the model has no discrimination capabilities

Gini Coefficient

- **Gini coefficient** is another commonly used performance measure that is just a linear rescaling of the AUC (ROC index):

$$\text{Gini coefficient} = (2 \times \text{AUC}) - 1$$

- The Gini coefficient can take values in the range [0,1]
- The Gini coefficient is a measure of the inequality of a distribution. A value of 0 expressing total equality and a value of 1 maximal inequality.
- In the context of classifier performance, **it quantifies the discriminative power of the model.**

Kolmogorov-Smirnov Statistic

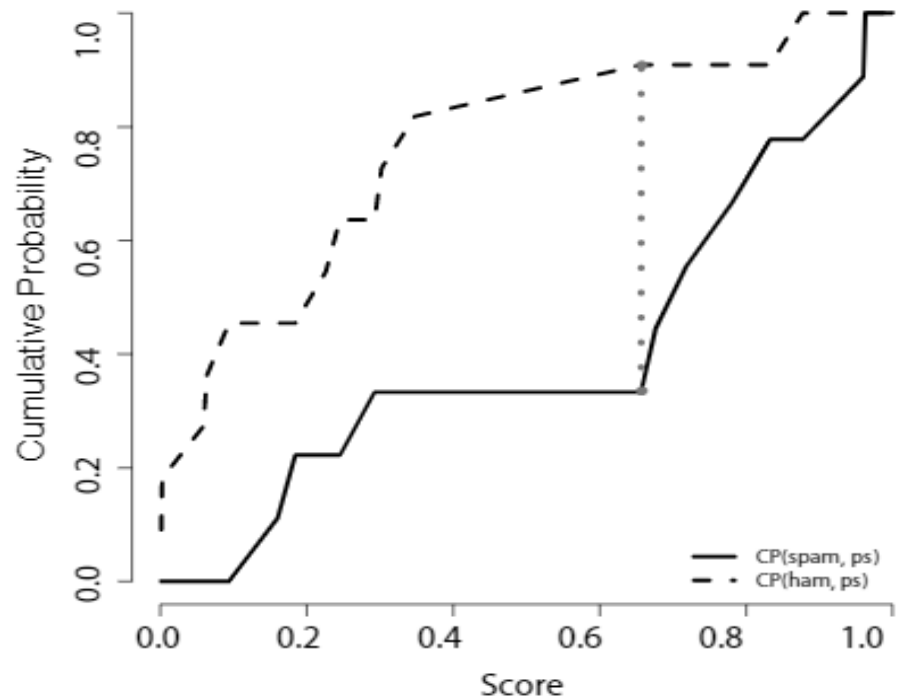
- The **Kolmogorov-Smirnov statistic (K-S statistic)** is another performance measure that **captures the separation between the distribution of prediction scores for the different target levels** in a classification problem.
- To calculate the K-S statistic, we first establish the *cumulative probability distributions* for the prediction scores corresponding to both positive and negative target values:

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

where ps is a prediction score value, $CP(positive, ps)$ is the cumulative probability distribution of positive value scores, and $CP(negative, ps)$ is the cumulative probability distribution of negative value scores.

Kolmogorov-Smirnov Statistic (cont.)

- These cumulative probability distributions can be plotted on a **Kolmogorov-Smirnov chart (K-S chart)**.
- The K-S chart is a tool that visualizes the difference between the cumulative distributions of two groups.
- The chart is often used to assess the **discriminatory power of a prediction model**.

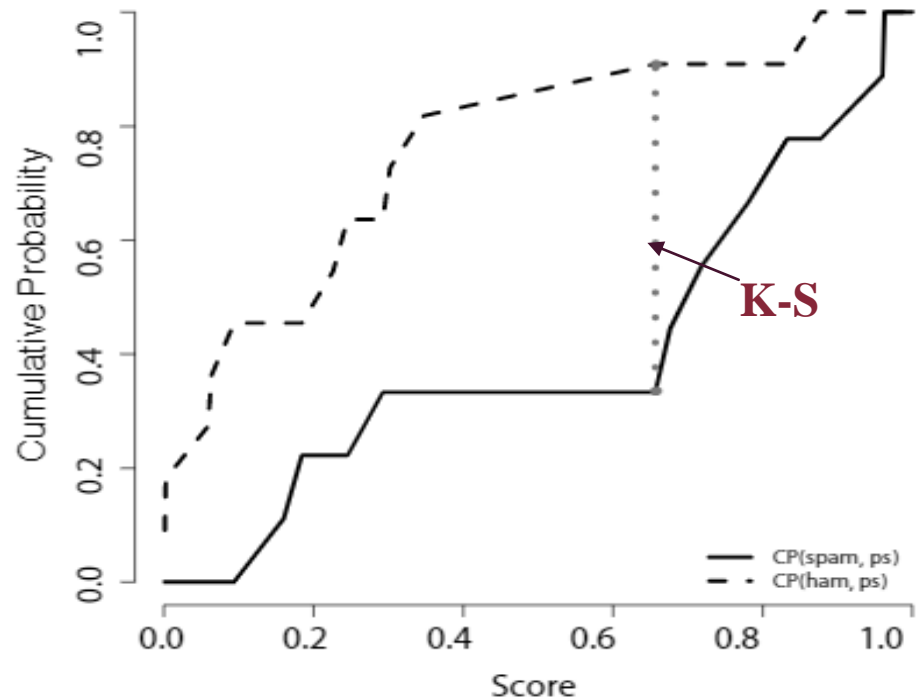


Kolmogorov-Smirnov Statistic (cont.)

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

$$K - S = \max_{ps} (CP(positive, ps) - CP(negative, ps))$$

- The K-S statistic **ranges from 0 to 1**.
- **Higher values indicate better model performance** which reflects that there is a clear distinction of the scores predicted by the model for the negative and the positive instances.



