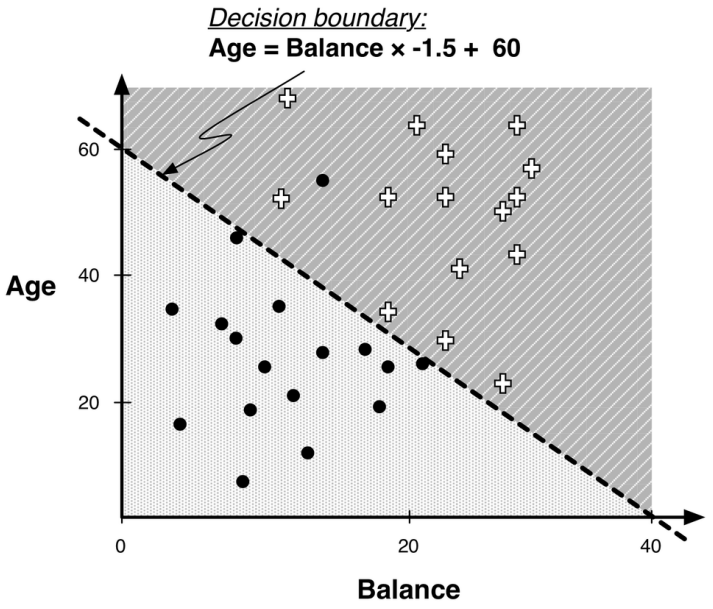
DI501 Introduction to Data Informatics

Lecture 5

Classification- Part2

Parametric Models for Classification

- Use a model to separate the data into regions with different values of the target variable.
- We can specify the structure of the model and estimate the parameters of the model.
  - E.g., a parameterized function of features.
- The parameters are estimated by using the training data.

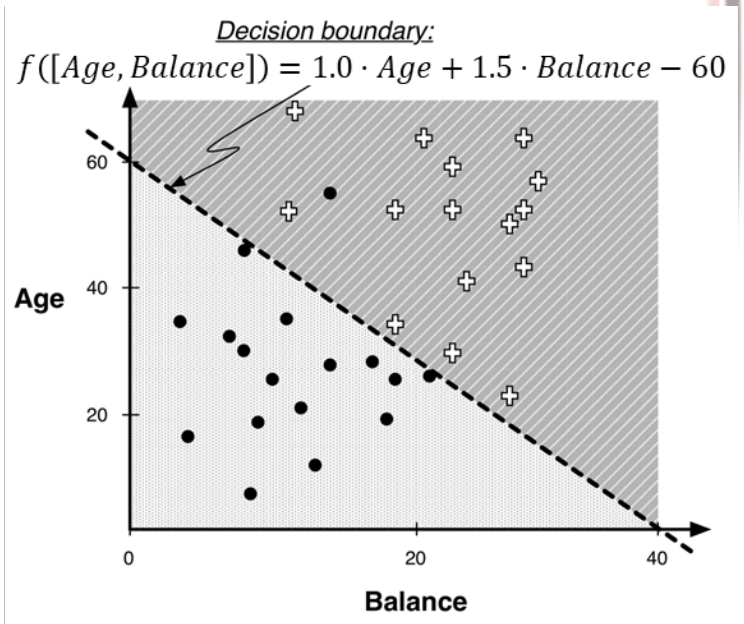


Linear Discriminant Functions

- The classification function given on the right is called a linear discriminant, because
  - it discriminates between the classes,
  - the decision boundary is a linear combination of attributes.
- A general linear model for a feature vector $\mathbf{x} = [x_1, x_2, \ldots]$ is

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots$$

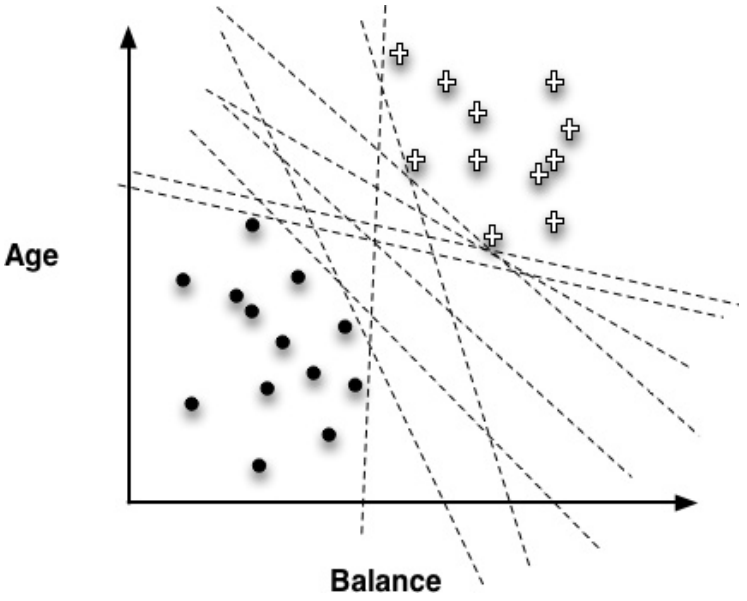
- This function corresponds to a line in 2D, a plane in 3D, a hyperplane in higher dimensions.



$$class(\mathbf{x}) = \begin{cases} + & \text{if } 1.0 \bullet Age + 1.5 Balance - 60 > 0 \\ \circ & \text{if } 1.0 \bullet Age + 1.5 Balance - 60 < 0 \end{cases}$$

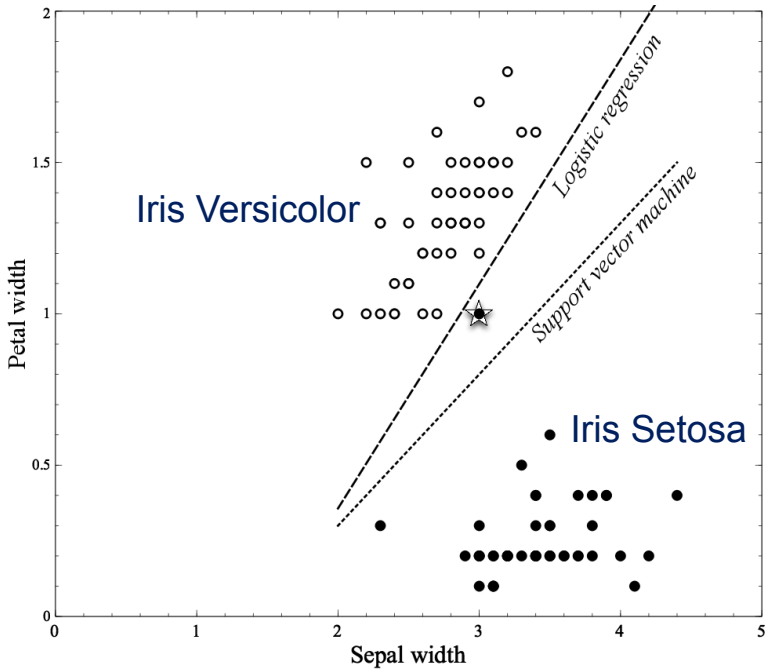
Linear Discriminant Functions

- Unfortunately, it's not trivial to choose the "best" line to separate the classes.
- Many different possible linear boundaries can separate the two groups.
- Which should we pick?
- What should be our objective in choosing parameters?



