UC DAVIS COLLEGE OF ENGINEERING

ELECTRICAL & COMPUTER ENGINEERING

EEC193A

Overview of ML models

Chen-Nee Chuah

Robust & Ubiquitous Networking Lab

http://www.ece.ucdavis.edu/rubinet

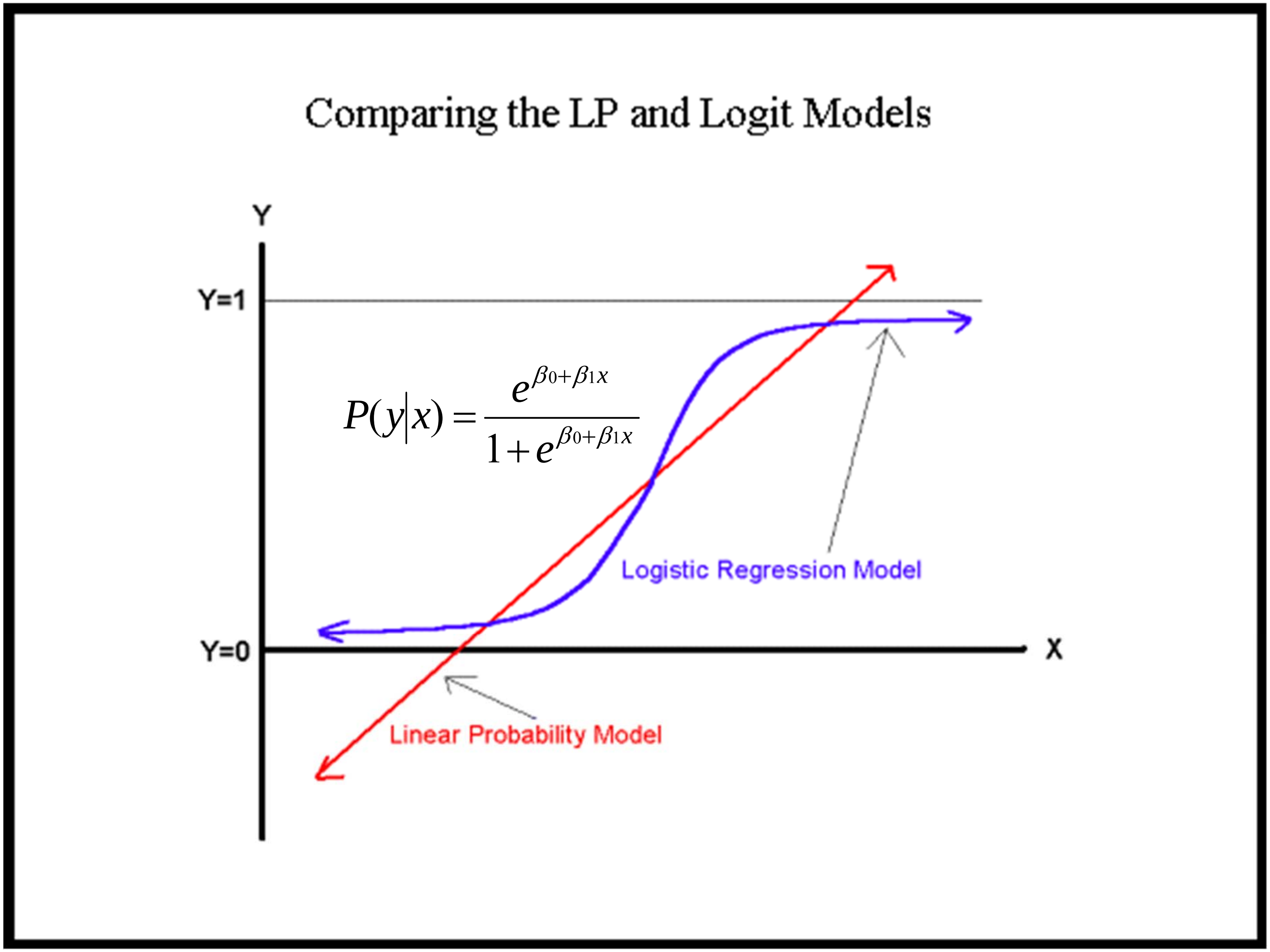
Data Analytics Span Many Areas…

- Signal Processing techniques
  - Time series analysis using Kalman Filter, Wavelet, Abrupt Change Detection, etc.
  - Image/video processing
- Statistical Learning
  - Estimation & Detection, Statistical Inferencing
  - Expectation Maximization (EM), PCA, matrix completion, Bayesian model, Hidden Markov Model (HMM), sequential hypothesis testing
- Machine Learning
  - Unsupervised learning (Clustering, PCA)
  - Supervised learning (SVM, Random Forest, etc.)
  - Deep Learning (Recurrent Neural Network, Convolutional Deep Neural Network, etc.)

Linear and Logistic Regression

- Problem: observe X (continuous or discrete) and try to predict dependent variable Y (discrete)
- Binary logistic regression: Y has two possible values (e.g., 0 or 1, true or false)

Linear and Logistic Regression



The Logistic Regression Model

The "logit" model solves these problems:
$\ln[p/(1-p)] = \beta_0 + \beta_1 X$

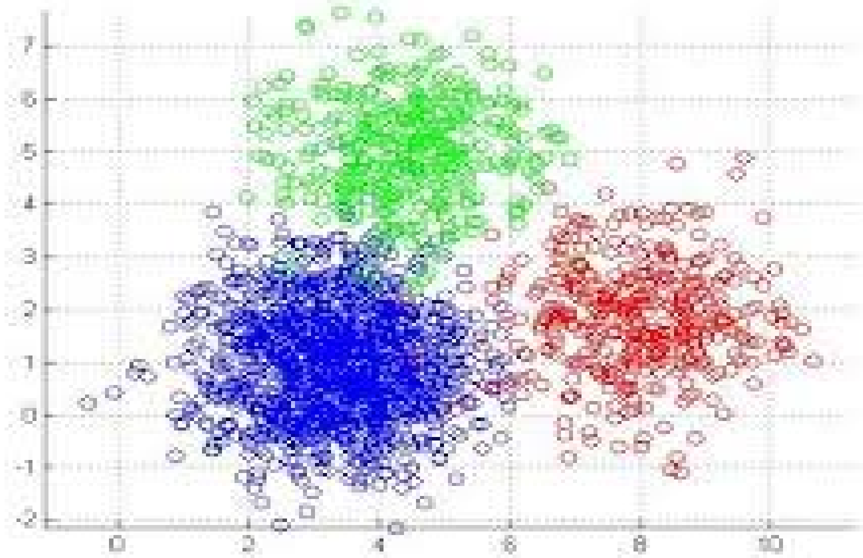
- p is the probability that the event Y occurs, p(Y=1)
  - [range=0 to 1]
- p/(1-p) is the "odds ratio"
  - [range=0 to ∞]
- ln[p/(1-p)]: log odds ratio, or "logit“
  - [range=-∞ to +∞]

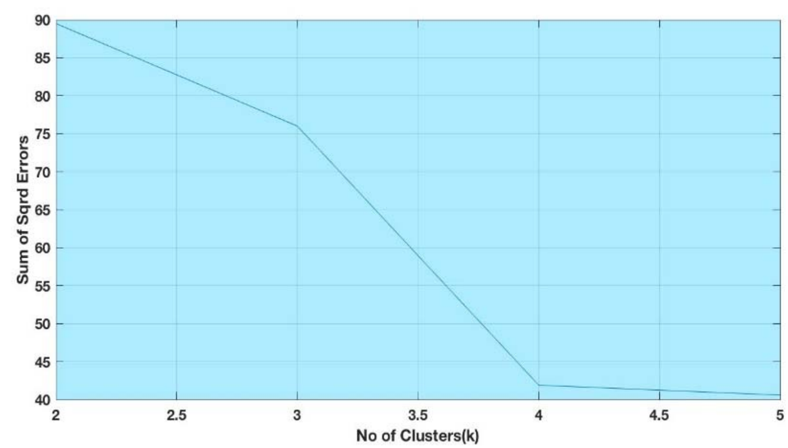
Machine Learning Models

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering e.g., K-Means clustering
Continuous	regression	dimensionality reduction e.g., PCA

Clustering

- **K-means**
  - **Iteratively re-assign points to the nearest cluster center**
- Agglomerative clustering
  - Start with each point as its own cluster and iteratively merge the closest clusters
- Mean-shift clustering
  - Estimate modes of pdf
- Spectral clustering
  - Split the nodes in a graph based on assigned links with similarity weights



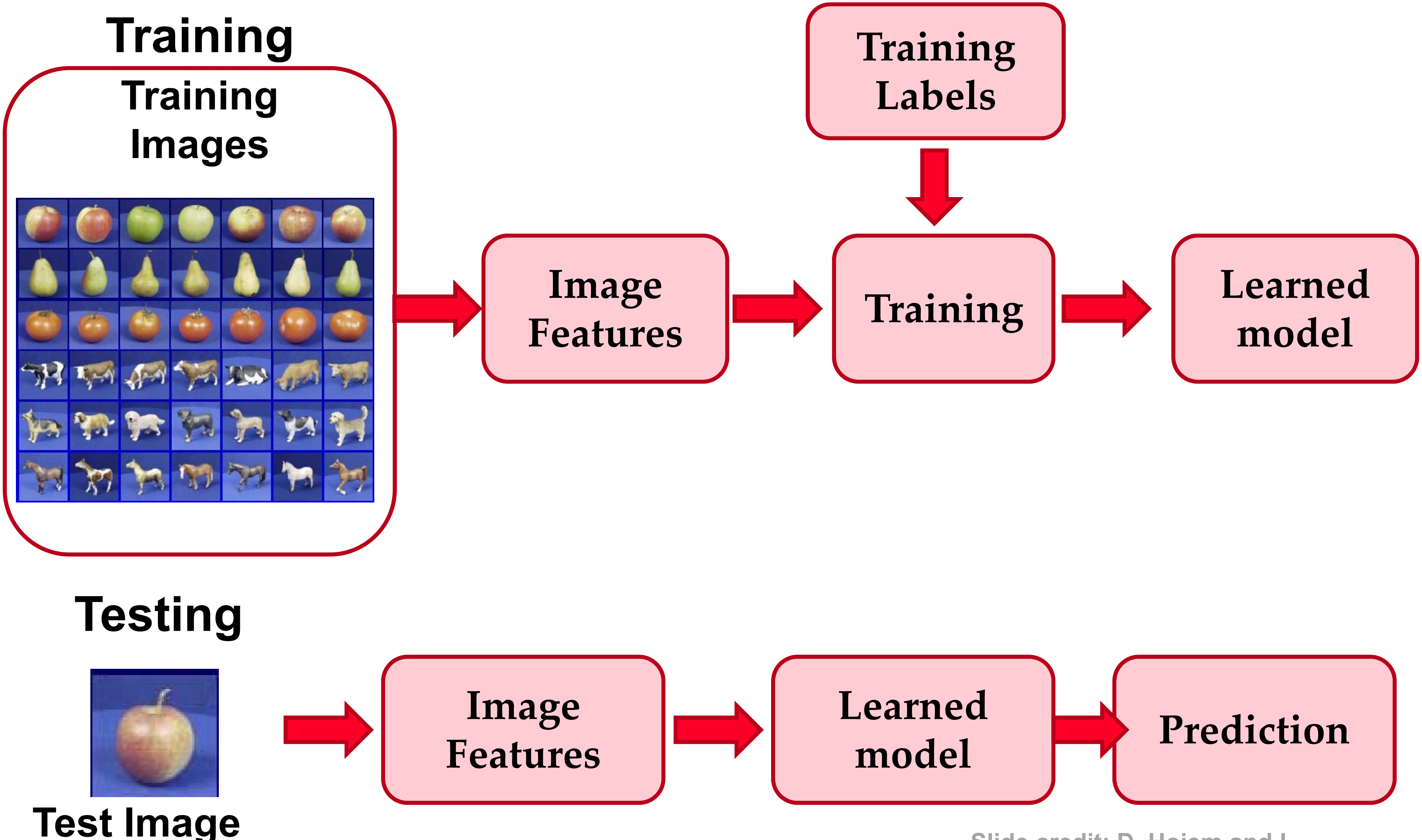


K-Means: The number of centroids was chosen based on an approximation method called the elbow graphing

Machine Learning Models

	Supervised Learning	*Unsupervised Learning*
Discrete	classification or categorization	clustering
Continuous	regression	dimensionality reduction