Example: K-S Chart and K-S Statistic

- The "ham" line is rising steeply and the "spam" line is more gradual. → **The model is more effectively identifying "ham" emails than "spam" emails for the given prediction scores.**

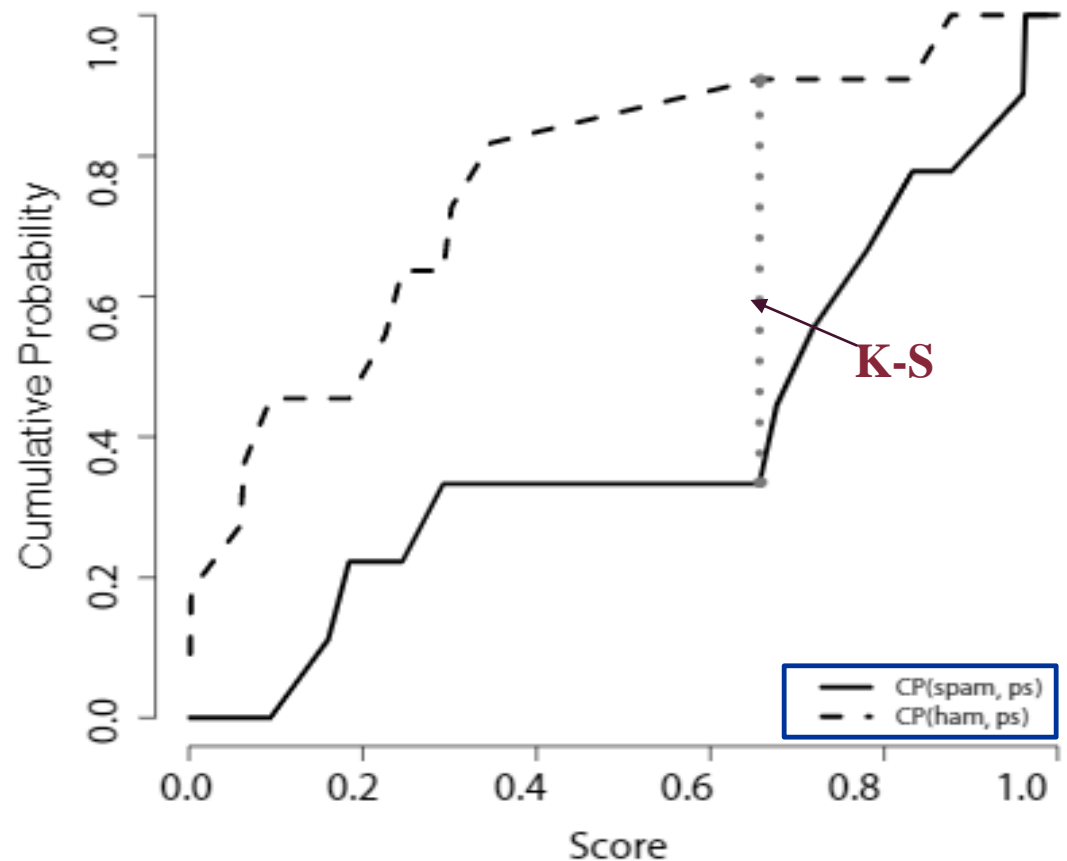


Figure The K-S chart for the **email classification** predictions.

Example (cont.)

- In this example, the K-S statistic is **0.576**

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

* marks the maximum distance, which is the K-S statistic.

Measuring Gain and Lift

- In situations where there's a **specific positive target level of particular interest** (e.g., detecting spam emails, identifying fraudulent transactions, or pinpointing customers likely to respond to an offer), it might be more beneficial to evaluate the model's proficiency in predicting those specific instances rather than assessing its ability to differentiate between both target levels.
- Two useful performance measures are **gain** and **lift** (**cumulative gain** and **cumulative lift**)
- The **fundamental premise** of both **gain** and **lift** is that when we rank test set examples based on their prediction scores in descending order (for positive target instances) from a high-performing model, we anticipate most of the positive examples to be positioned at the upper part of this ranking.

Gain and Cumulative Gain

- First rank the predictions made for a test set in descending order by prediction score and then divide them into deciles.
- A *decile* is a group containing 10% of a dataset.

- The **gain** in a given decile is

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

- **Gain** measures how many of the positive instances in the overall test set are found in a particular decile.
- If the percentage of positive instances within each decile **matches** the overall percentage of positive instances in the entire dataset, then **the model is performing no better than random guessing**
- The **cumulative gain** up to a particular decile is

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

Lift and Cumulative Lift

- The **lift** at an decile is

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

- **Lift** is the **ratio between percentage** of positive instances in that decile and the percentage of positive instances overall in the population
- It tells us **how much higher** the actual percentage of positive instances in a decile is than the rate expected.
- Lift can take in the range $[0, \infty]$.
- The **cumulative lift** up to a particular decile is

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

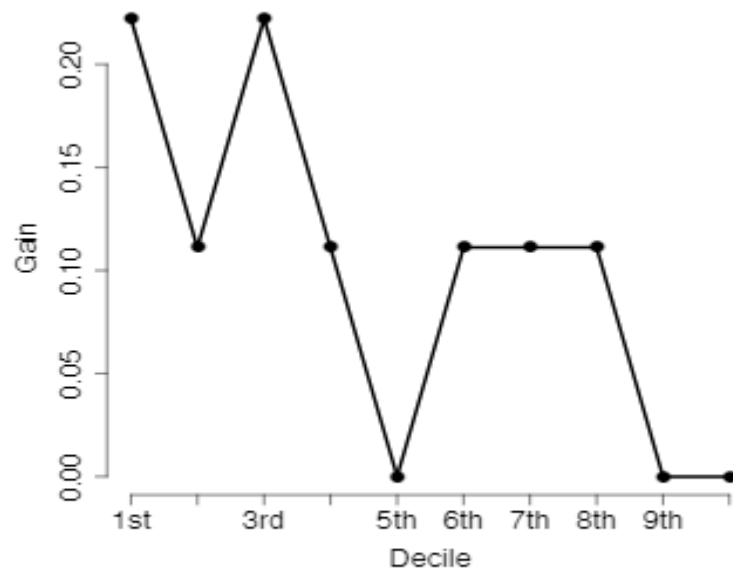
Example: Gain, Cum. Gain, Lift & Cum. Lift