Linear Discriminant Functions

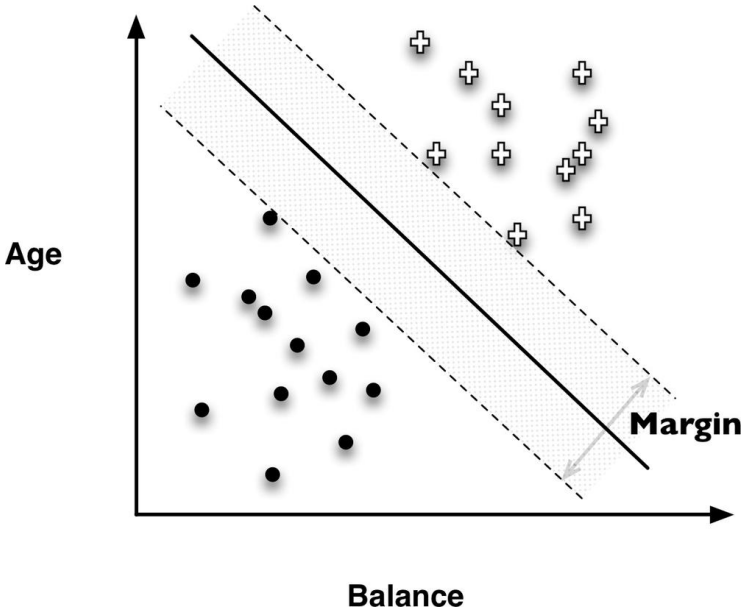
- There are several choices such as
  - Logistic Regression
  - Support Vector Machine
- Different methods produce different boundaries because they are optimizing different functions.



Two different classification models for just two species of irises, Iris Setosa and Iris Versicolor, based on two attributes.

Support Vector Machine (SVM)

- SVMs choose the discriminant line as the center of the bar maximizes the margin between the classes
  - That is, the width of the bar that can be placed between the classes is maximized.
- SVMs also can handle non-linearly separable data by penalizing errors or by using a more complicated similarity function ("kernel").

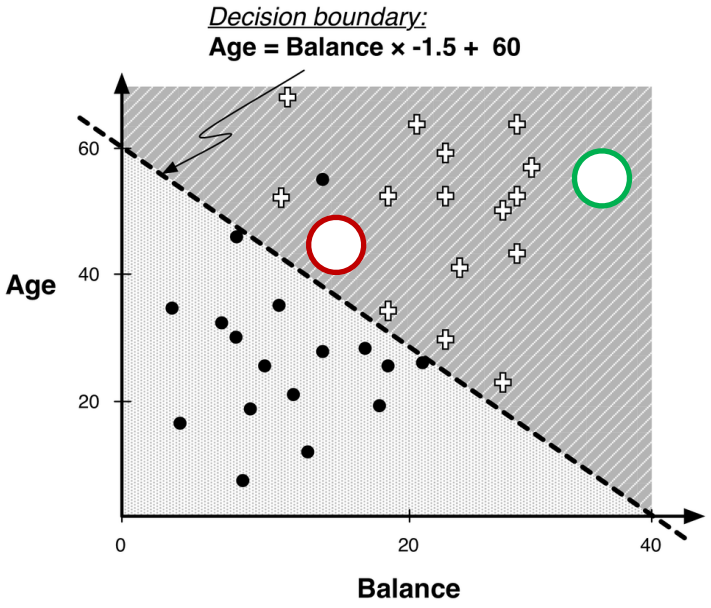


Linear Discriminant Functions for Scoring and Ranking Instances

- Sometimes, we want to predict how likely an instance belongs to a class instead of predicting whether an instance belongs to the class or not.
  - For example,
    - which consumers are most likely to respond to this offer?
    - which customers are most likely to leave when their contracts expire?
- In such cases, we can consider using models to estimate class membership probability.
- Alternatively, the model can provide a score that will rank instances by the likelihood of belonging to one class or the other.
  - We don't need to estimate the probabilities exactly as long as instances are ranked reasonably well.

Linear Discriminant Functions for Scoring and Ranking Instances

- The output of the linear discriminant function can give an intuitively satisfying ranking of the instances by their likelihood of belonging to the class of interest.
  - The farther from the line, the more likely the corresponding class. (but the relationship need not be linear)
- Suppose we have a new instance to classify
  - If the instance is near to decision boundary (e.g., “a”), we would be uncertain about the classification result.
  - If the instance is far away from the decision boundary (e.g., “b”), we would expect the highest likelihood of the classification result.

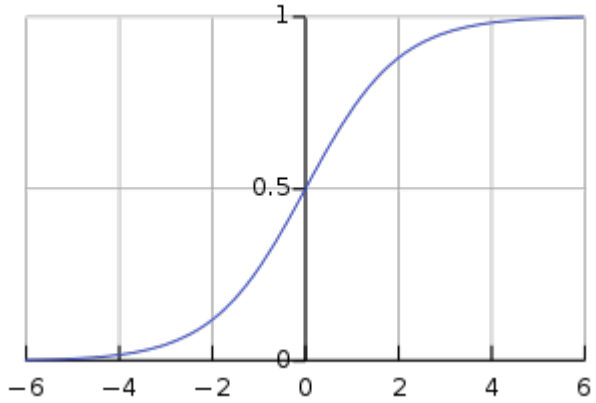


Logistic Regression

- Logistic regression turns linear predictions into probabilities by using the logistic (sigmoid) function.
- The logistic function:

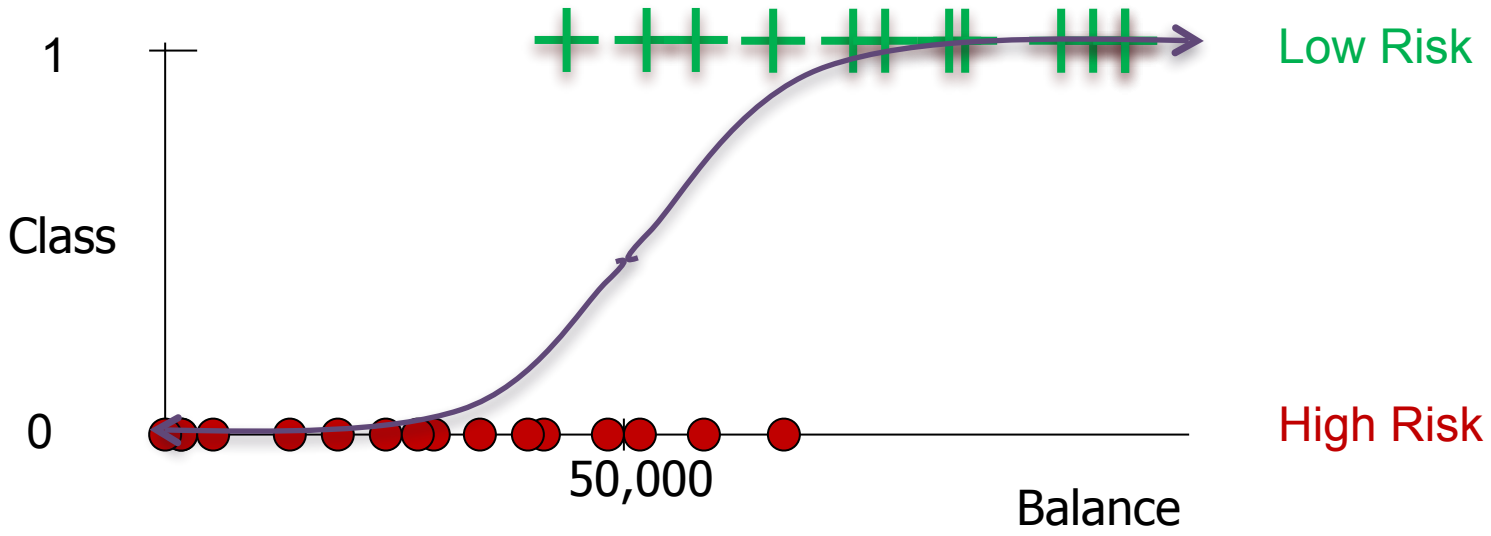
$$S(x) = \frac{1}{1 + e^{-x}}$$

- Sigmoid function projects $(-\infty, +\infty)$ to $[0, 1]$



Logistic Regression: A Simple Example

- *Estimate the probability of credit risk*



$$p_{LowRisk}(Balance) = \frac{1}{1 - e^{-(Balance - 50{,}000)}}$$

Logistic Regression

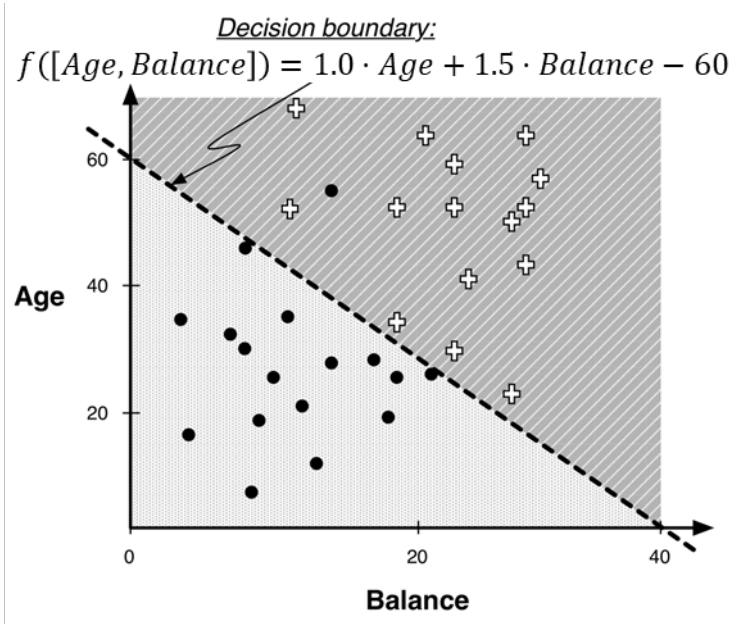
- Consider the linear discriminant function for a feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_m]$ of m features