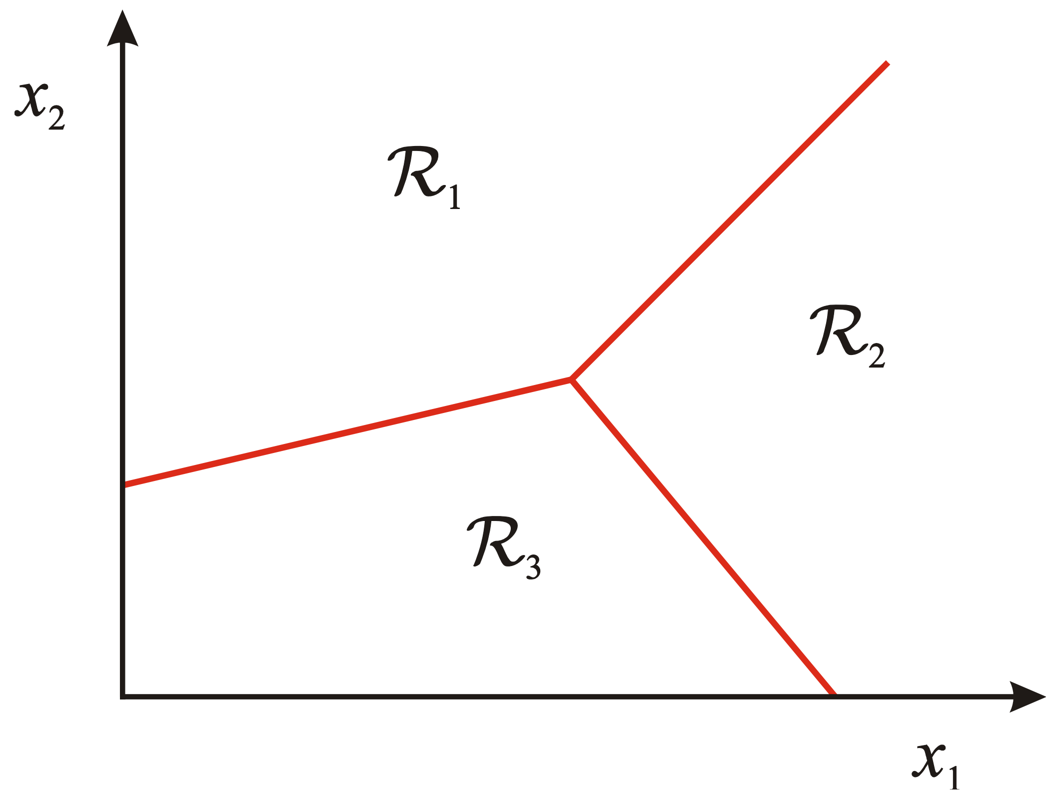
Steps



Slide credit: D. Hoiem and L. Lazebnik

Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into *decision regions* separated by *decision boundaries*



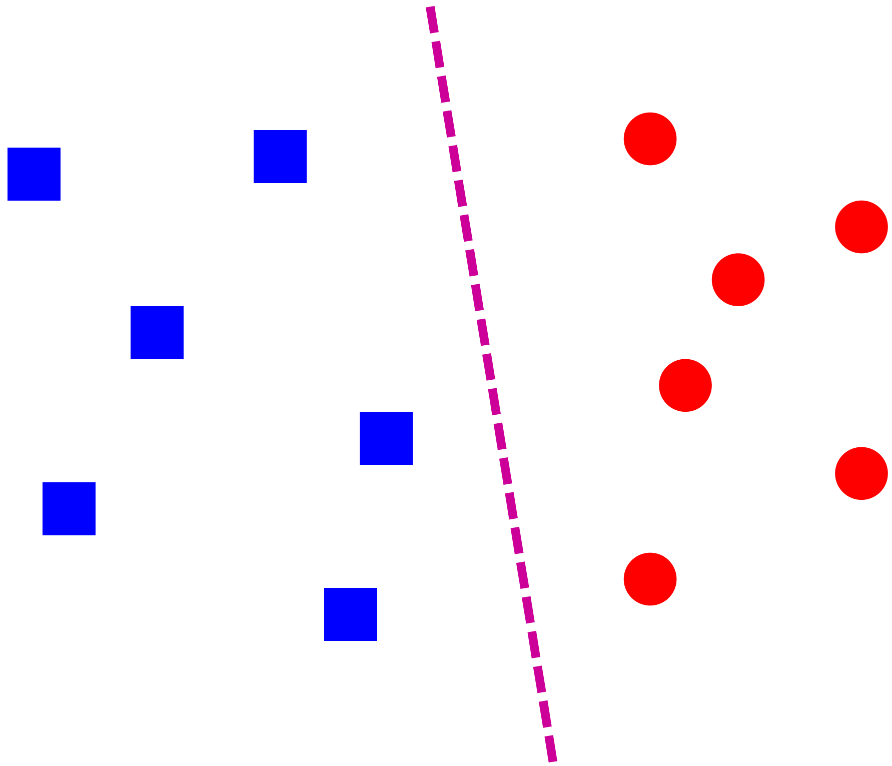
Slide credit: L. Lazebnik

Many classifiers to choose from

- SVM
- Neural networks
- Naïve Bayes
- Bayesian network
- Logistic regression
- Randomized Forests
- Boosted Decision Trees
- K-nearest neighbor
- RBMs
- Etc.

Slide credit: D. Hoiem

Classifiers: Linear



- Linear SVM: Find a *linear function* to separate the classes:

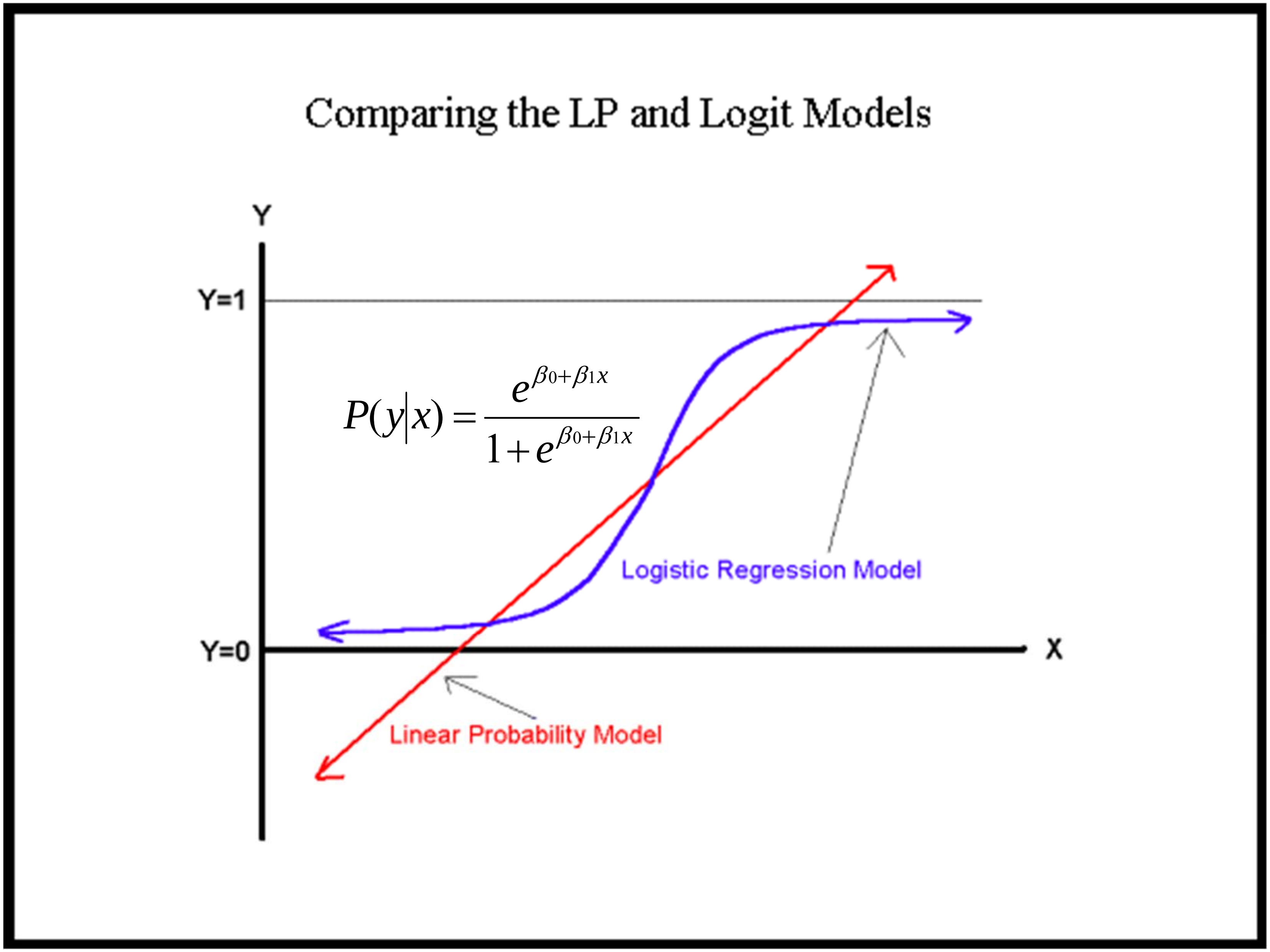
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w} \cdot \mathbf{x} + b)$$

Slide credit: L. Lazebnik

Logistic Regression: Nonlinear Decision

- In order for us to properly fit a curve for high-dimensional data we need algorithms that can create nonlinear boundaries
  - Higher dimensions need more non-linearities
- Logistic Regression is the most basic ML algorithm that can generate a nonlinear decision boundary