- The test set with model predictions and scores for the email classification problem
 - The predictions made for a test set is ranked **in descending order by prediction score** →
- Calculations of **gain, cumulative gain, lift and cumulative lift** ↓

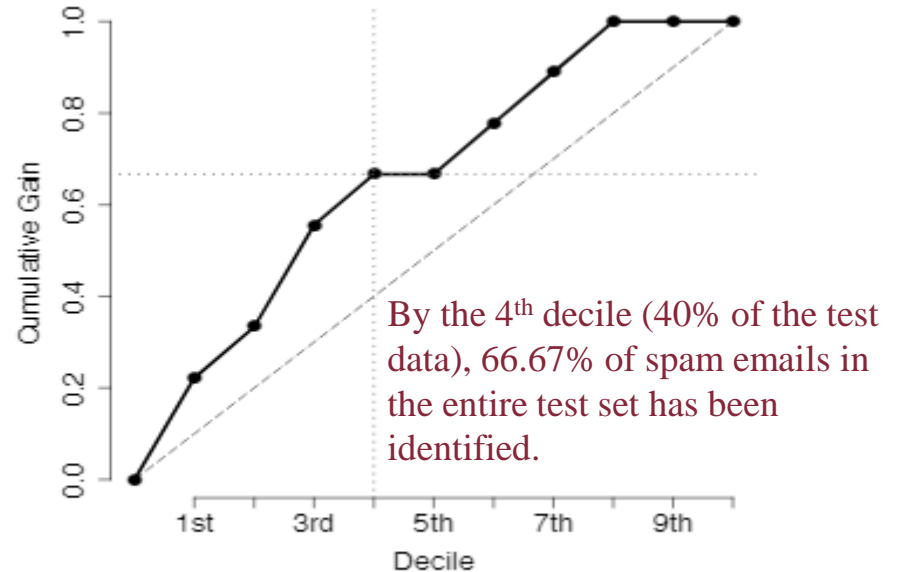
Decile	Positive (spam) Count	Negative (ham) Count	Gain	Cum. Gain	Lift	Cum. Lift
1 st	2	0	0.222	0.222	2.222	2.222
2 nd	1	1	0.111	0.333	1.111	1.667
3 rd	2	0	0.222	0.556	2.222	1.852
4 th	1	1	0.111	0.667	1.111	1.667
5 th	0	2	0.000	0.667	0.000	1.333
6 th	1	1	0.111	0.778	1.111	1.296
7 th	1	1	0.111	0.889	1.111	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

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

Example: Gain Chart, Cumulative Gain Chart



(a)

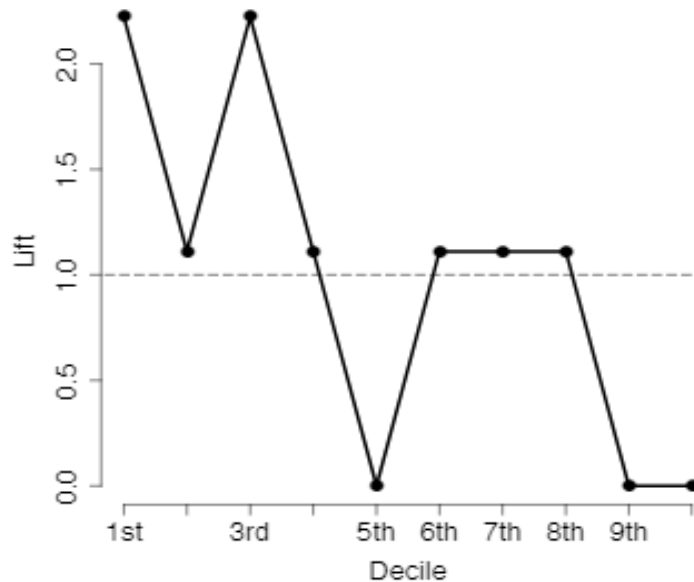


(b)

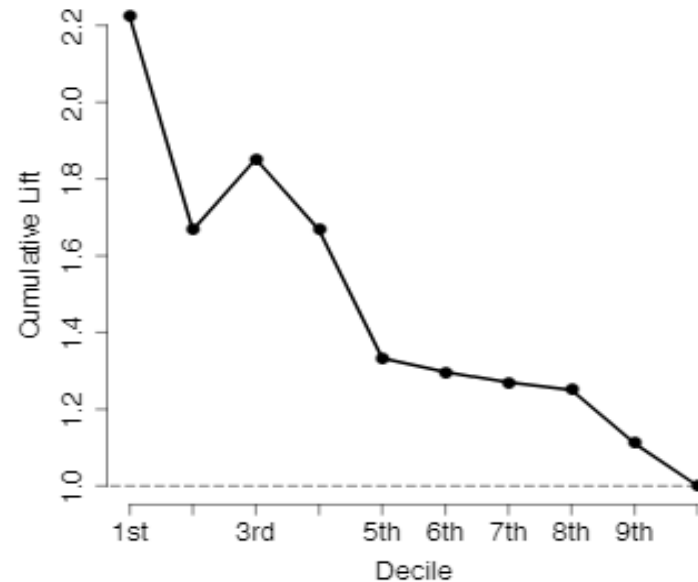
Figure. The (a) **gain** and (b) **cumulative gain** at each decile for the email predictions

- In (a), **the gain is higher for the lower deciles, which contains instances with the highest scores.** (Note that the predictions made for a test set is ranked in descending order by prediction score). **This is indicative for the fact the model is performing reasonably well for positive cases.**
- (b) shows how many of the positive instances in a complete set we can expect to have identified at each decile of the data set

Example: Lift Chart, Cumulative Lift Chart



(a)



(b)

Figure The (a) **lift** and (b) **cumulative lift** at each decile for the email predictions

- **For a well-performed model, the lift curve start well above 1.0 and cross 1.0 at one of the lower deciles**

Application

- **Cumulative gain** is especially useful in **customer relationship management (CRM) applications** such as cross-sell and upsell models.
- The cumulative gain tells us **how many customers we need to contact in order to reach a particular percentage of those who are likely to respond to an offer**, which is an incredibly useful piece of information to know when planning customer contact budgets.