$$f(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_m x_m$$

- $f(\mathbf{x})$ gives the distance from the separating boundary for the binary target (two classes) with labels $+$ and $-$.
- Let $p_+(\mathbf{x})$ denotes the probability of the instance represented by feature vector $\mathbf{x}$ is $+$ and $p\_(\mathbf{x}) = 1 - p_+(\mathbf{x})$ for class $-$.

$$p_+(\mathbf{x}) = S\big(f(\mathbf{x})\big) = \frac{1}{1 + e^{-f(\mathbf{x})}}$$

Logistic Regression Classifier



- We actually want to learn whether $p_+(\mathbf{x})$ is larger than $p_\circ(\mathbf{x})$ or not. Hence, in order to classify the instance as + we should have:

$$\frac{p_+(\mathbf{x})}{p_\circ(\mathbf{x})} = \frac{p_+(\mathbf{x})}{1 - p_+(\mathbf{x})} > 1$$

- By rearranging terms,

$$log\left(\frac{p_+(\mathbf{x})}{1 - p_+(\mathbf{x})}\right) > 0 \quad \Leftrightarrow \quad log\left(\frac{\frac{1}{1+e^{-f(\mathbf{x})}}}{\frac{e^{-f(\mathbf{x})}}{1+e^{-f(\mathbf{x})}}}\right) > 0$$

$$\log\left(e^{f(\mathbf{x})}\right) > 0 \quad \Leftrightarrow \quad f(\mathbf{x}) > 0$$

- Therefore, we can classify the instance represented by feature vector $\mathbf{x}$ as:

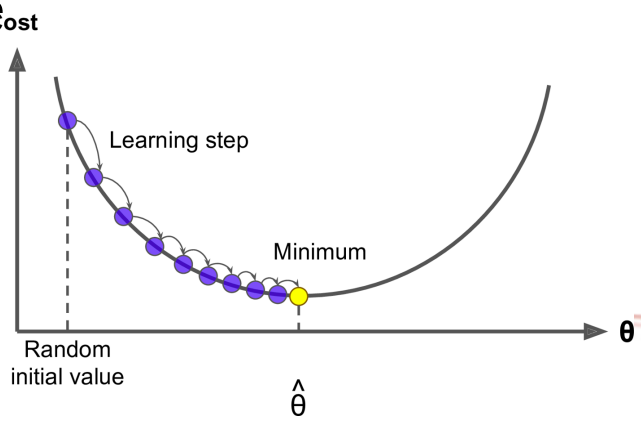
$$class(\mathbf{x}) = \begin{cases} + & f(\mathbf{x}) > 0 \\ \circ & otherwise \end{cases}$$

Fitting a Logistic Regression Model

- We estimate parameters $\mathbf{w} = [w_0, w_1, \ldots, w_m]$.
- We maximize the Log Likelihood for a training data set of n instances with feature vectors $\mathbf{x_i}$ and classes $y_i \in \{0,1\}$ for $\mathrm{i} = 1, \ldots, \mathrm{n}$.

$$L(\mathbf{w}) = \frac{1}{n}\sum_{i=0}^{n}\left[y_i \bullet \log\left\{p\left(y_i = 1 \middle| \mathbf{x_i}, \mathbf{w}\right)\right\} + (1 - y_i) \bullet \log\{p\left(y_i = 0 \middle| \mathbf{x_i}, \mathbf{w}\right)\}\right]$$

- Regularization can be used to overcome overfitting.
- There is no closed form solution.
- We can use gradient descent:
  - Start with an arbitrary $\mathbf{w}$.
  - Update $\mathbf{w}$ based on gradients.



Logistic Regression:

Multiclass Classification

- We can reduce a multi-class problem into multiple binary classification problems.
- Method 1: One-versus-rest (aka. one-versus-all)
  - Given m classes, train m classifiers: one for each class
  - Classifier k: treat tuples in class k as positive & all the rest as negative
  - To classify a tuple X, the set of classifiers vote as an ensemble
- Method 2: one-versus-one (aka. all-versus-all):
  - Learn a classifier for each pair of classes
  - Given m classes, construct $m(m-1)/2$ binary classifiers
  - A classifier is trained using tuples of the two classes
  - To classify a tuple X, each classifier votes
  - X is assigned to the class with maximal vote

Logistic Regression:

Categorical Variables

- How can we incorporate features with categorical values into a logistic regression model?
- We can create indicator (dummy) variables.
  - Two-categories: e.g., true/false, male/female, employed/ not employed, rent/own.
    - Assign 1 to one category and 0 to other category.
  - More than two categories: e.g., north/south/west/east, red/green/blue/yellow
    - One-Hot Encoding: $(c-1)$ indicator variables for c categories.
      - One of the variables is 1 and the rest is 0 for the first $c-1$ categories, and all variables are 0 for the c^{th} category.

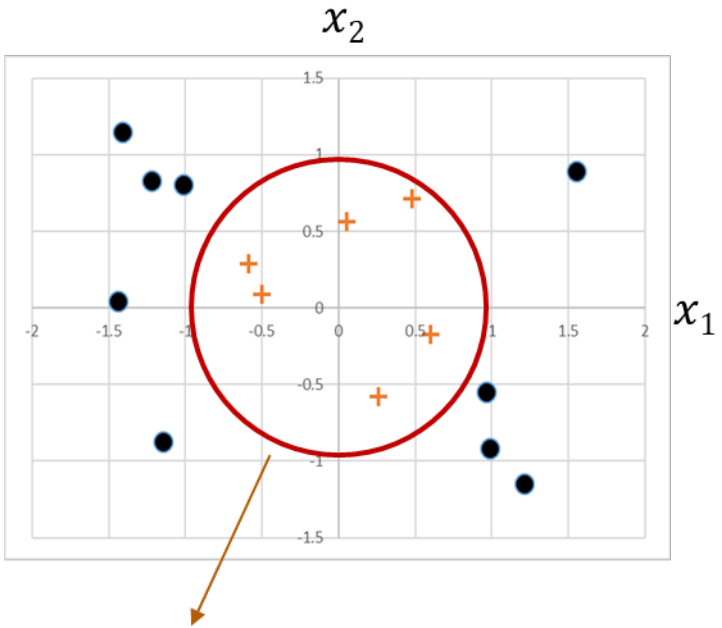
Logistic Regression:

Polynomial Features

- We can define non-linear decision boundaries by using polynomial expansion.
- For example, we can create a circular decision boundary for two features $[x1, x2]$ a by using:

$$f([x_1, x_2]) = x_1^2 + x_2^2 - r^2$$

- More complex decision boundaries can be defined by using higher order polynomials.