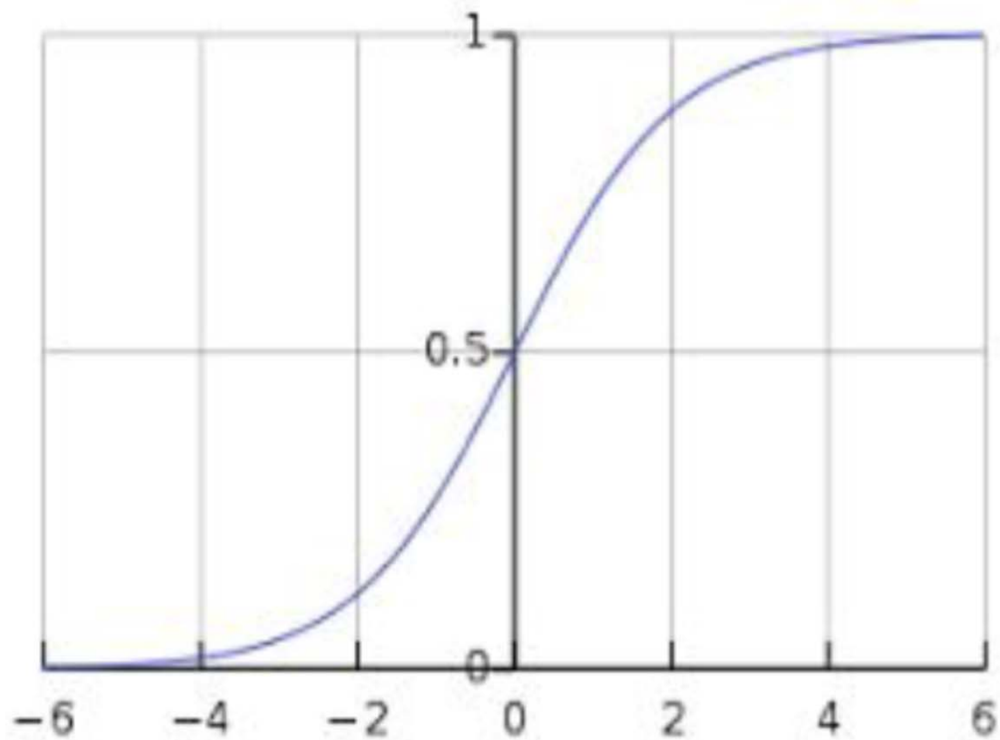
Linear and Logistic Regression



Inference – Sigmoid Function

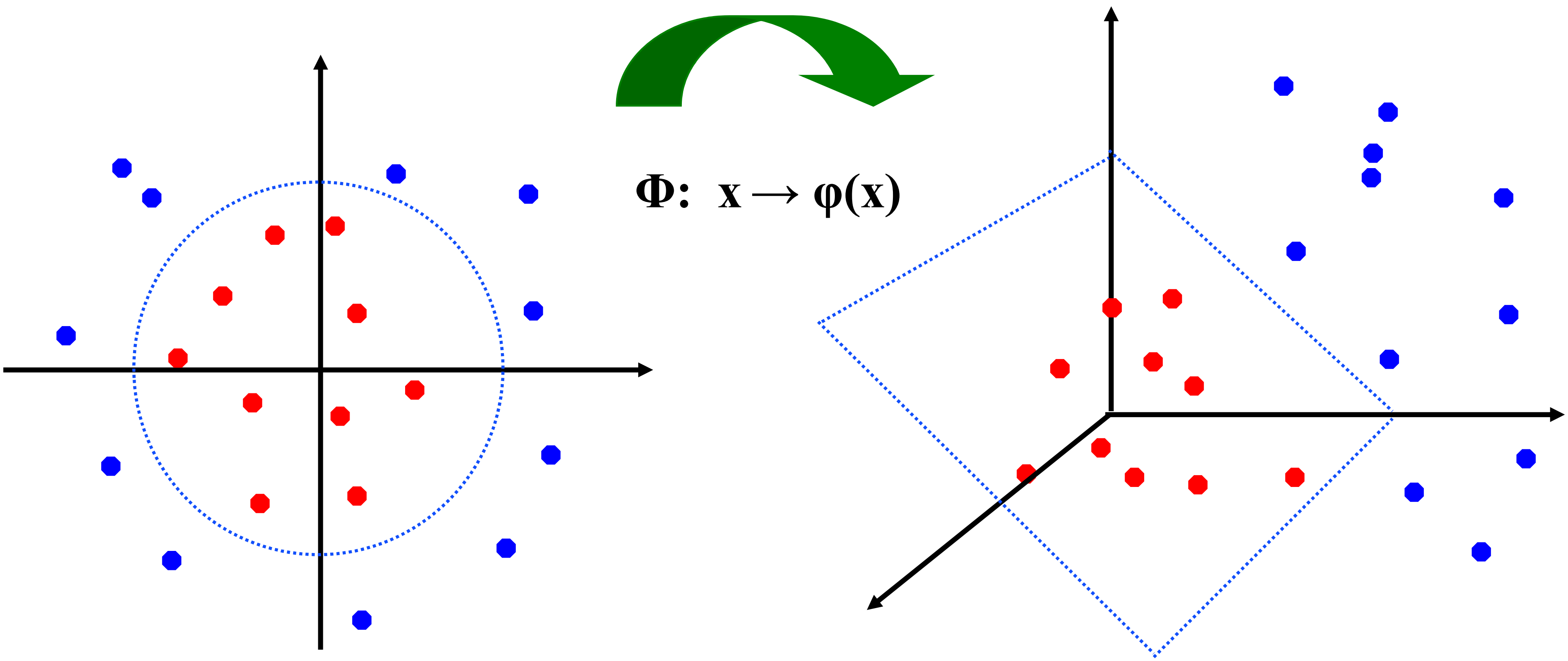
Equation: $\sigma(z) = \frac{1}{1+e^{-z}}$

- If z is a large positive number
    - Output will be close to 1
- If z is a large negative number
    - Output close to 0
- Only useful for 2 classes



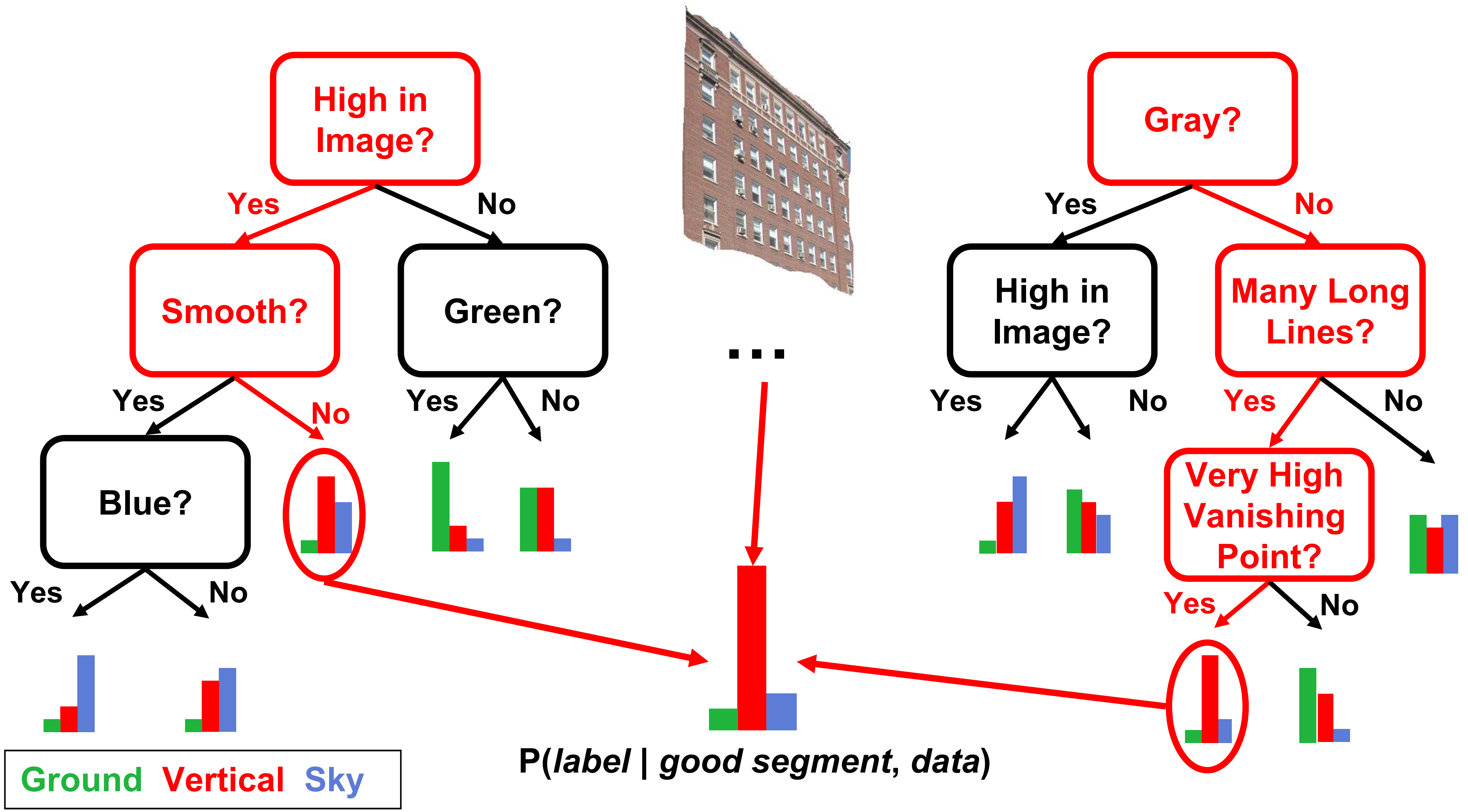
Nonlinear SVMs

- General idea: the original input space can always be mapped to some higher-dimensional feature space where the training set is separable:



Slide credit: Andrew Moore

Boosted Decision Trees



[Collins et al. 2002]

Supervised ML: Important Lessons

- Data Pre-Processing very important
  - Garbage in, garbage out
- Featurization
  - How do you represent your data?
  - How do you extract features from your signals
- Choice of Models depend on problems
  - Binary classification → Logistic/SVM may be sufficient
  - Multi-class classification → Tree algorithms?
- Validation: Cross-validation
  - K-Fold Validation: slice labeled data set into K sets, use K-1 for training, 1 for training
- Validation: heterogeneity & generalization
  - Training vs. test set
  - External validation – use a completely independent test set from different source

Performance Metrics: Depend on Domains!

- Sensitivity, recall, or true positive rate (TPR)

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

- Specificity, or true negative rate (TNR)

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP}$$

- Precision, or positive predictive value (PPV)

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

- False positive rate (FPR)

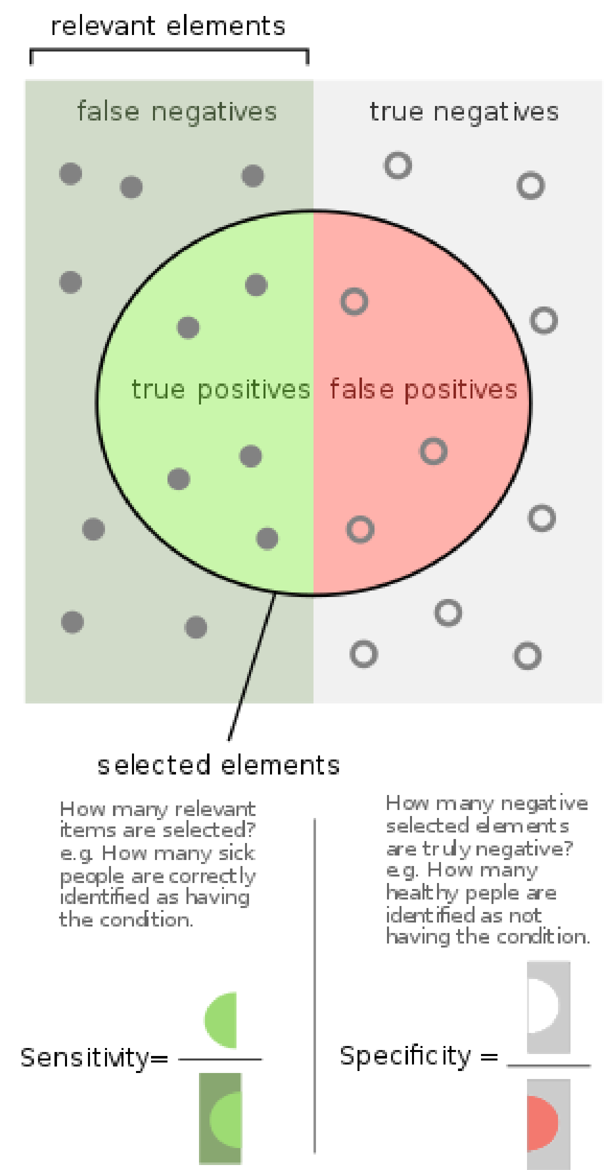
$$FPR = \frac{FN}{P} = \frac{FN}{FN + TP}$$

- Accuracy (ACC)

$$ACC = \frac{TP + TN}{P + N} = \frac{TP + TN}{TP + TN + FP + FN}$$

- F1 score: harmonic mean of precision & sensitivity

$$F_1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} \times \frac{2TP}{2TP + FP + FN}$$



Lab 2: PVA Detection

- Provide a numerical example of
  - How to featurizes continuous signals
  - How to extract meaningful features
  - How to select models for classifications

ML-Driven Ventilation Management

Detecting Patient-Ventilator Asynchrony (PVA) using Mechanical Ventilator Waveform Data

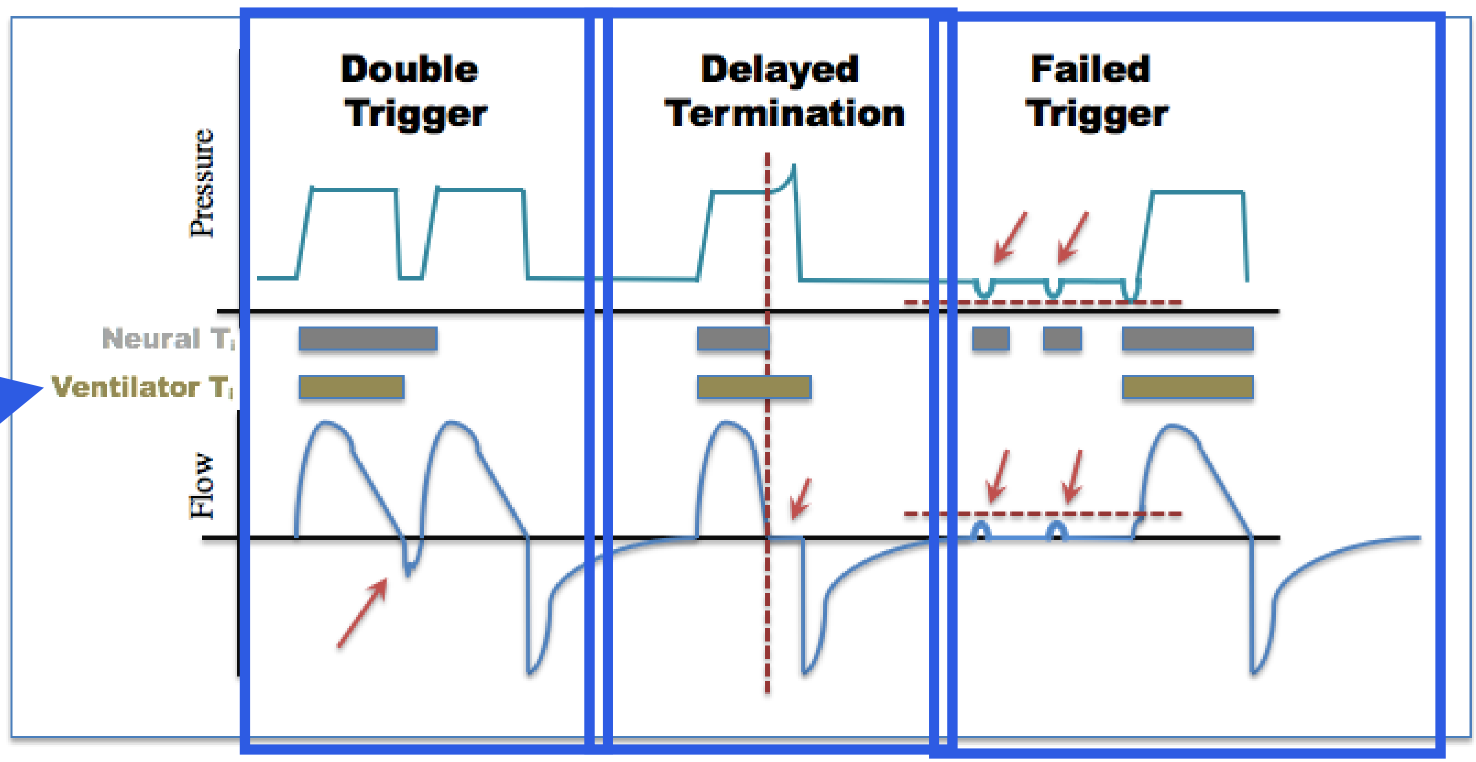
- Motivation:
  - Mechanical ventilation (MV) is a life-saving intervention but, if delivered inappropriately, can be harmful or even fatal
  - Patient-ventilator asynchrony (PVA) occurs when patient ventilatory demands are not matched by assistance from the mechanical ventilator
    ➔ How can we detect PVA and avoid respiratory failures without requiring 24/7 monitoring by a provider?