Outline

- Basic Concepts of Model Evaluation
- Approaches for Model Evaluation
- **Performance Metrics**
 - Accuracy and Error
 - Metrics for categorical targets
 - Prediction scores
 - 👉 **Metrics for multinomial targets**
 - Metrics for continuous targets
- Techniques for Model Comparison

Performance Measures: Multinomial Targets

- When we deal with **multinomial** (**multiple target levels**) prediction problems, we need a different set of performance measures.
- Confusion matrix with l target levels**

		Prediction					
		<i>level 1</i>	<i>level 2</i>	<i>level 3</i>	\dots	<i>level l</i>	Recall
Target	<i>level 1</i>	-	-	-		-	-
	<i>level 2</i>	-	-	-		-	-
	<i>level 3</i>	-	-	-		-	-
	\vdots				\ddots		\vdots
	<i>level l</i>	-	-	-		-	-
Precision		-	-	-	\dots	-	

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

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

- $TP(l)$ refers to the number of instances correctly given a prediction of the target level l
- $FP(l)$ refers to the number of instances incorrectly given a prediction of the target level l
- $FN(l)$ refers to the number of instances that should have been given a prediction of target level l but were given some other prediction.

- Average class accuracy**

$$average\ class\ accuracy_{HM} = \frac{1}{|levels(t)|} \sum_{l \in levels(t)} \frac{1}{recall_l}$$

Example

		Prediction				Recall
		<i>durionis</i>	<i>ficulneus</i>	<i>fructosus</i>	<i>pseudo.</i>	
Target	<i>durionis</i>	5	0	2	0	0.714
	<i>ficulneus</i>	0	6	1	0	0.857
	<i>fructosus</i>	0	1	10	0	0.909
	<i>pseudo.</i>	0	0	2	3	0.600
Precision		1.000	0.857	0.667	1.000	

- Overall accuracy is 80%
- The individual recall scores for each target level show that the performance of the model is not the same for all four levels
 - the accuracy on the *ficulneus* and *fructosus* levels is quite high (85.714% and 90.909% respectively),
 - while for the *durionis* and *pseudoficulneus* levels, the accuracy is considerably lower (71.429% and 60.000%).

- The average class accuracy_{HM} performance measure is
- $$\text{average class accuracy}_{HM} = \frac{1}{\frac{1}{4}(\frac{1}{0.714} + \frac{1}{0.857} + \frac{1}{0.909} + \frac{1}{0.600})} = 75\%$$

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

Performance Measures: Continuous Targets

■ Basic measures of errors with continuous targets

When t_1, \dots, t_n is a set of n expected target values, and $\mathbb{M}(d_1), \dots, \mathbb{M}(d_n)$ is a set of n predictions for a set of test instances, d_1, \dots, d_n

■ Sum of Squared Errors (L_2) :

$$\text{sum of squared errors} = \frac{1}{2} \sum_{i=1}^n (t_i - \mathbb{M}(d_i))^2$$

■ Mean Squared Errors (MSE) :

$$\text{mean squared errors} = \frac{\sum_{i=1}^n (t_i - \mathbb{M}(d_i))^2}{n}$$

- MSE values in the range $[0, \infty]$. Smaller values indicate better model performance

■ Root Mean Squared Errors (RMSE) :

$$\text{root mean squared errors} = \sqrt{\frac{\sum_{i=1}^n (t_i - \mathbb{M}(d_i))^2}{n}}$$

- RMSE values are in the same units as the target value.
- The root mean squared error tends to overestimate error slightly as it overemphasizes individual large errors.

Performance Measures: Continuous Targets (cont.)

- **Mean Absolute Errors (MAE) :**

$$\text{mean absolute errors} = \frac{\sum_{i=1}^n \text{abs}(t_i - \mathbb{M}(d_i))}{n}$$

- MAE values are in the same units as the target value.
- MAE values in the range $[0, \infty]$. Smaller values indicate better model performance

- Overall, the use of **Root Mean Squared Errors (RMSE)** is recommended.

Domain Independent Measure of Error

- The **R^2 coefficient** is a **normalized, domain independent measure** of model performance that is frequently use for prediction problems with a continuous target.

$$\begin{aligned} R^2 &= 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}} \\ &= \frac{\frac{1}{2} \sum_{i=1}^n (t_i - \mathbb{M}(d_i))^2}{\frac{1}{2} \sum_{i=1}^n (t_i - \bar{t})^2} \end{aligned}$$