Generative vs. Discriminative Classifiers

- X: observed variables (features)
- Y: target variables (class labels)
- A generative classifier models p(Y, X) models how the data was "generated“? "what is the likelihood this or that class generated this instance?" and pick the one with higher probability
  - E.g., Naïve Bayes
- A discriminative classifier models p(Y|X) use the data to create a decision boundary
  - E.g., Decision Trees, Logistic Regression, SVM

Further Comments on Discriminative Classifiers

- Strength
  - Prediction accuracy is generally high
    - As compared to generative models
    - Robust, works when training examples contain errors
  - Fast evaluation of the learned target function
- Criticism
  - Long training time
  - Difficult to understand the learned function (weights)
    - Bayesian networks can be used easily for pattern discovery
  - Not easy to incorporate domain knowledge
    - Easy in the form of priors on the data or distributions

Classifier Evaluation Metrics:

Accuracy

$$accuracy = \frac{\#\ of\ correct\ decisions\ made}{\#\ of\ decisions\ made}$$

$$error\ rate = 1 - accuracy$$

- Classification accuracy is a popular metric because it's very easy to measure and it reduces classifier performance to a single number.
- Unfortunately, it is usually too simplistic for real business problems.

Classifier Evaluation Metrics:

Problems with Unbalanced Classes

- We need to think carefully about model evaluation if one class is rare.
  - Classifiers often are used to sift through a large population of normal entities in order to find a relatively small number of unusual ones.
    - e.g., checking an assembly line for "defective" parts.
  - The class distribution will be unbalanced or skewed when the unusual class is rare among the general population.
- If the class distribution is skewed, evaluation based on accuracy breaks down.
  - If only 0.1% of parts are "defective", a model classifying everything as "not defective" will have 99.9% accuracy!

Classifier Evaluation Metrics:

Problems with Unequal Costs and Benefits

- Classification accuracy tacitly assumes that all kinds of classification errors are equally important.
  - i.e., it does not make any distinction between errors.
- In some cases, different kinds of errors have very different costs. For example,
  - A patient was wrongly informed that he has a disease when he does not --> might be expensive, inconvenient and stressful for the patient but it would not be life threatening.
  - A patient has a disease but she is wrongly told that she does not --> could have far more serious consequences.

Classifier Evaluation Metrics:

Confusion Matrix

- Each instance in a test set has an actual class label as well as the label predicted by the classifier.
- A confusion matrix for a problem involving n classes is an $n \times n$ matrix.
  - columns: actual classes
  - rows: predicted classes
- A confusion matrix makes explicit how one class is being confused for another.

A three class example:

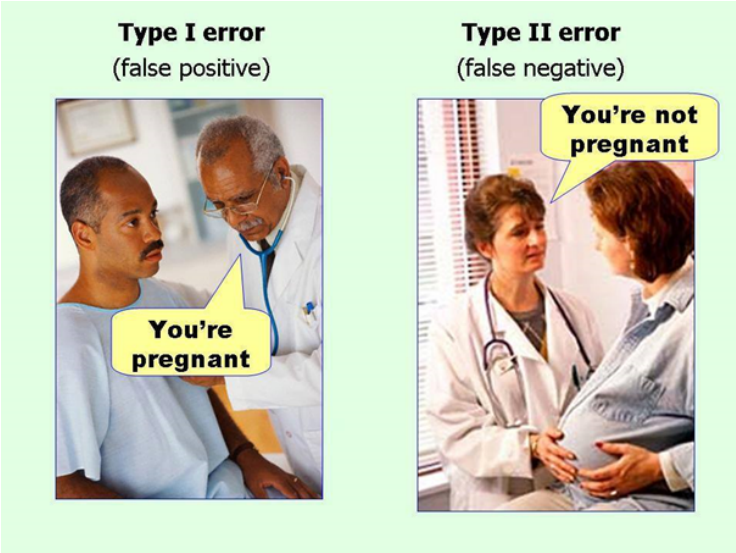
actual \ prediction	a	b	c	Total
a	n_{aa}	n_{ab}	n_{ac}	N_a
b	n_{ba}	n_{bb}	n_{bc}	N_b
c	n_{ca}	n_{cb}	n_{cc}	N_c
Total	n_a	n_b	n_c	N

$$accuracy = \frac{n_{aa} + n_{bb} + n_{cc}}{N}$$

Classifier Evaluation Metrics:

Binary Classification

- The true positives (TP) and true negatives (TN) are correct classifications.
- A false positive (FP) occurs when the outcome is incorrectly predicted as yes (or positive) when it is actually no (negative).
- A false negative (FN) occurs when the outcome is incorrectly predicted as negative when it is actually positive.



		Predicted class: yes	Predicted class: no
Actual class	yes	true positive	false negative
	no	false positive	true negative

Classifier Evaluation Metrics:

Binary Classification

- A false positive is where you receive a positive result for a test, when you should have received a negative result.
- It's sometimes called a "false alarm" or "false positive error."
- It's usually used in the medical field, but it can also apply to other arenas.
  - A cancer screening test comes back positive, but you don't have the disease.
  - Virus software on your computer incorrectly identifies a harmless program as a malicious one.
- A false negative is where a negative test result is wrong. In other words, you get a negative test result, but you should have got a positive test result.
  - Quality control in manufacturing; a false negative in this area means that a defective item passes through the cracks.
  - In software testing, a false negative would mean that a test designed to catch something (i.e. a virus) has failed.

Classifier Evaluation Metrics:

Binary Classification

Binary performance metrics are categorized into four:

- *Confusion-matrix derived instruments* such as true positive rate (TPR), F1, MCC
- *Entropy-based instruments* (a subset of confusion-matrix derived instruments) such as mutual information (*MI*), outcome entropy (*HO*), class entropy (*HC*), joint entropy (*HOC*), and normalized mutual information (*nMI*);
- *Graphical performance metrics* such as area-under-ROC-curve (*AUCROC*, *ROC*: receiver operating characteristic) or area-under-precision-recall-curve (*AUCPR*);
- *The instruments based on a probabilistic interpretation of classification error* such as mean squared error (*MSE*, also known as Brier score), mean absolute error (*MAE*), root mean square error (*RMSE*), and *LogLoss* (also known as binary cross-entropy or relative entropy).

Classifier Evaluation Metrics:

Binary Classification Metrics

- Accuracy: Recognition rate.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- Precision: Exactness: What % of instances that the classifier labeled as positive are actually positive?

$$precision = \frac{TP}{TP + FP}$$

- Recall: Completeness: what % of positive instances did the classifier label as positive?
- Sensitivity: True positive recognition rate.

$$recall = sensitivity = \frac{TP}{TP + FN}$$

- Specificity: True negative recognition rate.

$$specificity = \frac{TN}{TN + FP}$$

		Predicted class			
		yes		no	
Actual class	yes	true positive	TP	FN	false negative
	no	false positive	FP	TN	true negative

Classifier Evaluation Metrics:

Binary Classification Metrics

- Recall:

$$recall = \frac{TP}{TP + FN}$$

- F measure: harmonic mean of precision and recall

$$F1\ measure = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}$$

F_1 score gives equal importance to precision and recall

- $$F\beta\ measure = (1 + \beta^2)\frac{precision \times recall}{(\beta^2 \times precision) + recall}$$

- $F\beta$ uses a positive real factor β, where β is chosen such that recall is considered β times as important as precision
- Matthews Correlation Coefficient:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Classifier Evaluation Metrics:

Binary Classification Example

Actual Class\Predicted class	disease = yes	disease = no	Total	Recognition(%)
disease = yes	**90 (TP)**	**210 (FN)**	300	30.00 (*sensitivity)*
disease = no	**140 (FP)**	**9560 (TN)**	9700	98.56 (*specificity)*
Total	230	9770	10000	96.40 (*accuracy*)

Precision = 90/230 = 39.13% *Recall* = 90/300 = 30.00%

- Precision shows how accurate my «yes» responses are (I labeled 230 instances as «yes» and 90 out of my 230 instances turned out to be correct at the end).
- Recall (sensitivity) How good I am classifying «yes» instances (There were 300 «yes» instances but I could only label 90 of them as «yes»).

Classifier Evaluation Metrics:

Binary Classification Example

True Label (Those who are healthy)	True Label (Those who have diabetes)	N	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 Score
37	37	74	26	19	18	11	0.608	0.591	0.703	0.642
25078	300	25378	19258	176	124	5820	0.766	0.994	0.768	0.866
11836	33	11869	8716	16	17	3120	0.736	0.998	0.736	0.847

The table shows the performance of three classifiers performance on three different datasets. Comment on their performance and dataset characteristics.