Slide credit: L. Lazebnik

Patient-Ventilator Asynchrony

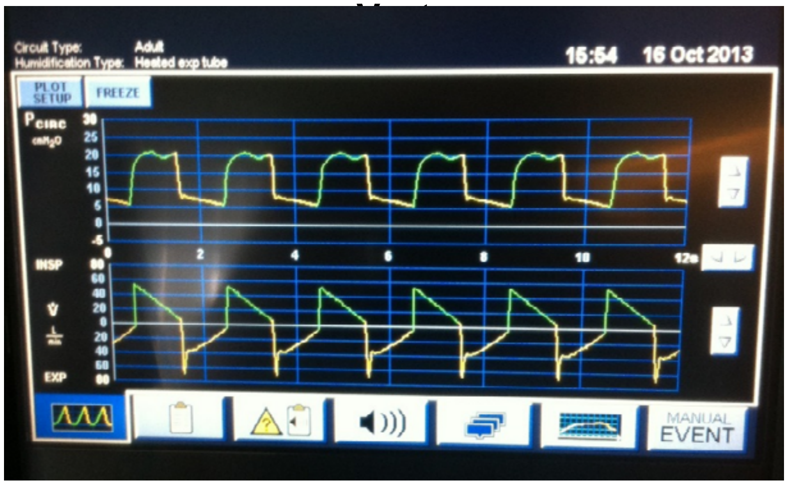
- Patient-ventilatory demands are not matched by assistance from mechanical ventilator

Patient Demand
≠
Ventilator Assistance



From ASCII to Asynchrony: Getting the Raw Data

Waveforms Encoded as ASCII Stream by the



Raspberry Pi 2 Model B

- **Hardwired to PB840**
- **Wireless data transmission to server**

- **900MHz quad-core CPU**
- **1GB RAM**
- **4 USB ports**
- **8 GB Micro SD card hard drive**

CSV File

```
2015-08-27 00:40:53.672, BE
2015-08-27 00:40:53.691, BS, S:51431,
2015-08-27 00:40:53.712, -0.93, 12.83
2015-08-27 00:40:53.732, 4.32, 12.89
2015-08-27 00:40:53.758, 10.65, 13.46
2015-08-27 00:40:53.779, 23.83, 14.86
2015-08-27 00:40:53.799, 42.40, 16.37
2015-08-27 00:40:53.819, 58.47, 18.78
2015-08-27 00:40:53.839, 71.28, 21.42
2015-08-27 00:40:53.859, 76.62, 24.62
2015-08-27 00:40:53.879, 74.93, 27.05
2015-08-27 00:40:53.899, 69.57, 28.52
2015-08-27 00:40:53.919, 59.86, 29.45
2015-08-27 00:40:53.938, 56.48, 30.13
2015-08-27 00:40:53.958, 49.09, 30.44
2015-08-27 00:40:53.978, 43.64, 30.65
2015-08-27 00:40:53.998, 39.35, 30.85
2015-08-27 00:40:54.018, 37.20, 30.95
2015-08-27 00:40:54.038, 31.56, 30.97
2015-08-27 00:40:54.058, 27.31, 31.16
2015-08-27 00:40:54.083, 23.08, 31.16
2015-08-27 00:40:54.103, 19.96, 31.07
2015-08-27 00:40:54.123, 16.75, 31.12
2015-08-27 00:40:54.143, 14.51, 31.03
2015-08-27 00:40:54.163, 10.45, 30.95
2015-08-27 00:40:54.183, 8.59, 30.94
2015-08-27 00:40:54.203, 6.70, 30.94
2015-08-27 00:40:54.223, 4.43, 30.87
2015-08-27 00:40:54.243, 2.73, 30.81
2015-08-27 00:40:54.263, 1.93, 30.80
2015-08-27 00:40:54.283, 1.60, 30.79
2015-08-27 00:40:54.303, 1.09, 30.73
2015-08-27 00:40:54.323, 1.66, 30.78
2015-08-27 00:40:54.342, 2.08, 30.81
2015-08-27 00:40:54.362, 2.33, 30.88
2015-08-27 00:40:54.382, 2.54, 30.85
2015-08-27 00:40:54.417, 3.32, 30.97
2015-08-27 00:40:54.447, 3.19, 30.94
2015-08-27 00:40:54.480, 3.44, 31.00
2015-08-27 00:40:54.509, 3.14, 31.06
2015-08-27 00:40:54.542, 3.18, 31.07
2015-08-27 00:40:54.576, 2.49, 31.10
2015-08-27 00:40:54.608, 2.06, 31.11
2015-08-27 00:40:54.639, 1.06, 31.13
2015-08-27 00:40:54.665, 0.15, 31.10
2015-08-27 00:40:54.700, -0.36, 30.96
2015-08-27 00:40:54.733, -0.29, 30.93
2015-08-27 00:40:54.766, -0.83, 30.91
```

From ASCII to Asynchrony: Transforming Raw Data into Information

```
2015-08-27 00:40:53.672, BE
2015-08-27 00:40:53.691, BS, S:51431,
2015-08-27 00:40:53.712, -0.93, 12.83
2015-08-27 00:40:53.732, 4.32, 12.89
2015-08-27 00:40:53.758, 10.65, 13.46
2015-08-27 00:40:53.779, 23.83, 14.86
2015-08-27 00:40:53.799, 42.40, 16.37
2015-08-27 00:40:53.819, 58.47, 18.78
2015-08-27 00:40:53.839, 71.28, 21.42
2015-08-27 00:40:53.859, 76.62, 24.62
2015-08-27 00:40:53.879, 74.93, 27.05
2015-08-27 00:40:53.899, 69.57, 28.52
2015-08-27 00:40:53.919, 59.86, 29.45
2015-08-27 00:40:53.938, 56.48, 30.13
2015-08-27 00:40:53.958, 49.09, 30.44
2015-08-27 00:40:53.978, 43.64, 30.65
2015-08-27 00:40:53.998, 39.35, 30.85
2015-08-27 00:40:54.018, 37.20, 30.95
2015-08-27 00:40:54.038, 31.56, 30.97
2015-08-27 00:40:54.058, 27.31, 31.16
2015-08-27 00:40:54.083, 23.08, 31.16
2015-08-27 00:40:54.103, 19.96, 31.07
2015-08-27 00:40:54.123, 16.75, 31.12
2015-08-27 00:40:54.143, 14.51, 31.03
2015-08-27 00:40:54.163, 10.45, 30.95
2015-08-27 00:40:54.183, 8.59, 30.94
2015-08-27 00:40:54.203, 6.70, 30.94
2015-08-27 00:40:54.223, 4.43, 30.87
2015-08-27 00:40:54.243, 2.73, 30.81
2015-08-27 00:40:54.263, 1.93, 30.80
2015-08-27 00:40:54.283, 1.60, 30.79
2015-08-27 00:40:54.303, 1.09, 30.73
2015-08-27 00:40:54.323, 1.66, 30.78
2015-08-27 00:40:54.342, 2.08, 30.81
2015-08-27 00:40:54.362, 2.33, 30.88
2015-08-27 00:40:54.382, 2.54, 30.85
2015-08-27 00:40:54.417, 3.32, 30.97
2015-08-27 00:40:54.447, 3.19, 30.94
2015-08-27 00:40:54.480, 3.44, 31.00
2015-08-27 00:40:54.509, 3.14, 31.06
2015-08-27 00:40:54.542, 3.18, 31.07
2015-08-27 00:40:54.576, 2.49, 31.10
2015-08-27 00:40:54.608, 2.06, 31.11
2015-08-27 00:40:54.639, 1.06, 31.13
2015-08-27 00:40:54.665, 0.15, 31.10
2015-08-27 00:40:54.700, -0.36, 30.96
2015-08-27 00:40:54.733, -0.29, 30.93
2015-08-27 00:40:54.766, -0.83, 30.91
2015-08-27 00:40:54.799, -62.05, 30.89
2015-08-27 00:40:54.818, -82.91, 27.97
2015-08-27 00:40:54.834, -84.59, 22.29
2015-08-27 00:40:54.850, -66.67, 17.82
2015-08-27 00:40:54.866, -54.67, 16.00
2015-08-27 00:40:54.882, -46.36, 15.48
```

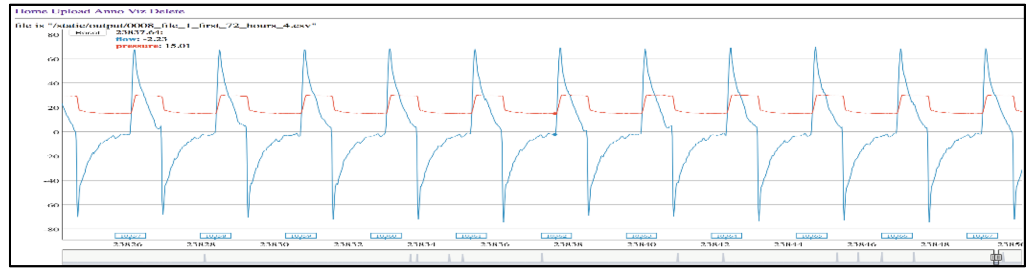
Line 1 Col 1 (none) Unicode (UTF-8) Unix (LF)

Clinical Rules → Python Rules Engine

```python
cross0_time = []
for i in range(len(waveform) - 2): #if change to append s
    if waveform[i] >= 0:
        if waveform[i + 1] <= -5 and waveform[i + 2] < 0:
                cross0_time.append(t[i + 1])
        elif waveform[i + 1]<0 and waveform[i + 4] <= -5:
                cross0_time.append(t[i + 1])
        elif waveform[i+1]<0 and waveform[i+2]<=-5:
                cross0_time.append(t[i + 1])
        elif waveform[i+1]<0 and waveform[i+2]<0 and wave
                cross0_time.append(t[i + 1])
```

Breath Type-Specific Algorithms

Waveform Visualization Application (dygraphs-based)