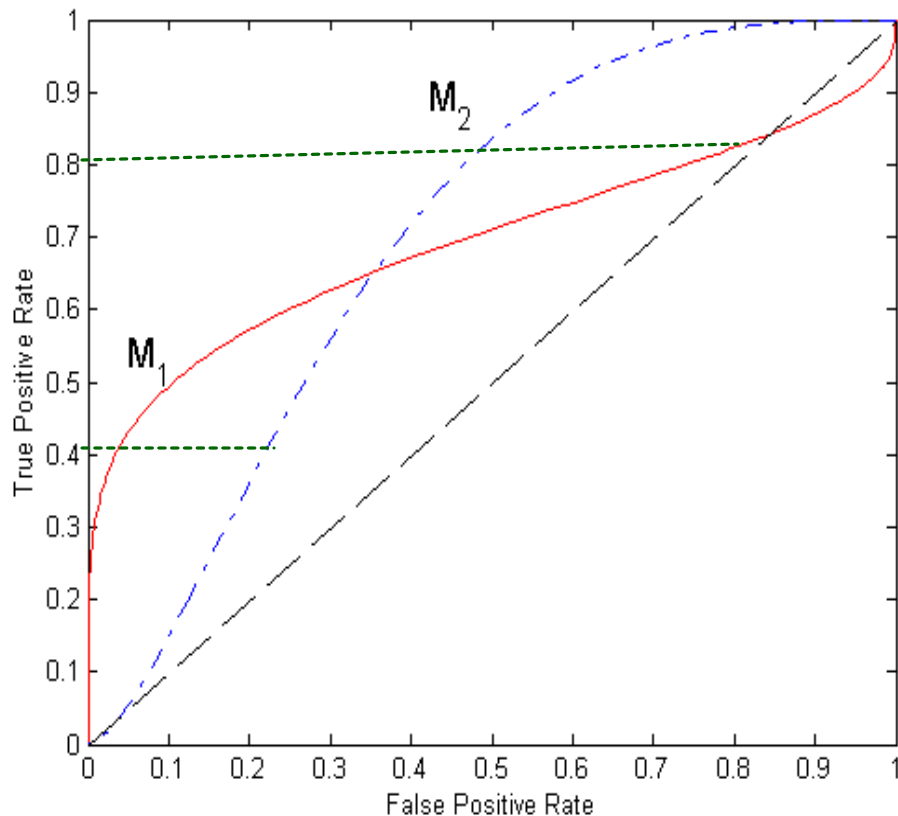
- R^2 values in the range $[0, 1)$.
- **Larger values indicate better model performance**

Outline

- Basic Concepts of Model Evaluation
- Approaches for Model Evaluation
- Performance Metrics
- ☞ **Techniques for Model Comparison**

Model Comparison Using ROC Curve

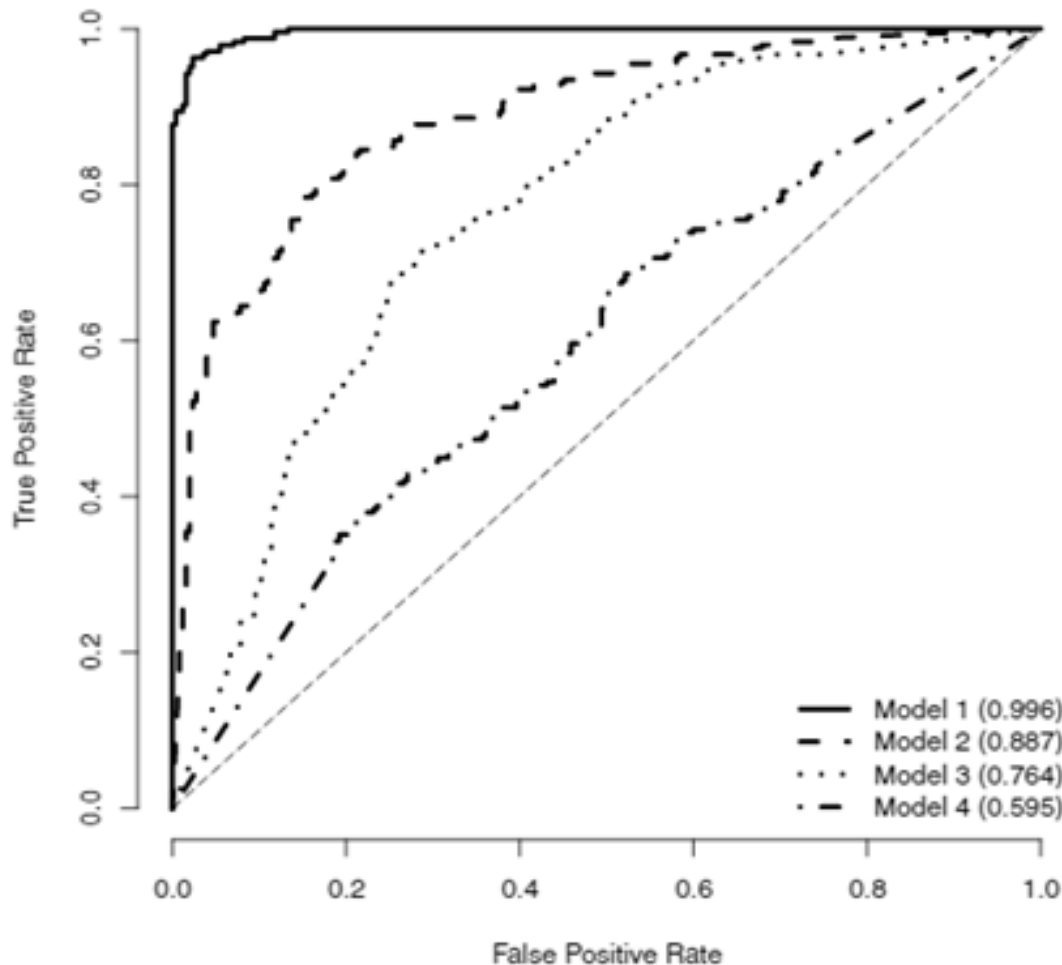
- It is useful to visually compare the performance of different models using **ROC curves**



- No model consistently outperforms the other.

- Model M_1 excels if a small, focused sample is sought.
 - That is if you aim to cover just 40 % of the true positives
 - You should choose Model M_1 which gives a false positive rate of around 5%, rather than Model M_2 , which gives more than 20% false positive
- But Model M_2 excels if you are planning a large sample:
 - If you are covering 80% of the true positives,
 - M_1 will give a false positive rate of 60% as compared with Model M_2 's 80%.

Model Comparison Using ROC Curve

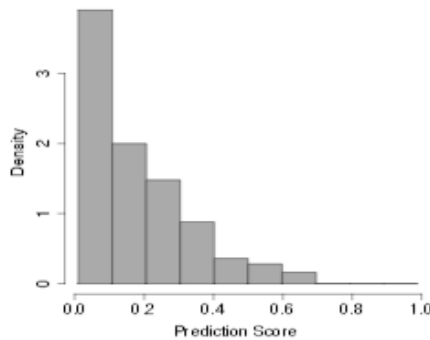


- Model 1 approaches perfect performance
- Model 4 is barely better than random guessing, and
- Models 2 and 3 sit somewhere in between these two extremes.