Classifier Evaluation Metrics:

Binary Classification Example

True Label (Those who are healthy)	True Label (Those who have diabetes)	TP	TN	FP	FN	Accuracy	F1 Score	MCC	Baseline Accuracy
37	37	26	19	18	11	0.608	0.642	0.220	0.500
25078	300	19258	176	124	5820	0.766	0.866	0.090	0.988
11836	33	8716	16	17	3120	0.736	0.847	0.026	0.997

The table shows the performance of three classifiers performance on three different datasets. Comment on their performance based on F1 and MCC.

Baseline model is the model which predicts healthy for all data points. Compare the model performance versus baseline models.

Classifier Evaluation Metrics:

Binary Classification Example

- Given a query, a Web search engine produces a list of hits that represent documents supposedly relevant to the query.
- Compare one system that locates 100 documents, 40 of which are relevant, with another that locates 400 documents, 80 of which are relevant. There are 100 documents that are exactly relevant to the query.
- Which is better?
  - Recall= number of documents retrieved that are relevant / total number of documents that are relevant
  - Precision= number of documents retrieved that are relevant / total number of documents that are retrieved

Classifier Evaluation Metrics:

Binary Classification Example

- Compare one system that locates 100 documents, 40 of which are relevant, with another that locates 400 documents, 80 of which are relevant. There are 100 documents that are exactly relevant to the query.
- Which is better?
  - Recall= number of documents retrieved that are relevant / total number of documents that are relevant
  - Precision= number of documents retrieved that are relevant / total number of documents that are retrieved
  - System 1: Recall=40/100=0.4 Precision=40/100=0.4
  - System 2: Recall=80/100=0.8 Precision=80/400=0.2

Classifier Evaluation Metrics:

Multiclass Classification

		True/Actual		
		Finance	Politics	Parent
Predicted	Finance	40	60	30
	Politics	10	20	0
	Parent	10	20	60
	Total	60	100	90

F1-score is a function of precision and recall. Here is a summary of the precision and recall for our three classes:

Class	Precision	Recall	F1-score	TP	TN	FN	FP
Finance	0.308	0.667	0.421	40	100	20	90
Politics	0.667	0.200	0.308	20	140	80	10
Parent	0.667	0.667	0.667	60	130	30	30

		True/Actual	
		Finance	Not Finance
Predicted	Finance	40	90
	Not Finance	20	100

Classifier Evaluation Metrics:

Multiclass Classification

		True/Actual		
		Finance	Politics	Parent
Predicted	Finance	40	60	30
	Politics	10	20	0
	Parent	10	20	60
	Total	60	100	90

Class	Precision	Recall	F1-score	TP	TN	FN	FP
Finance	0.308	0.667	0.421	40	100	20	90
Politics	0.667	0.200	0.308	20	140	80	10
Parent	0.667	0.667	0.667	60	130	30	30

Let's combine them into one score:

Macro-averaged F1-score, or the macro-F1 for short is computed as a simple arithmetic mean of our per-class F1-score.

Macro-F1 = (42.1% + 30.8% + 66.7%) / 3 = 46.5%

In a similar way, we can also compute the ***macro-averaged precision*** and the **macro-averaged recall:**

Macro-precision = (31% + 67% + 67%) / 3 = 54.7%

Macro-recall = (67% + 20% + 67%) / 3 = 51.1%

There is an another Macro-F1 metrics, which is calculated as:

Macro-F1*= 2 × (54.7% × 51.1%) / (54.7% + 51.1%) = 52.8%

Python's sklearn library provides the sklearn.metrics.f1_score function, which computes Macro-F1 (and not Macro-F1*).

Classifier Evaluation Metrics:

Multiclass Classification

		True/Actual		
		Finance	Politics	Parent
Predicted	Finance	40	60	30
	Politics	10	20	0
	Parent	10	20	60
	Total	60	100	90

Class	Precision	Recall	F1-score	TP	TN	FN	FP
Finance	0.308	0.667	0.421	40	100	20	90
Politics	0.667	0.200	0.308	20	140	80	10
Parent	0.667	0.667	0.667	60	130	30	30

When averaging the macro-F1, we gave equal weights to each class. We don't have to do that: in weighted-average F1-score, or weighted-F1, we weight the F1-score of each class by the number of samples from that class.

$$\text{Weighted-F1} = (60 \times 42.1\% + 100 \times 30.8\% + 90 \times 66.7\%) / 250 = 46.4\%$$

Classifier Evaluation Metrics:

Multiclass Classification

		True/Actual		
		Finance	Politics	Parent
Predicted	Finance	40	60	30
	Politics	10	20	0
	Parent	10	20	60
	Total	60	100	90

Class	Precision	Recall	F1-score	TP	TN	FN	FP
Finance	0.308	0.667	0.421	40	100	20	90
Politics	0.667	0.200	0.308	20	140	80	10
Parent	0.667	0.667	0.667	60	130	30	30
Total				120	370	130	130

Micro-averaged F1-score, or the **micro-F1:** first compute micro-averaged *precision* and micro-averaged *recall* over all the samples, and then combine the two.
TP= 40+20+60=120 (orange colored cells)
FP= 10+10+60+20+30+0=130 (yellow colored cells)
micro-F1 = micro-precision = micro-recall=accuracy

$$recall = \frac{TP}{TP + FN} = 120/(120+130)=48\%$$

$$precision = \frac{TP}{TP + FP} = 120/(120+130)=48\%$$

Classifier Evaluation Metrics:

Multiclass Classification

		True/Actual		
		Finance	Politics	Parent
Predicted	Finance	40	60	30
	Politics	10	20	0
	Parent	10	20	60
	Total	60	100	90

Class	Precision	Recall	F1-score	TP	TN	FN	FP
Finance	0.308	0.667	0.421	40	100	20	90
Politics	0.667	0.200	0.308	20	140	80	10
Parent	0.667	0.667	0.667	60	130	30	30
Total				120	370	130	130

Macro-averaging gives equal weight to each class, whereas micro-averaging gives equal weight to each per-document classification decision. Because the F1 measure ignores true negatives and its magnitude is mostly determined by the number of true positives, large classes dominate small classes in micro-averaging.

Micro-averaged results are therefore really a measure of effectiveness on the large classes in a test collection. To get a sense of effectiveness on small classes, you should compute macro-averaged results.

Classifier Evaluation Metrics:

Log Loss

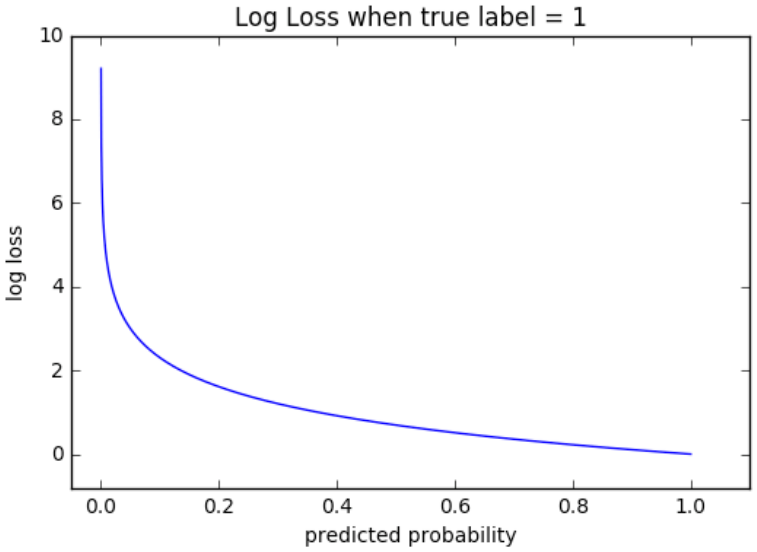
Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

- where **y** is the **label** and **p(y)** is the predicted **probability of the point** for all **N** points.
- Reading this formula, it tells you that, for each point (*y=1*), it adds *log(p(y))* to the loss. Conversely, it adds *log(1-p(y))*, that is, the **log probability of it being the other class** (*y=0*).