Breath Metadata Output

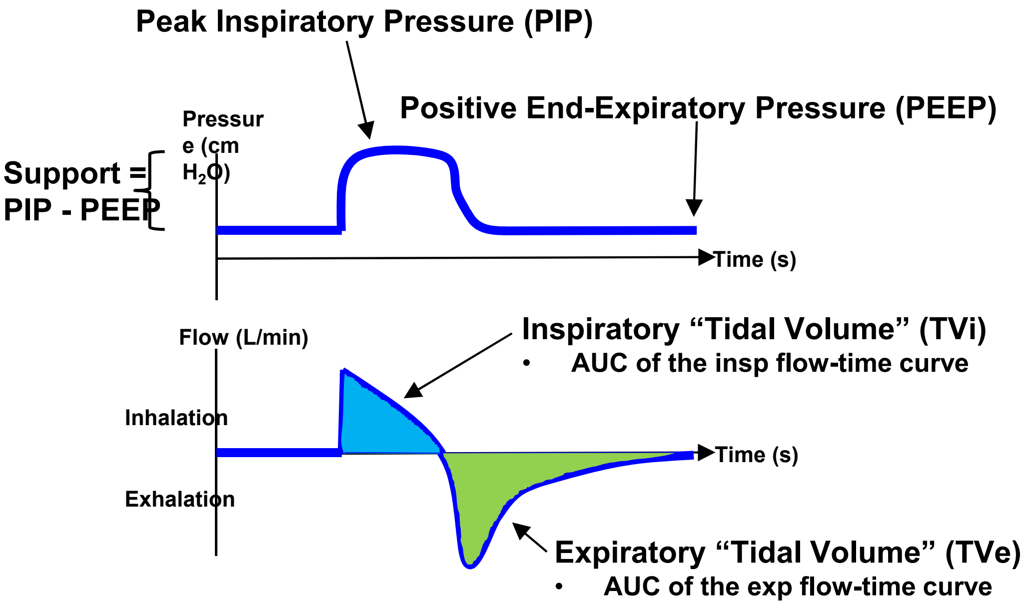
1	BN	ventBN	BS	IEnd	I:E ratio	inst_RR	TVi	TVe	TVe:TVi ratio	x02	TVi2	TVe2	maxF	PIP	PEEP
2	1	3738	0.02	0.92	0.49	22	475	483	1.02	0.92	475	483	62	33	11
3	2	3739	2.76	3.66	0.49	22	488	489	1.00	3.66	488	489	64	33	11
4	3	3740	5.5	6.4	0.49	22	477	485	1.02	6.4	477	485	64	33	11
5	4	3741	8.24	9.14	0.49	22	474	477	1.01	9.14	474	477	60	33	11
6	5	3742	10.98	11.88	0.49	22	484	486	1.00	11.88	484	486	64	33	11
7	6	3743	13.72	14.62	0.49	22	472	480	1.02	14.62	472	480	60	33	11
8	7	3744	16.46	17.36	0.49	22	466	471	1.01	17.36	466	471	62	33	11
9	8	3745	19.2	20.1	0.49	22	461	467	1.01	20.1	461	467	60	33	11
10	9	3746	21.94	22.84	0.49	22	456	457	1.00	22.84	456	457	61	33	11
11	10	3747	24.68	25.58	0.49	22	470	480	1.02	25.58	470	480	62	33	11

Breath Type Classification Output

1	BN	ventBN	BS	TVV	double_trig	breath_stack	delayed_term	cough	suction	vent_disconnect	flow_asynch
5001	5000	8737	12138	0	0	0	0	0	0	0	0
5002	5001	8738	12139.1	0	0	0	0	0	0	0	0
5003	5002	8739	12141.8	**50**	**1**	0	0	0	0	0	0
5004	5003	8740	12142.9	0	0	0	**1**	0	0	0	0
5005	5004	8741	12145.7	50	0	0	0	0	0	0	0
5006	5005	8742	12148.4	0	0	0	**1**	0	0	0	0
5007	5006	8743	12150.2	**51**	0	**1**	0	0	0	0	0
5008	5007	8744	12153	0	**1**	0	0	0	0	0	0
5009	5008	8745	12154	0	0	0	**1**	0	0	0	0
5010	5009	8746	12156.8	0	**1**	0	0	0	0	0	0

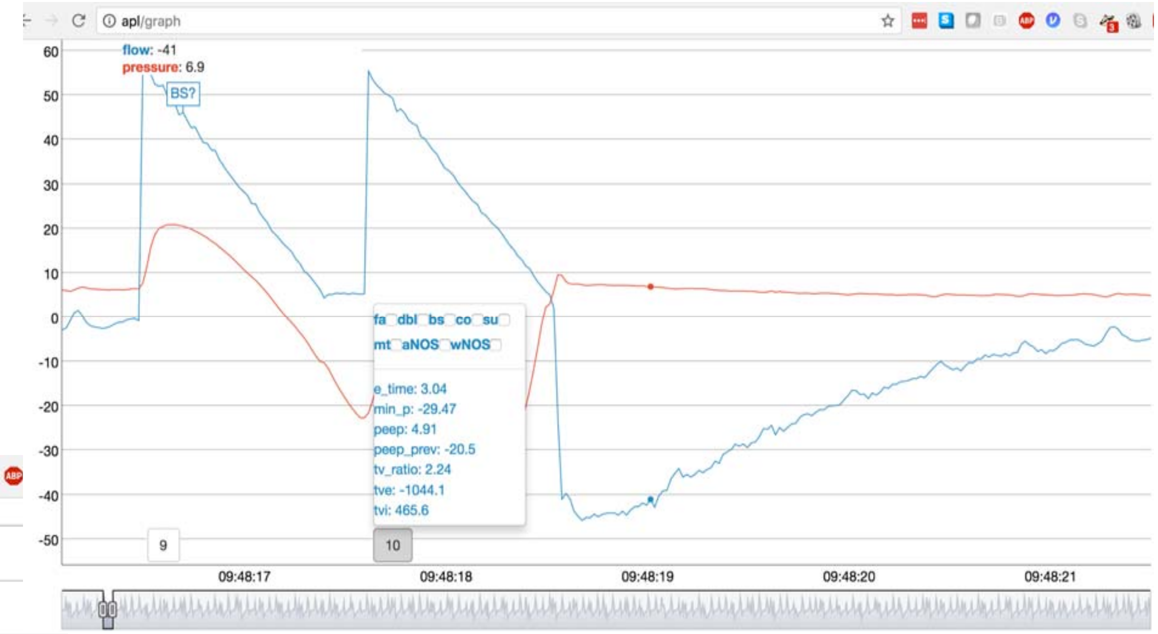
Anatomy of a Normal Ventilator-Assisted Breath

Normal, Synchronous Breath



AIM 1: What are we Trying to Detect?

- ❑ Double Trigger (DBLA)



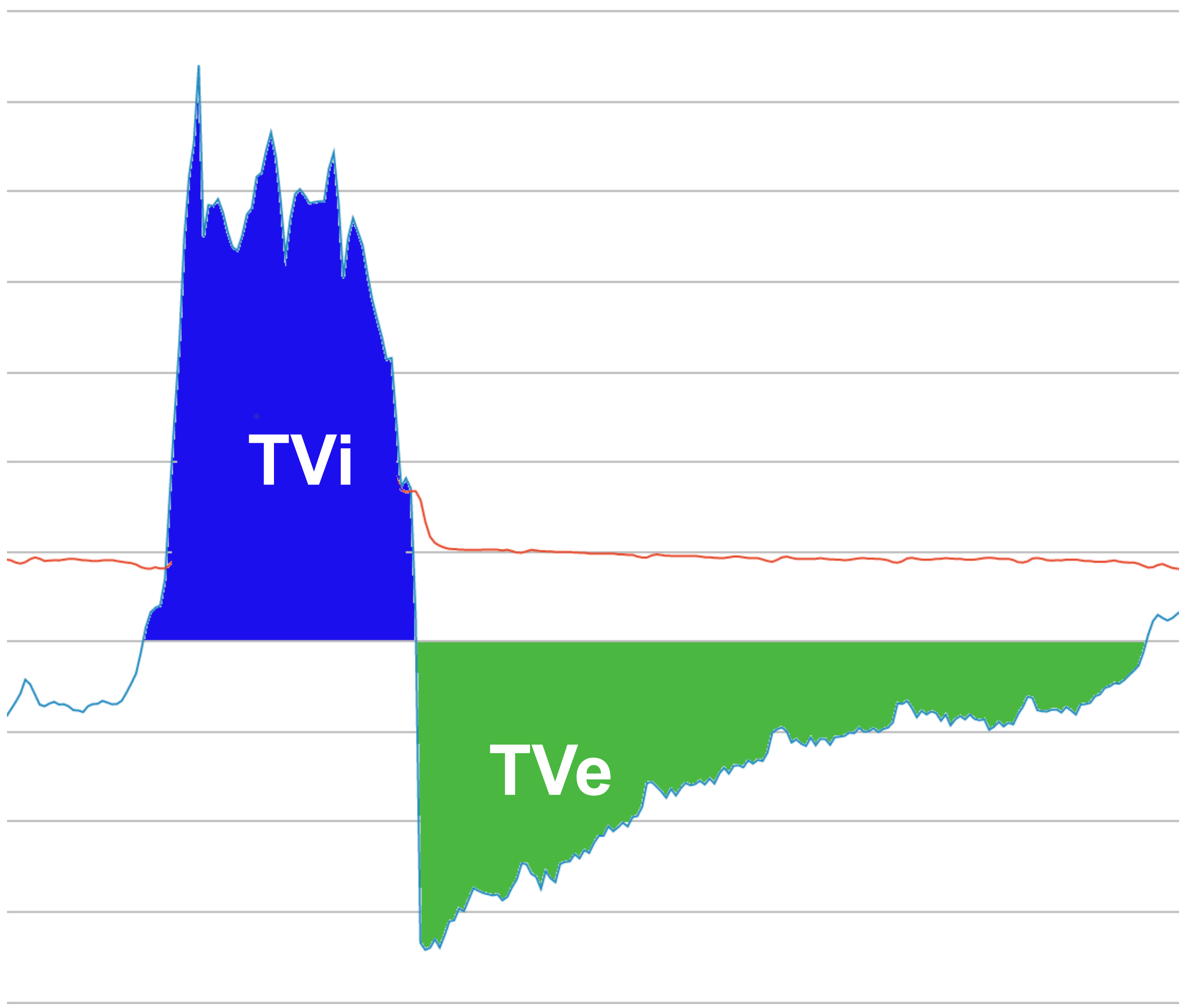
- ❑ Breath Stacking (BSA)

Features

- Metadata
  - Clinically relevant statistics derived from raw ventilator data (pressure/flow)
  - **64 base features derived the current and previous breaths**

- Examples
  - TVi: Total volume of inhaled air
  - Tve: Total volume of exhaled air
  - iTime: Amount of time a patient used to inhale
  - eTime: Amount of time a patient used to exhale

TVi and TVe



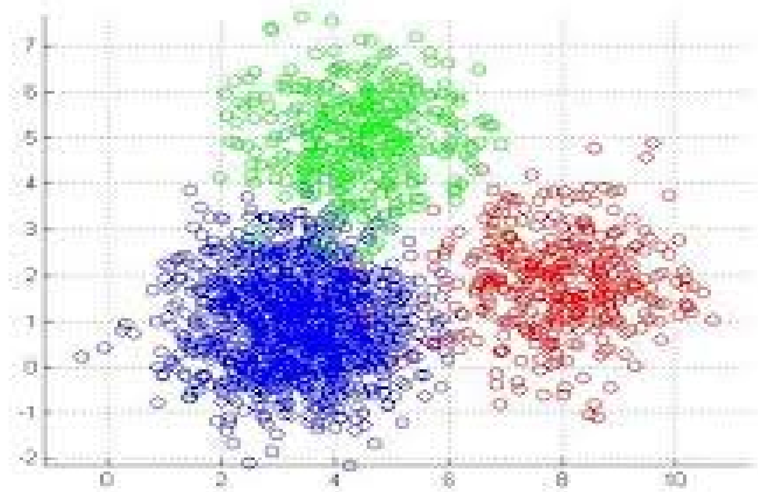
Dataset

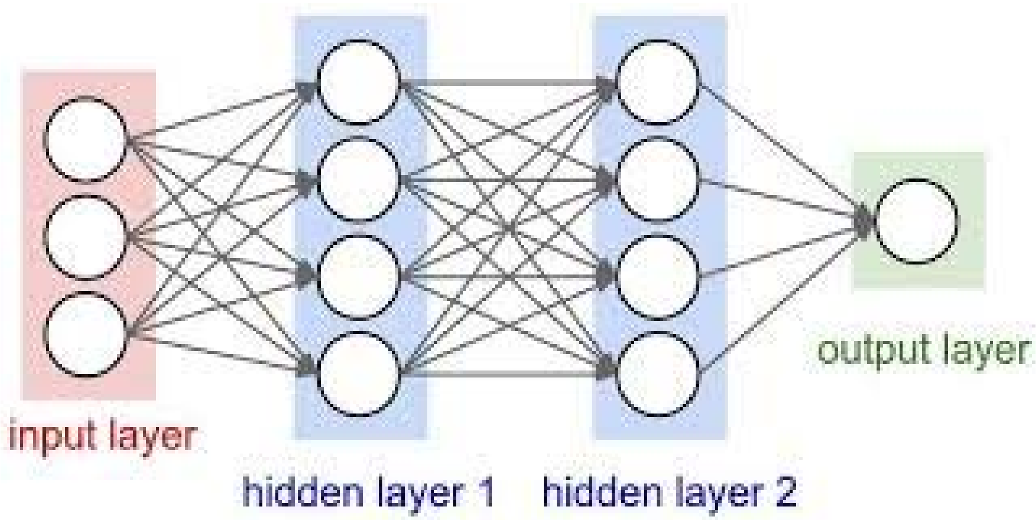
- 300-350 breaths for ~20 patients
- Two clinicians manually annotate each breath