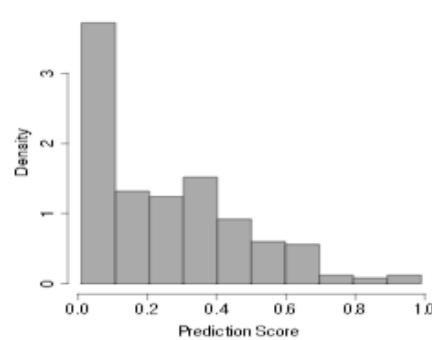
Figure. The ROC curves of different models trained on the email classification task.

Density Histogram, K-S Chart on Email Classification

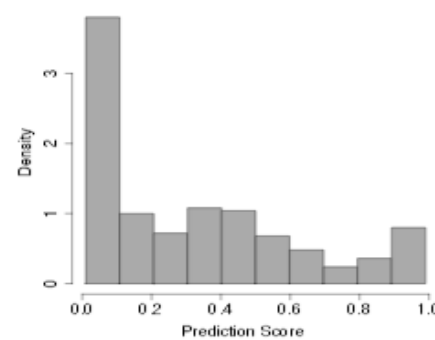
(a) Model 1



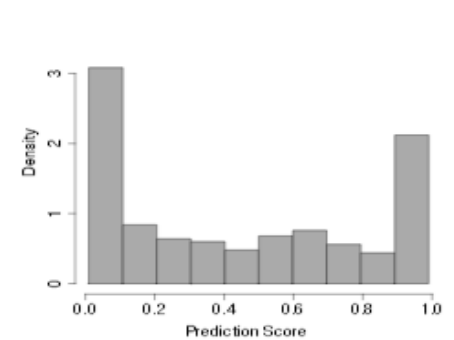
(b) Model 2



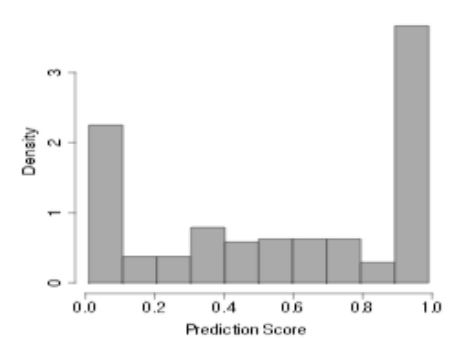
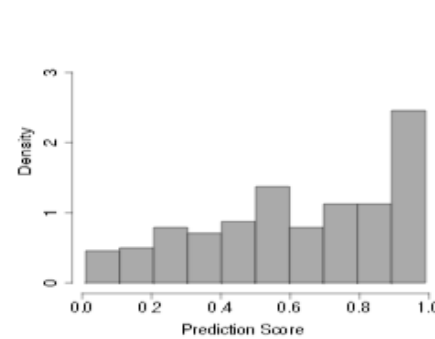
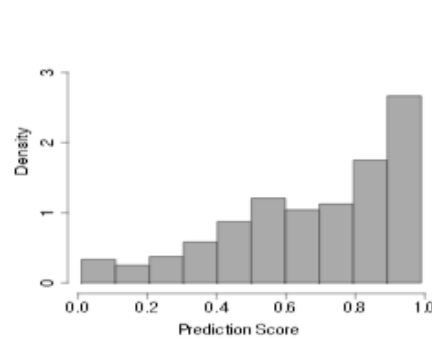
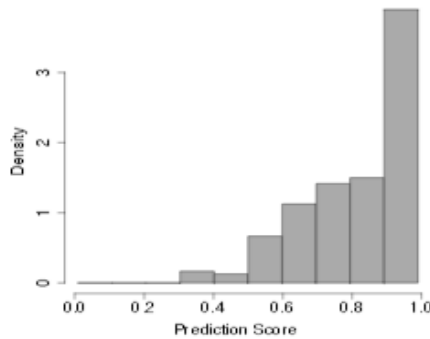
(c) Model 3



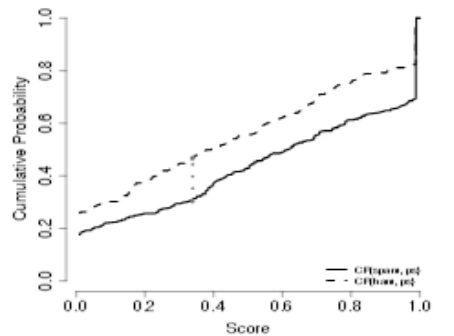
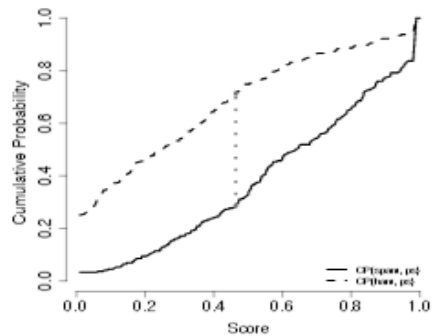
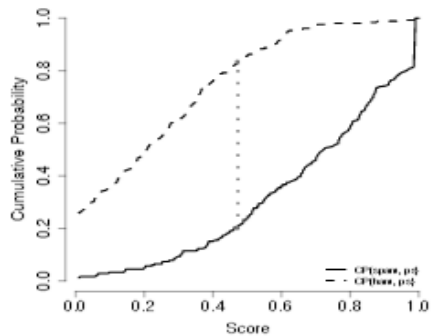
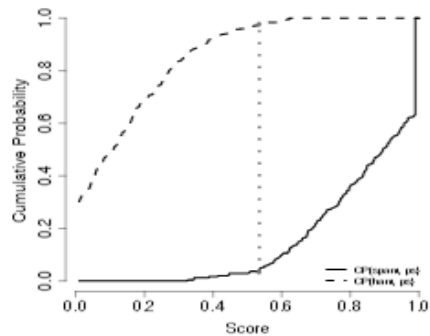
(d) Model 4