Classifier Evaluation Metrics:

Log Loss



In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$-(y\log(p) + (1-y)\log(1-p))$$

If $M > 2$ (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

$$-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$$

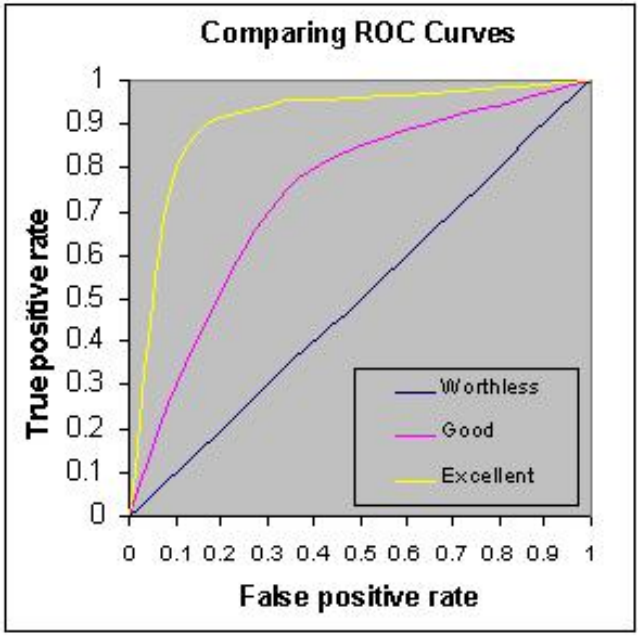
Note

- M - number of classes (dog, cat, fish)
- log - the natural log
- y - binary indicator (0 or 1) if class label c is the correct classification for observation o
- p - predicted probability observation o is of class c

Model Selection:

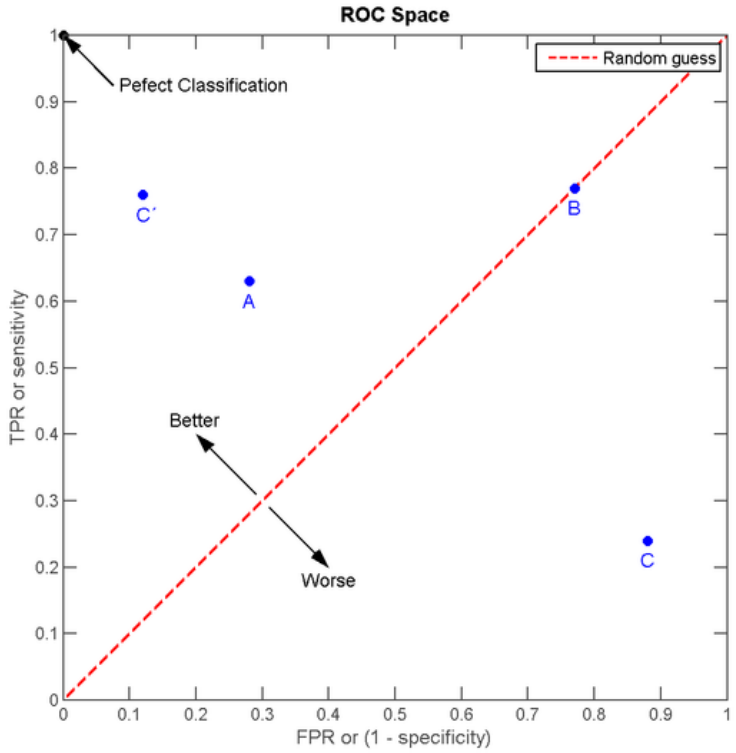
ROC Curves

- ROC (Receiver Operating Characteristics) curves: for visual comparison of classification models
- Originated from signal detection theory
- Shows the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity)
- The area under the ROC curve (AUC: Area Under Curve) is a measure of the accuracy of the model
- Rank the test tuples in decreasing order: the one that is most likely to belong to the positive class appears at the top of the list
- The closer to the diagonal line (i.e., the closer the area is to 0.5), the less accurate is the model



- Vertical axis represents the true positive rate
- Horizontal axis rep. the false positive rate
- The plot also shows a diagonal line
- A model with perfect accuracy will have an area of 1.0

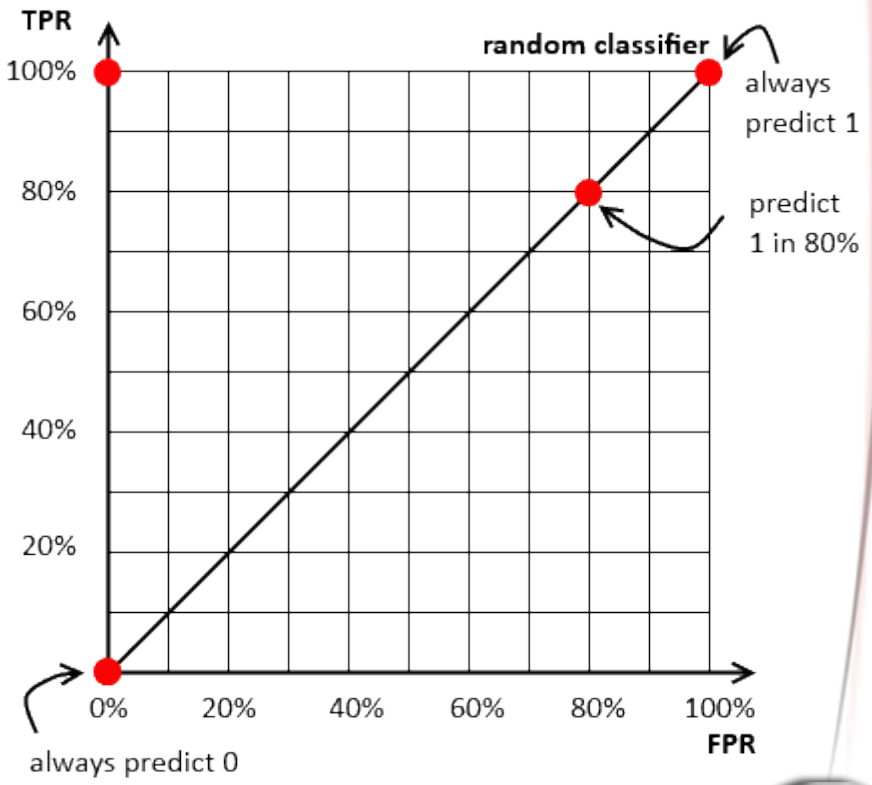
ROC Example:



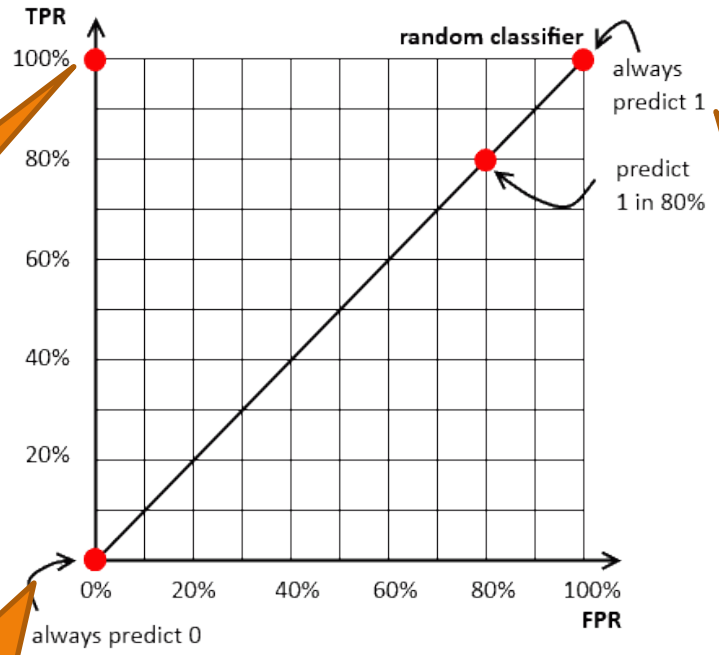
A			B			C			C′		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=76	FP=12	88
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=24	TN=88	112
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.76		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.12		
PPV = 0.69			PPV = 0.50			PPV = 0.21			PPV = 0.86		
F1 = 0.66			F1 = 0.61			F1 = 0.22			F1 = 0.81		
ACC = 0.68			ACC = 0.50			ACC = 0.18			ACC = 0.82		

ROC

- Baseline: a random classifier that predicts 1 with some probability
- On the right we have 3 random classifiers:
  - always predict 0 (0% chance to predict 1)
  - predict 1 in 80% cases
  - always predict 1 (in 100% cases)



ROC



Get everything perfect! this perfect classier commits no false positive errors and gets all true positives

Unconditionally issue positive classification! Such a classier predicts all positive instances correctly but at the cost of predicting all negative instances wrongly.
TPR=1 since it predicted all the positive instances correctly.
FPR=1 since it predicted all the negative instances as positive.