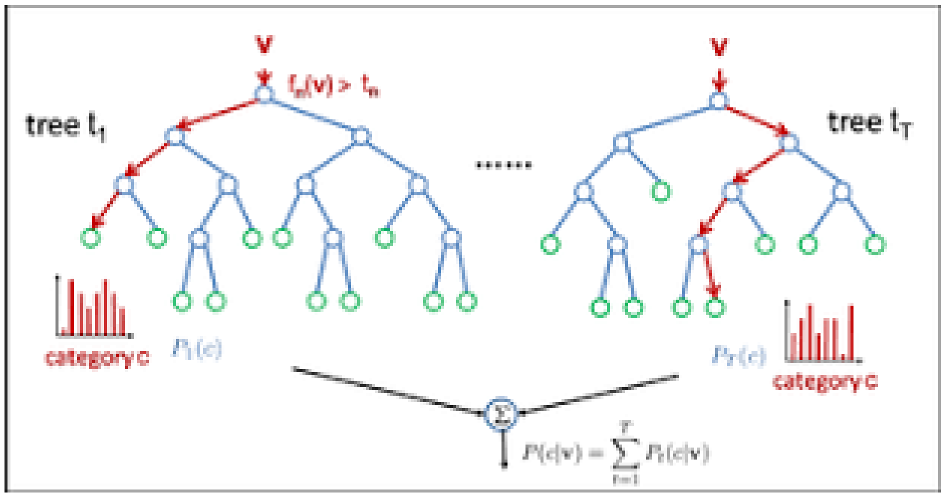
Breath Type	Number of Breaths	Percentage of Dataset
Normal	6548	67.37
Cough	123	1.27
Suction	368	3.79
Non-PVA (normal + suction + cough)	7039	72.45
Double trigger (DTA)	752	7.74
Breath stacking (BSA)	1928	19.83

Machine Learning Models (1)

- K-Mean Clustering
- Multilayer Perceptron (MLP)
- Gradient Boosted Classifier (GBC)
- Random Forest (RF)
- Extremely Random Tree Classifier (ERTC)
- Ensemble of Multiple Models

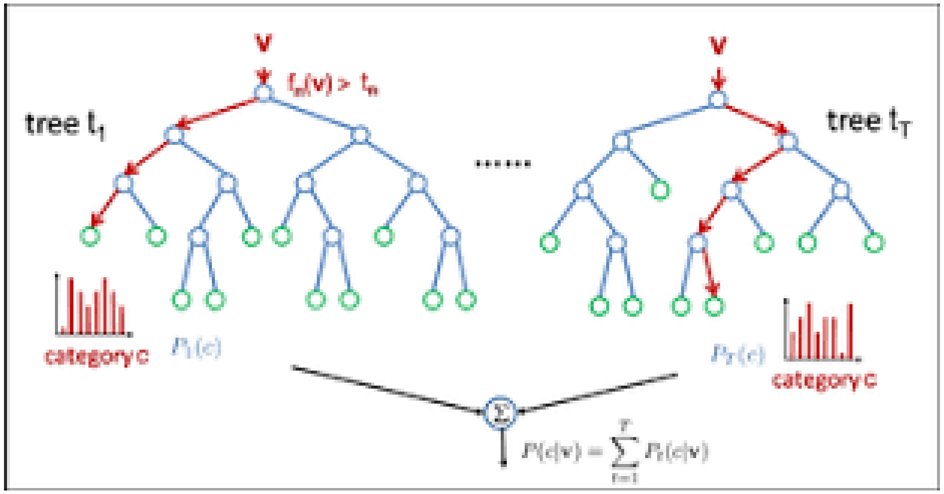


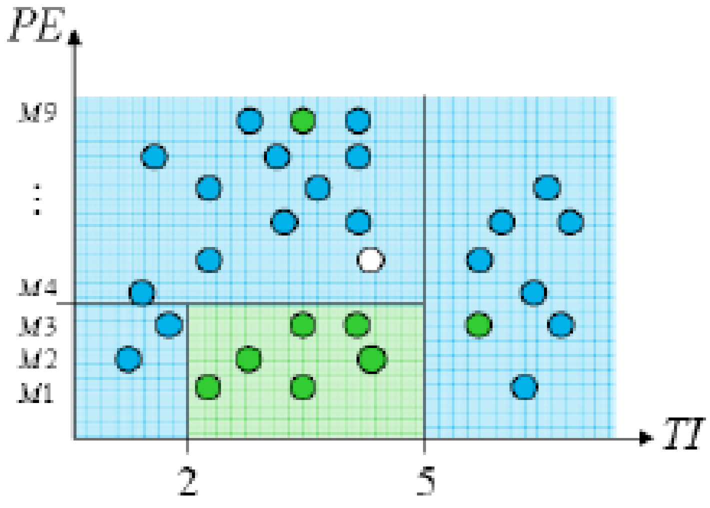


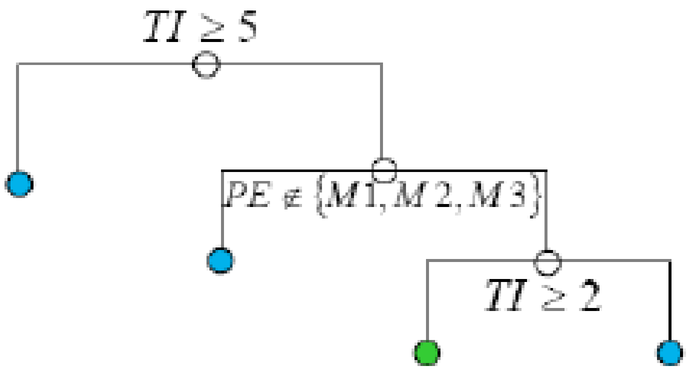


Machine Learning Models (2)

- Random Forest (RF)
  - Uses the classification and regression tree (CART) algorithm to perform tree splitting and the cross-entropy criteria to minimize the impurity function

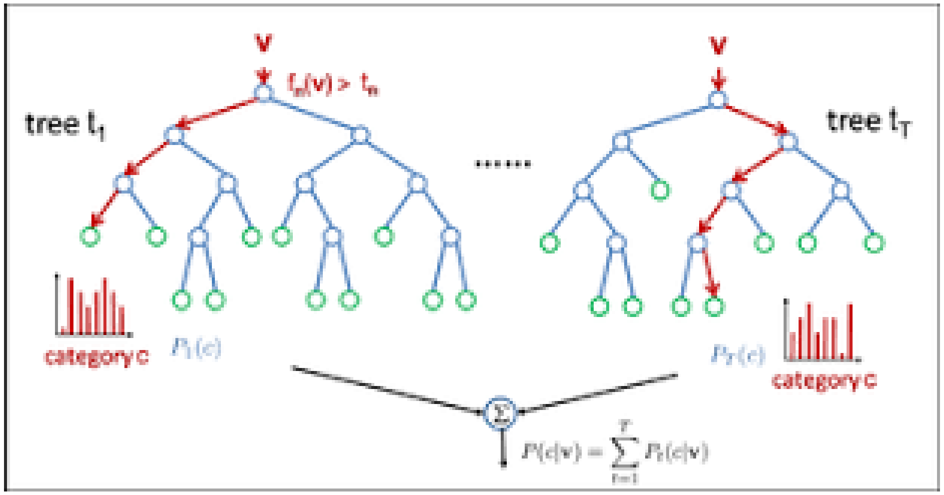






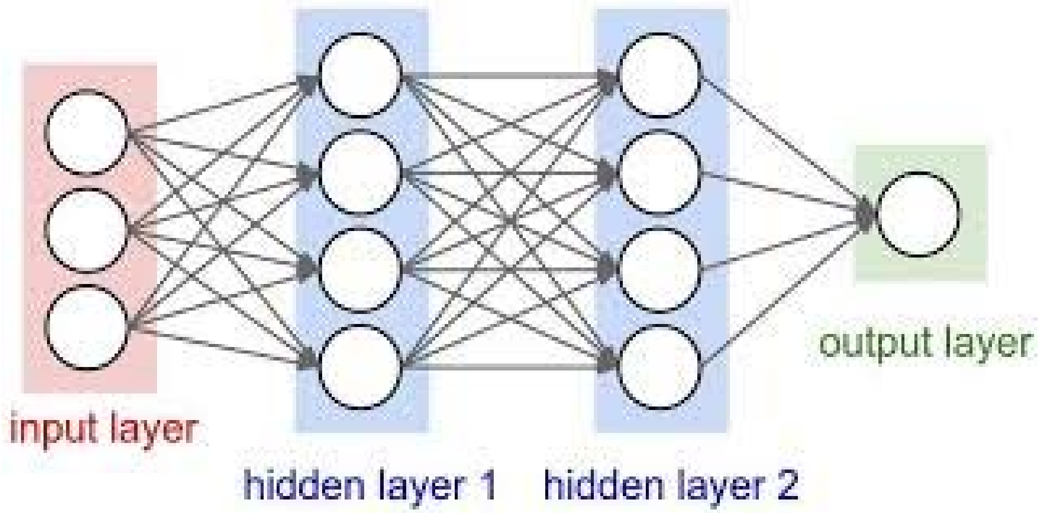
Machine Learning Models (3)

- Random Forest (RF)
  - Uses the classification and regression tree (CART) algorithm to perform tree splitting and the cross-entropy criteria to minimize the impurity function
- Extremely Random Tree Classifier (ERTC)
  - Tree splits in ERTC are performed randomly
- Gradient Boosted Classifier (GBC)
  - Use deviance for its loss function



Machine Learning Models (4)

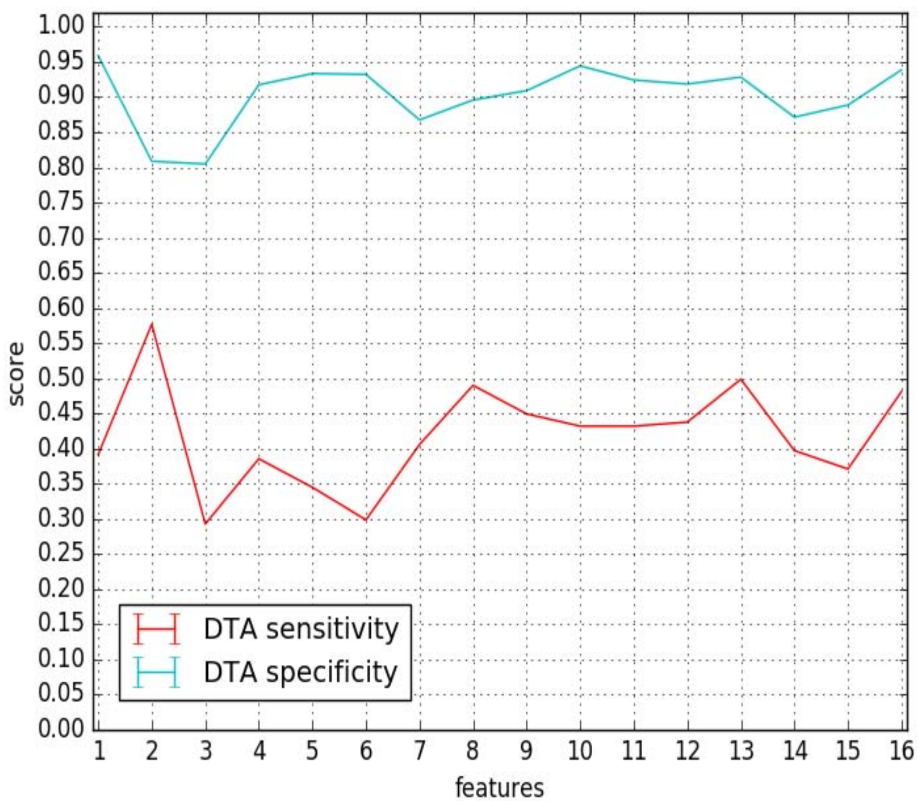
- Multilayer Perceptron (MLP)
  - A class of feedforward artificial neural network
  - Consists of 3 layers; Except for input nodes, each node is a neuron that uses a nonlinear activation function
  - Uses backpropagation with the *tanh* activation function and the cross-entropy loss function