(I) Histograms of the spam scores predicted by the models

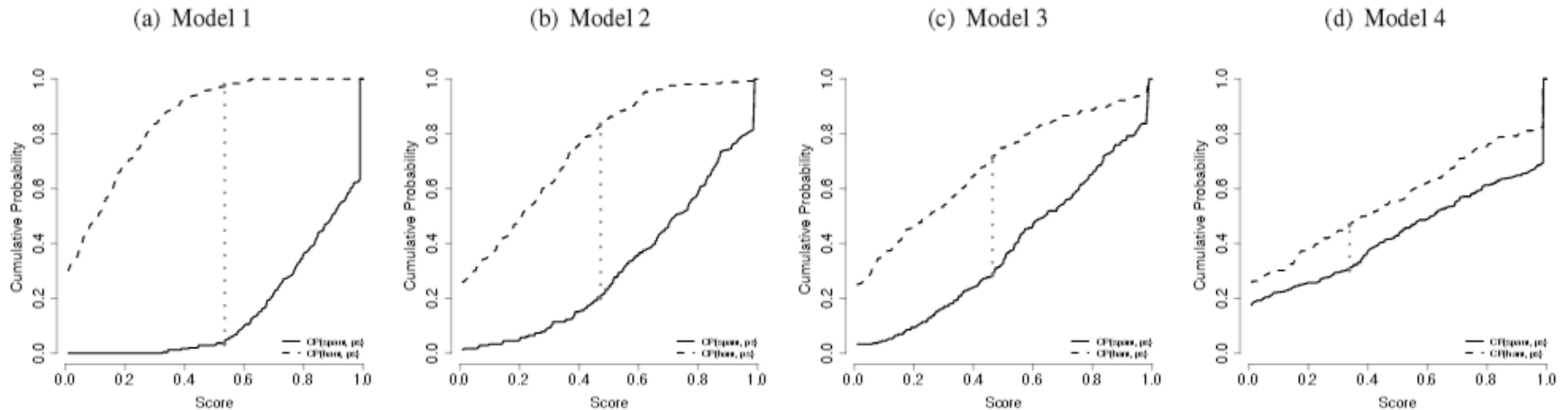


(II) Histograms of the ham scores predicted by the models



(III) Resulting K-S Charts

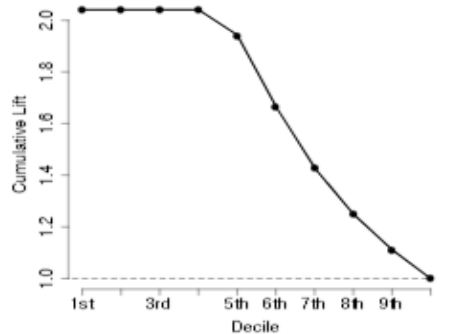
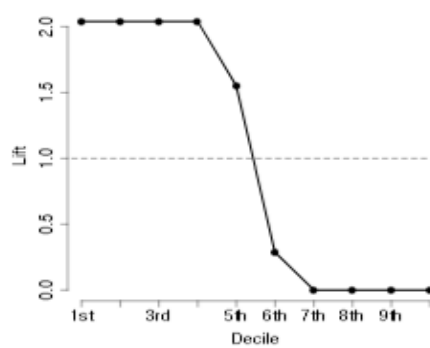
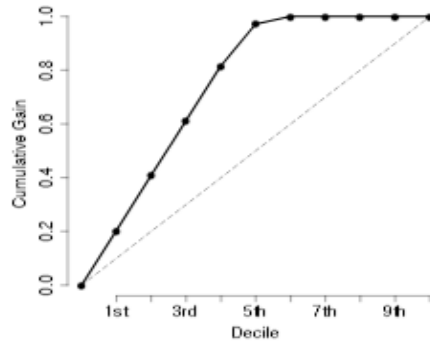
Model Comparison using K-S Statistic



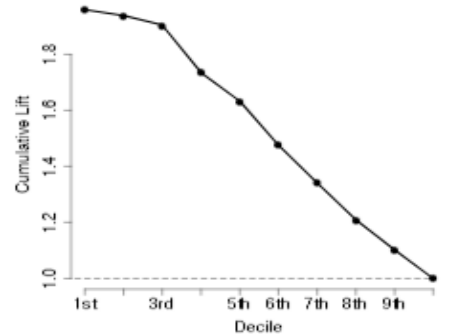
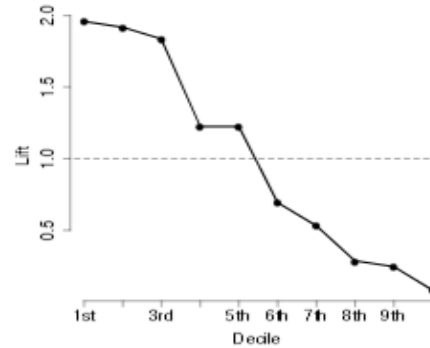
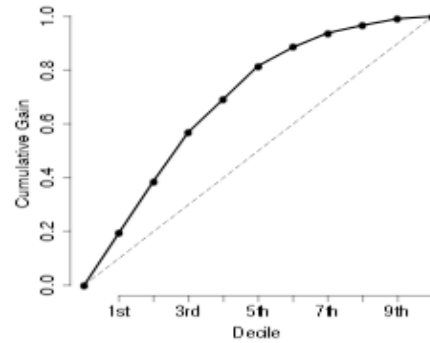
- The resulting K-S statistics for Model 1, 2, 3 and 4 are 0.940, 0.631, 0.432, and 0.164, respectively
- **Model 1 is doing a much better job** of separating the two target levels than the other models.