Never issue a positive classification! Such a classifier commits no false positive errors but also gains no true positives.
TPR=0 since TP is zero, it didn't predict all the positive instances correctly.
FPR=0 since FP is zero, there is no negative instance predicted as positive.

ROC: Example

Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

ROC: Example

Inst#	Class	Score	Inst#	Class	Score
1	p	.9	11	p	.4
2	p	.8	12	n	.39
3	n	.7	13	p	.38
4	p	.6	14	n	.37
5	p	.55	15	n	.36
6	p	.54	16	n	.35
7	n	.53	17	p	.34
8	n	.52	18	n	.33
9	p	.51	19	p	.30
10	n	.505	20	n	.1

Class label shows the real labels.
Score shows the probability of the instance belonging to positive class which was calculated by the model.

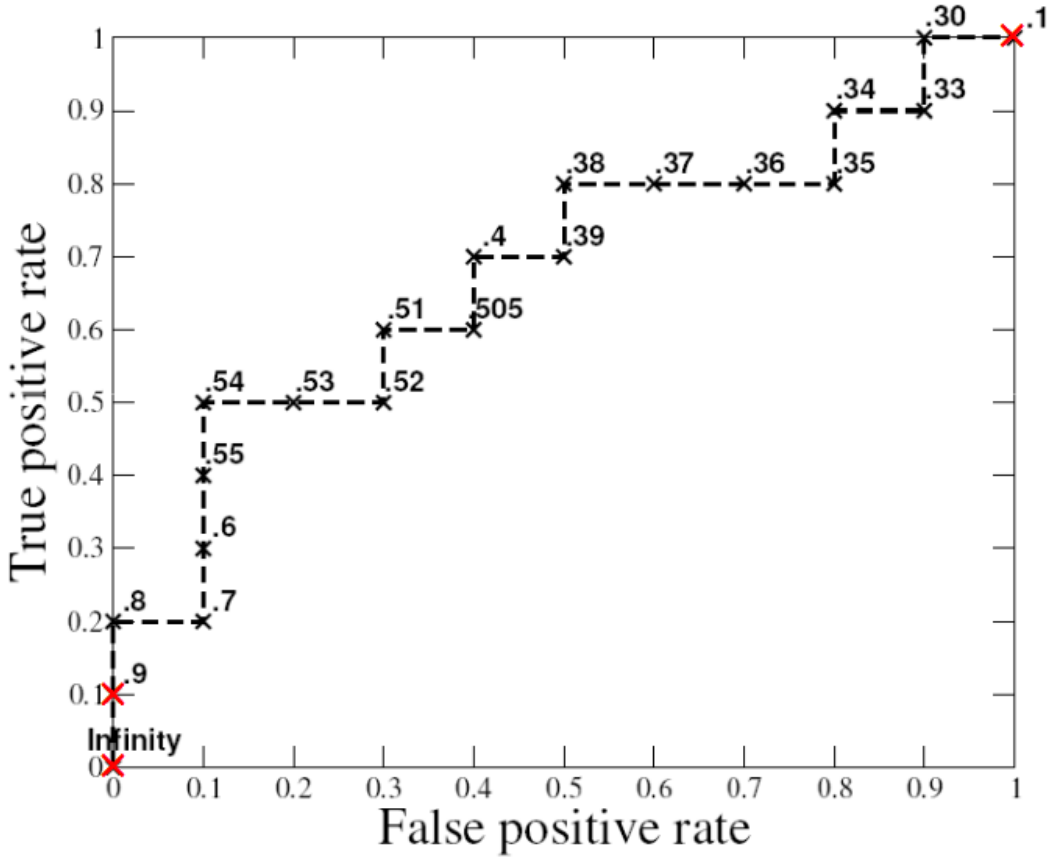
The number of instances for Actual class=No (n) is 10.
The number of instances for Actual class=Yes (p) is 10.

We first ranked the tuples according to their scores in decreasing order.

Assume that logistic regression classified the first two positive instances as positive (TPR=2/10=0.2) and one negative instance as positive (FPR=1/10=0.1) when the threshold>=0.7).

We mark this point on the ROC curve.
We iterate this process for all instances.

ROC: Example

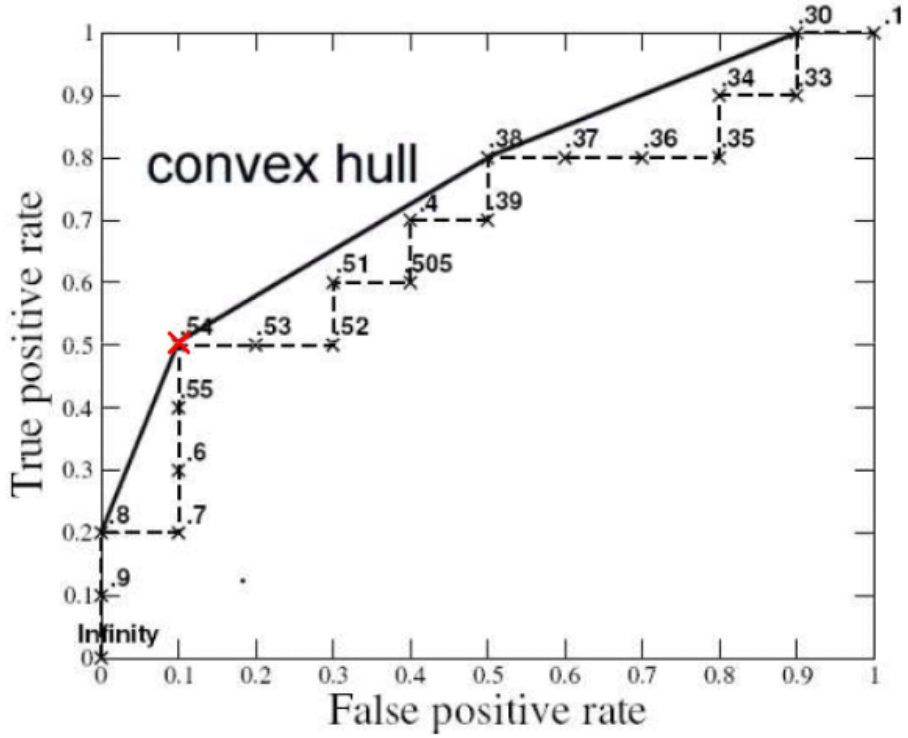


A threshold of +inf produces the point (0, 0).

As we lower the threshold to 0.9 the first positive instance is classified positive, classified positive, yielding (0, 0.1).