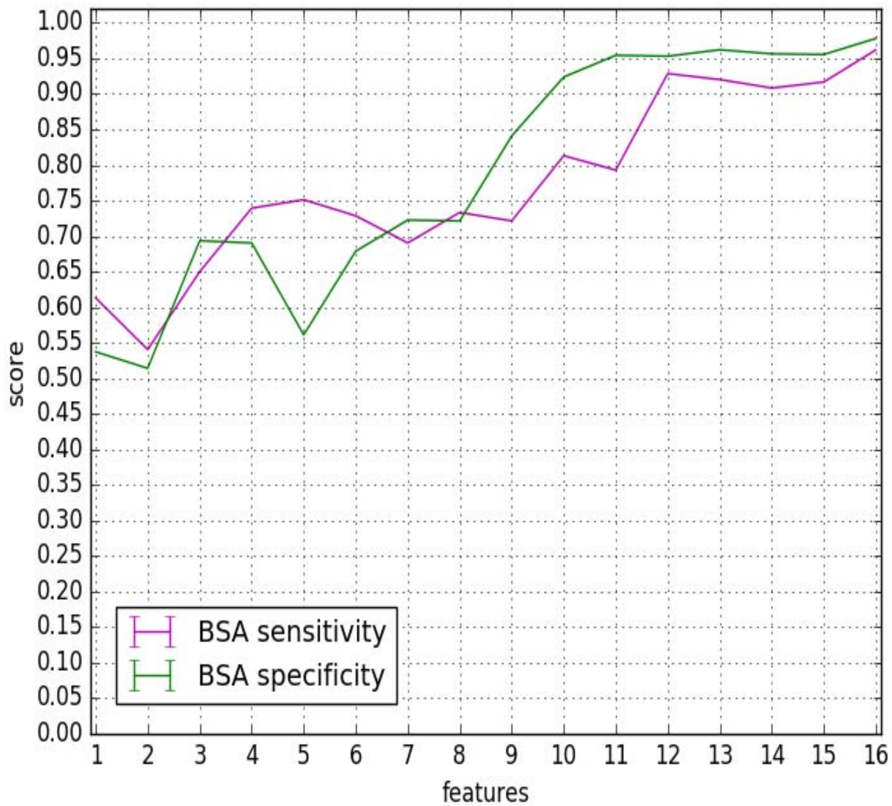


- Ensemble of Multiple Models
  - Weighted combination of multiple classifiers

Binary Classification Results

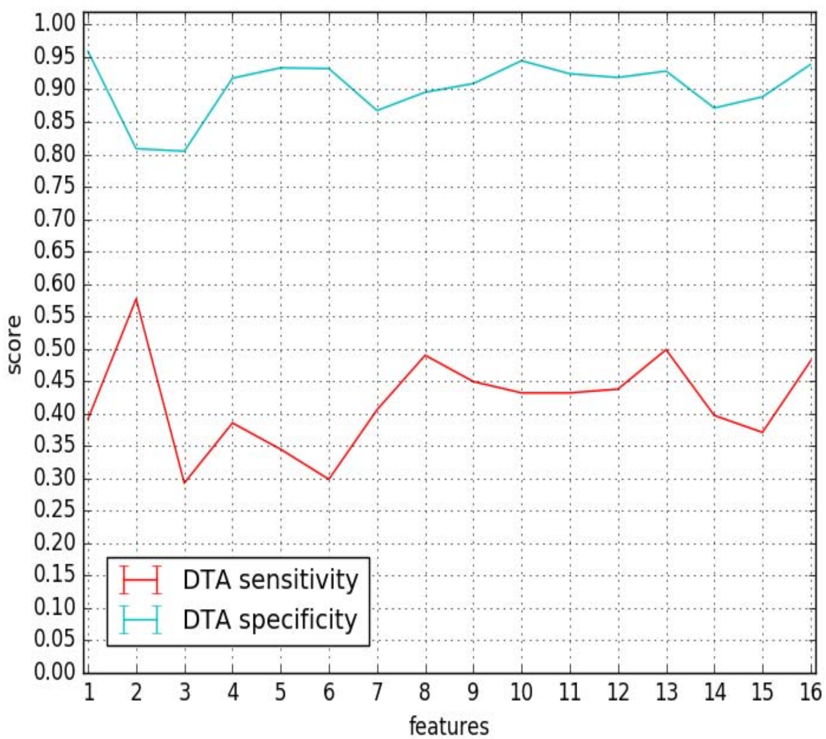
- ❑ Detect only Double Trigger or Breath Stacking, but not at the same time.
- ❑ Feature Engineering: Chi-Square Sensitivity Analysis
  - Sensitivity = # of true positive / (true positive + false negative)
  - Specificity = # of true negative / (true negative + false positive)

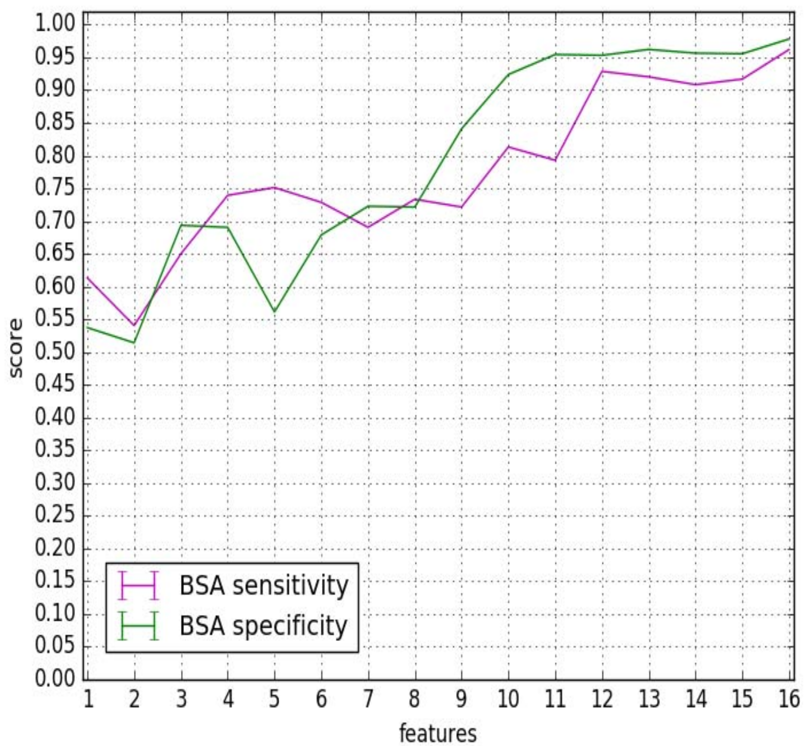




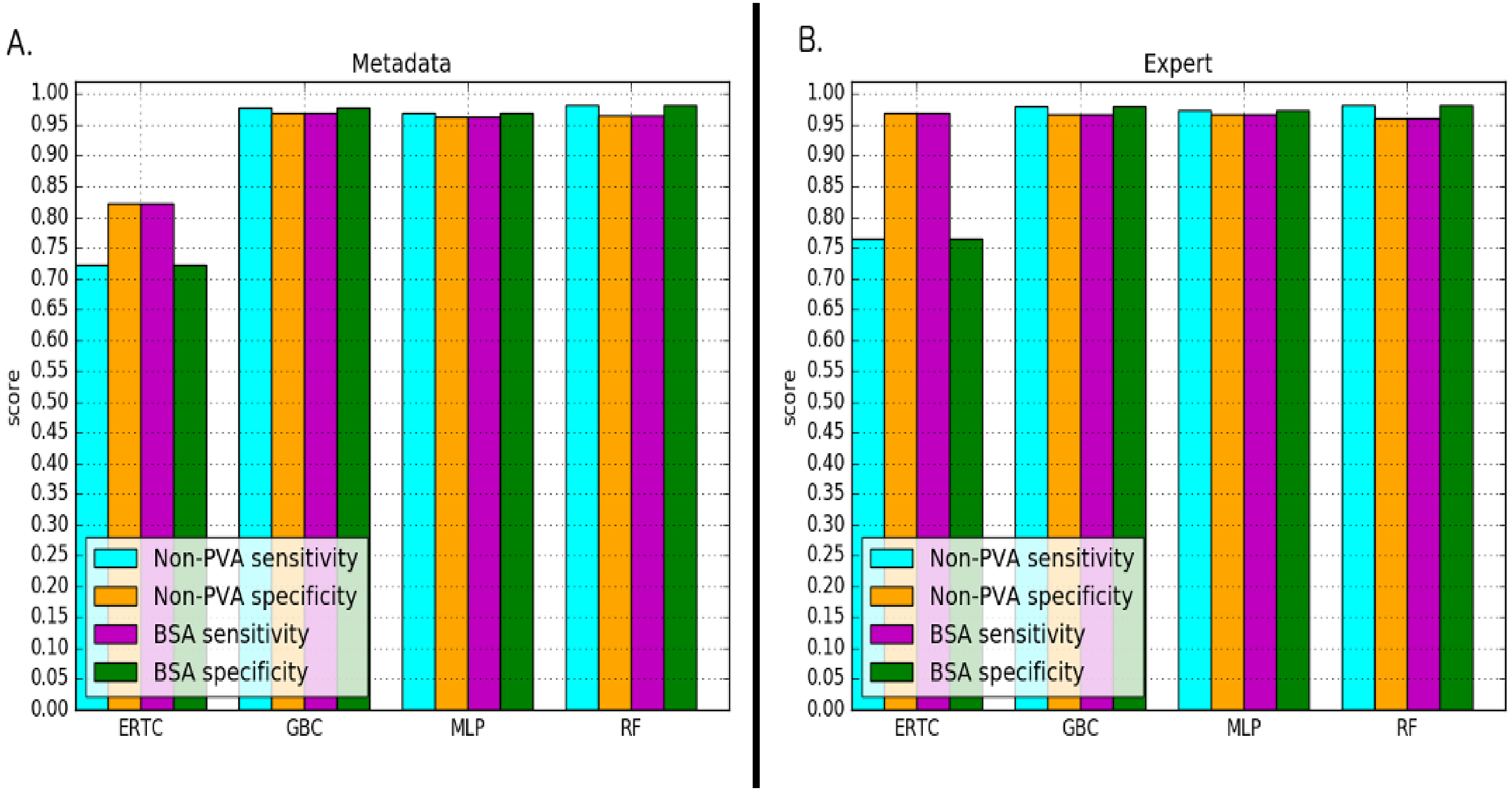
Feature Selection

- Feature Engineering: Chi-Square Sensitivity Analysis
  - 32 out of 64 features chosen
  - 7 optimal for BSA, 11 optimal for DT
- Expert (doctors) selection: 3 features
  - TVe:TVi ratio
  - TVe:TVi ratio – previous breath
  - eTime – previous breath