Cum. Gain, Lift, and Cum. Lift Charts on Email Classification

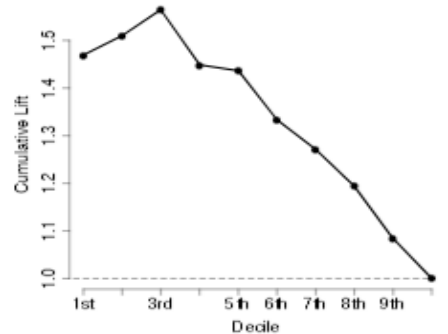
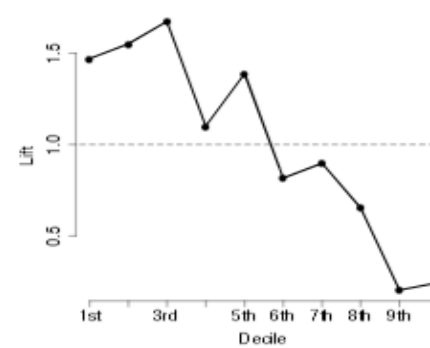
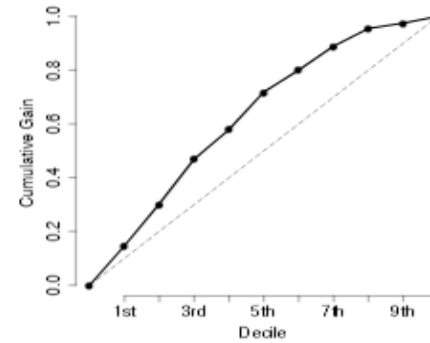
(a) Model 1



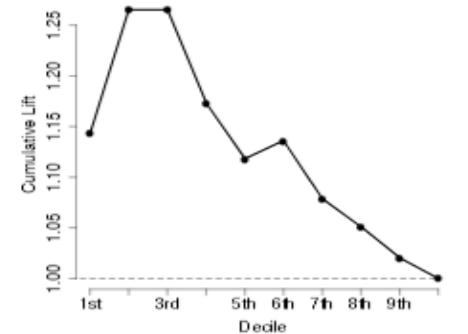
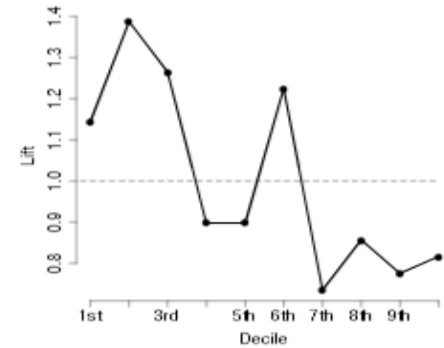
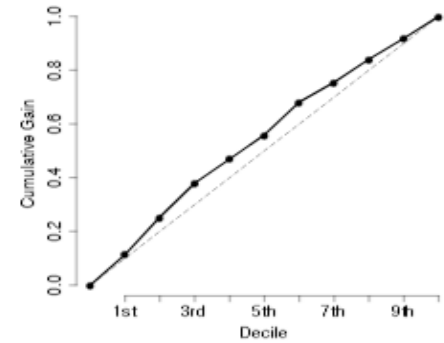
(b) Model 2



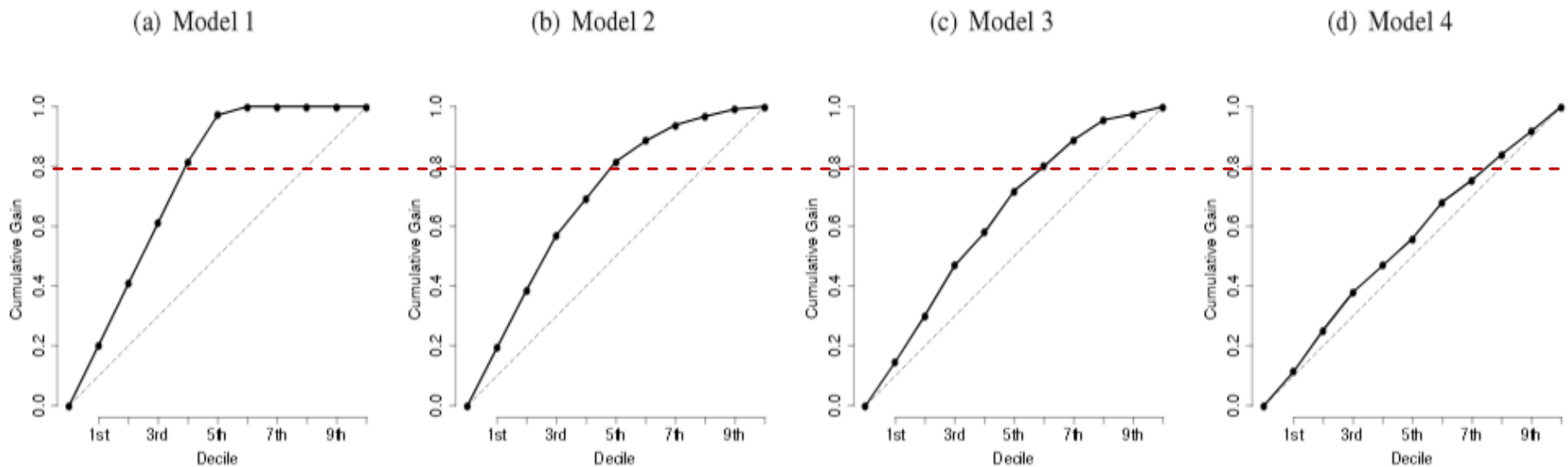
(c) Model 3



(d) Model 4



Model Comparison using Cumulative Gain



- For Model 1, 80% of the spam messages are identified in the top 40% of the model predictions.
- For Model 2, we need to look almost as far as the top 50% of predictions to find the same percentage of spam messages.
- For Models 3 and 4, we need to go as far as 60% and 75% respectively.
- **Model 1 distinguishes between the target levels most effectively.**

Model Comparison using Mean Squared Error

- The mean squared error (MSE) allows us to rank the performance of multiple models on a prediction problem with a continuous target.
- In this example, the regression model is more accurately predicting the target value than the nearest neighbor model

ID	Target	Linear Regression		k-NN	
		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