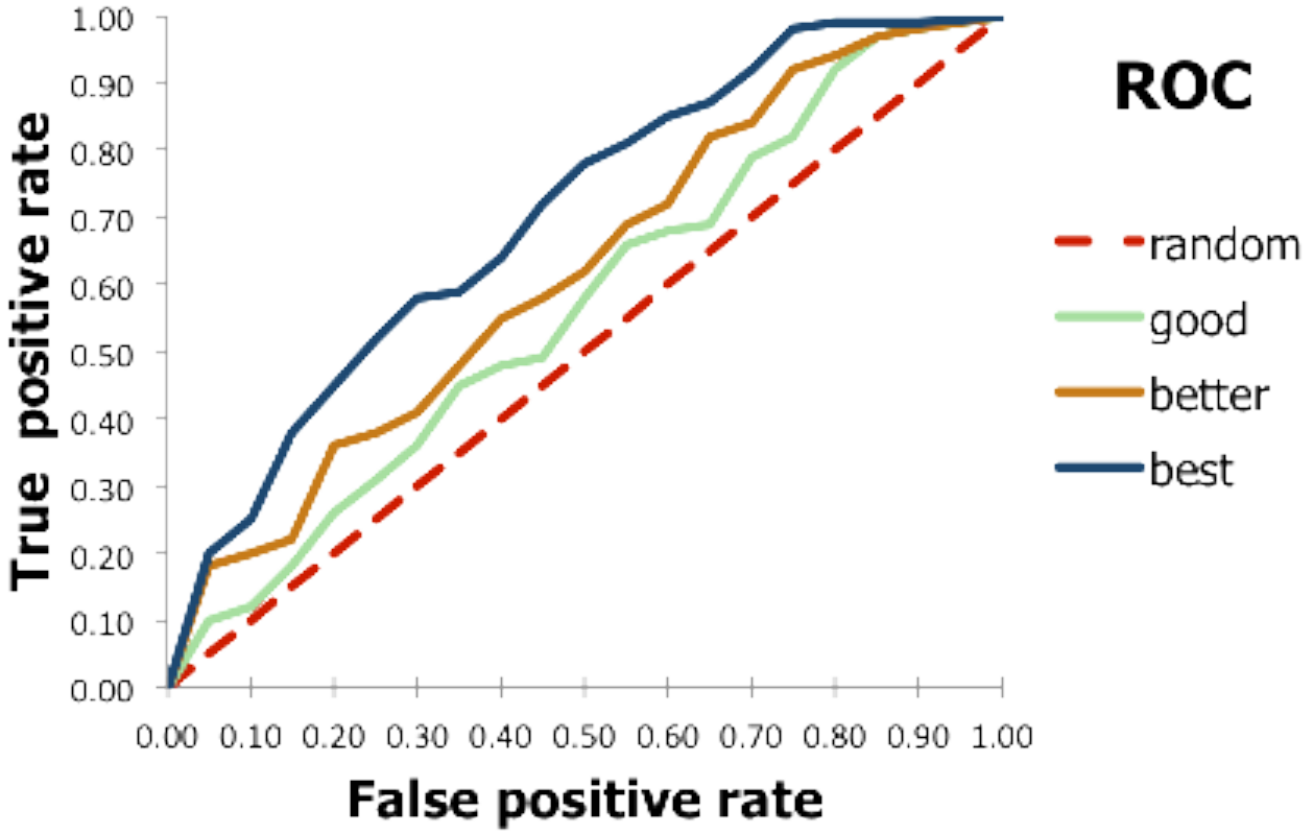
As the threshold is further reduced, the curve climbs up and to the right, ending up at (1, 1) with a threshold of 0.1.

ROC: Example



The ROC point at (0.1, 0.5) produces its highest accuracy (70%)
Note that the classifier's best accuracy occurs at a threshold of .54, rather than at .5 as we might expect with a balanced class distribution.

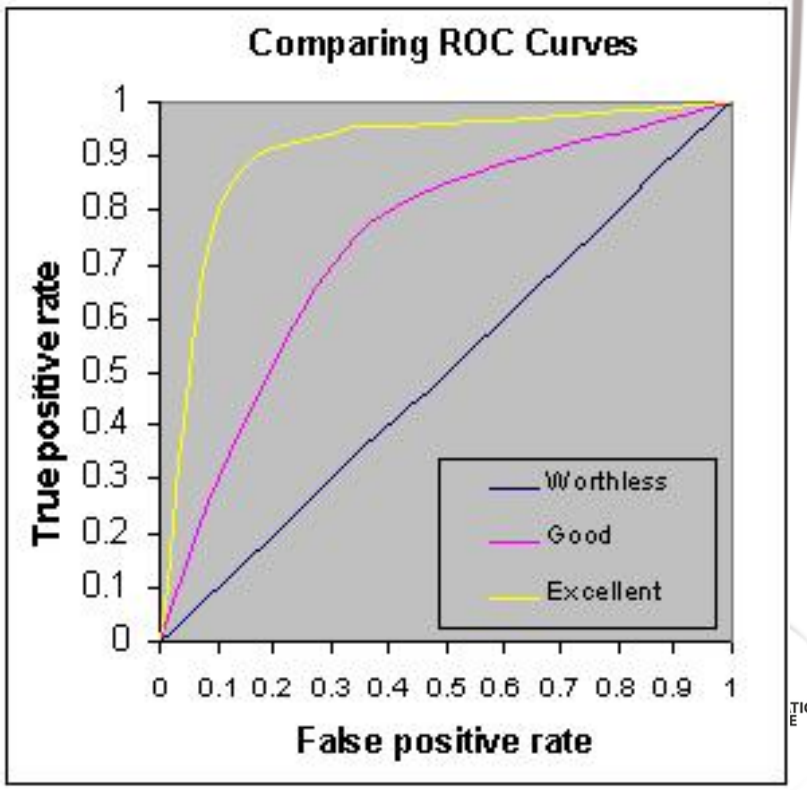
ROC: Example


True positive rate
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
0.00 0.10 0.20 0.30 0.40 0.50 0.60 0.70 0.80 0.90 1.00
False positive rate
ROC
random
good
better
best

The Area Under Curve

Accuracy is measured by the area under the ROC curve. An area of 1 represents a perfect test; an area of .5 represents a worthless test. A rough guide for classifying the accuracy of a diagnostic test is the traditional academic point system:

- .90-1 = excellent (A)
- .80-.90 = good (B)
- .70-.80 = fair (C)
- .60-.70 = poor (D)
- .50-.60 = fail (F)



Imbalanced Datasets: Important Metrics

- Problem: Accuracy Paradox
- Matthews Correlation Coefficient
- LogLoss
- Balanced Accuracy: simple arithmetic mean per-class accuracy measurements of all classes (i.e., each class' accuracy contributes evenly to the overall accuracy).

Case Study:

Feature Sets	Training F1 Score	Testing F1 score
A	0.926	0.489
B	0.791	0.49
C	0.984	0.48
D	0.64	0.327
E	1	0.383
F	0.966	0.36

The following scores were obtained from models using different set of features. The cross validation was performed for each model.

- Comment on the table in detail.
- Assume that you obtained these results using (i) kNN or (ii) Decision trees.

How can you improve the performance of the model in each case?

Case Study:

Predicted \ Truth	Altındağ	Çankaya	Etimesgut	Gölbaşı	Keçiören	Mamak	Sincan	Yenimahalle
Altındağ	0	0	0	0	0	0	0	0
Çankaya	9	159	35	15	24	18	6	32
Etimesgut	0	3	2	0	2	1	1	4
Gölbaşı	0	0	1	2	0	0	0	1
Keçiören	1	4	0	0	2	1	2	1
Mamak	1	0	0	0	2	0	2	0
Sincan	0	0	2	0	1	2	4	1
Yenimahalle	1	4	3	0	0	0	2	3

The above confusion table shows the district prediction results of a model. For instance, 43 instances labelled with Etimesgut in our database but only 2 were correctly estimated by the model. Which performance metrics would be most appropriate to use for reporting this case? Can you suggest a metrics which can summarise all the table with a single numeric value?

Case Study:

	1	2	3	4	5	6	7	8	9	10
Minimum	42.96	43.47	42.61	43.59	41.97	43.00	43.72	43.32	43.59	43.20
Maximum	44.56	46.06	45.56	48.42	45.67	45.06	45.56	46.06	45.81	46.20
Average	**43.62**	44.40	43.93	44.95	43.63	44.00	44.37	44.47	44.50	44.59
Std. dev.	0.4784	0.8111	0.9442	1.4576	1.0244	0.6343	0.6624	0.9954	0.6714	1.0952

Consider the following table which shows the performance results of the decision tree models. Comment on the performance of the first and fourth models in terms of bias and variance by comparison.