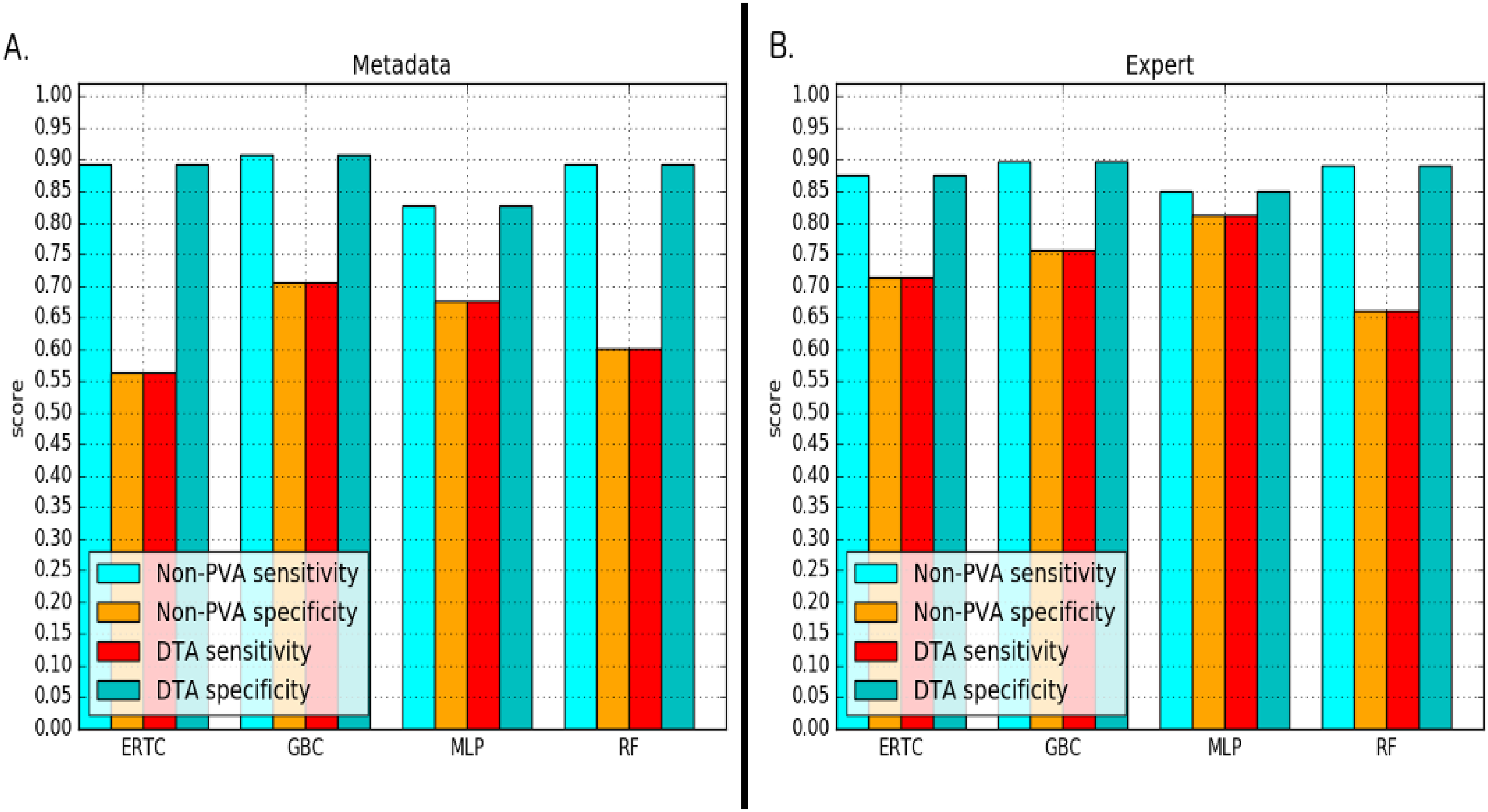




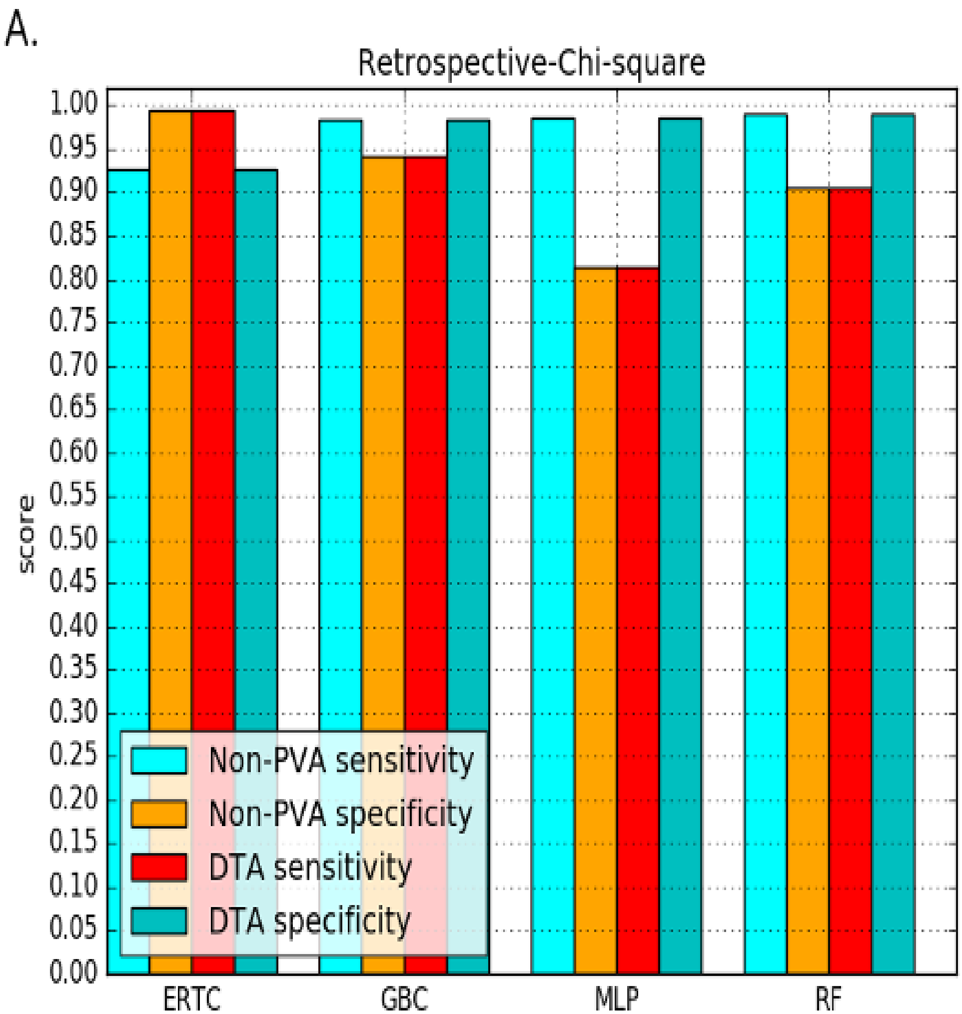
Breath Stacking (BS) Classification Results

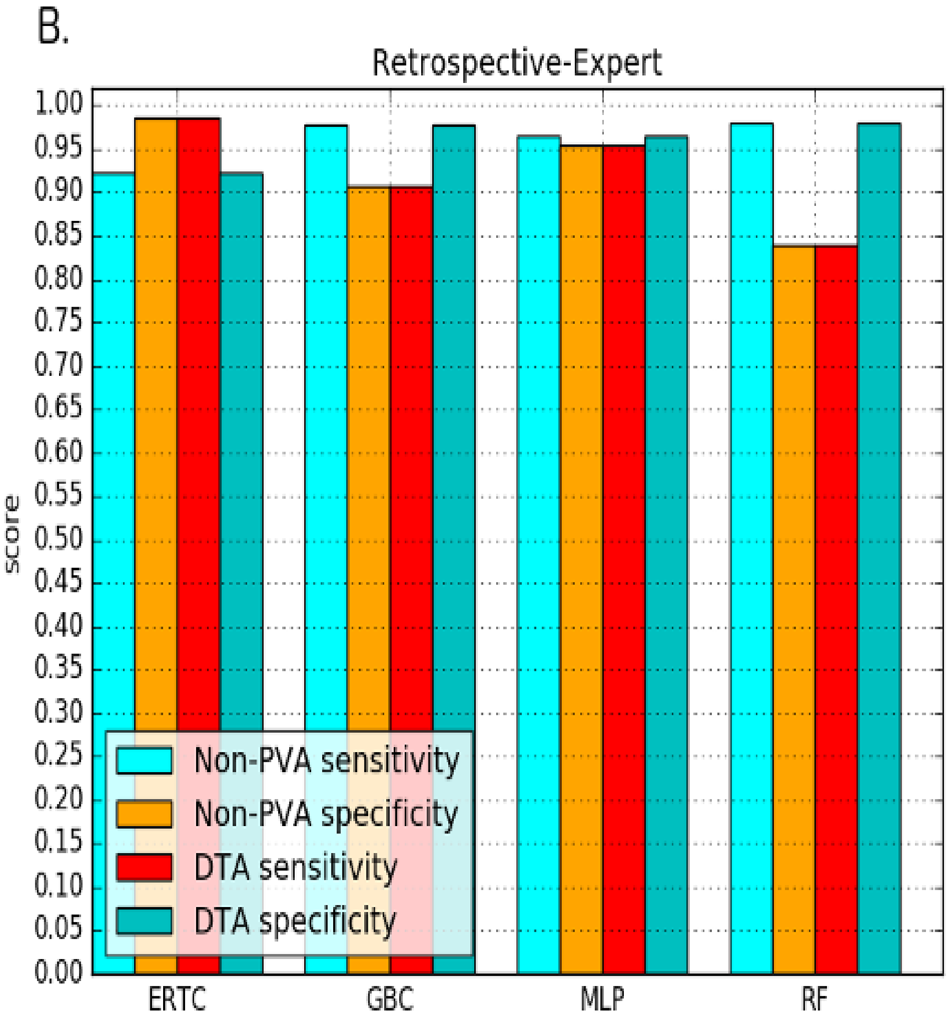


Double Trigger (DT) Classification Results



Double Trigger (DT) w/ Time Varying Features

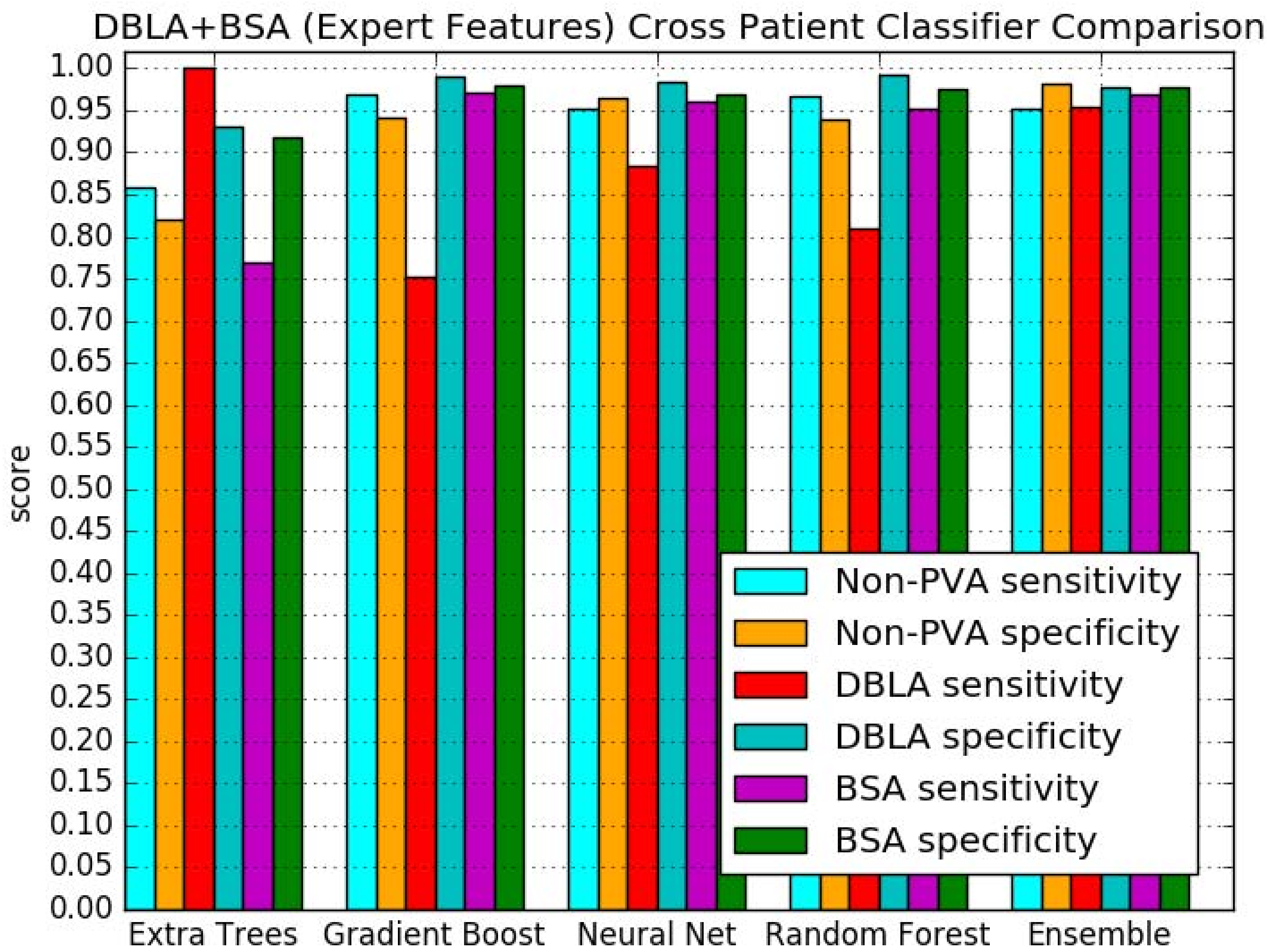




Multi-Class Classification

- Analyze data for both DTA and BSA at the same time
- Add features chosen by experts to those chosen by statistical methods
- Class imbalance issue
  - Apply SMOTE (Synthetic Minority Over-sampling Technique) was used to improve results for DTA specifically

Detection Results



- ❑ Ensemble models superior for classification when data is noisy or has high complex decision boundaries

Detection Results

Algorithm	Class	Accuracy	Sensitivity	Specificity
Ensemble	Non-PVA	0.971	0.9673	0.9806
	DTA	0.9742	0.9601	0.9754
	BSA	0.9793	0.9445	0.9879
ERTC	Non-PVA	0.7245	0.6744	0.856
	DTA	0.8693	0.9934	0.8589
	BSA	0.7683	0.5835	0.814
GBC	Non-PVA	0.9707	0.9692	0.9746
	DTA	0.9745	0.9335	0.9779
	BSA	0.9779	0.9445	0.9861
Neural Net	Non-PVA	0.954	0.9439	0.9806
	DTA	0.9576	0.9628	0.9572
	BSA	0.9678	0.9155	0.9807