Classification Analysis

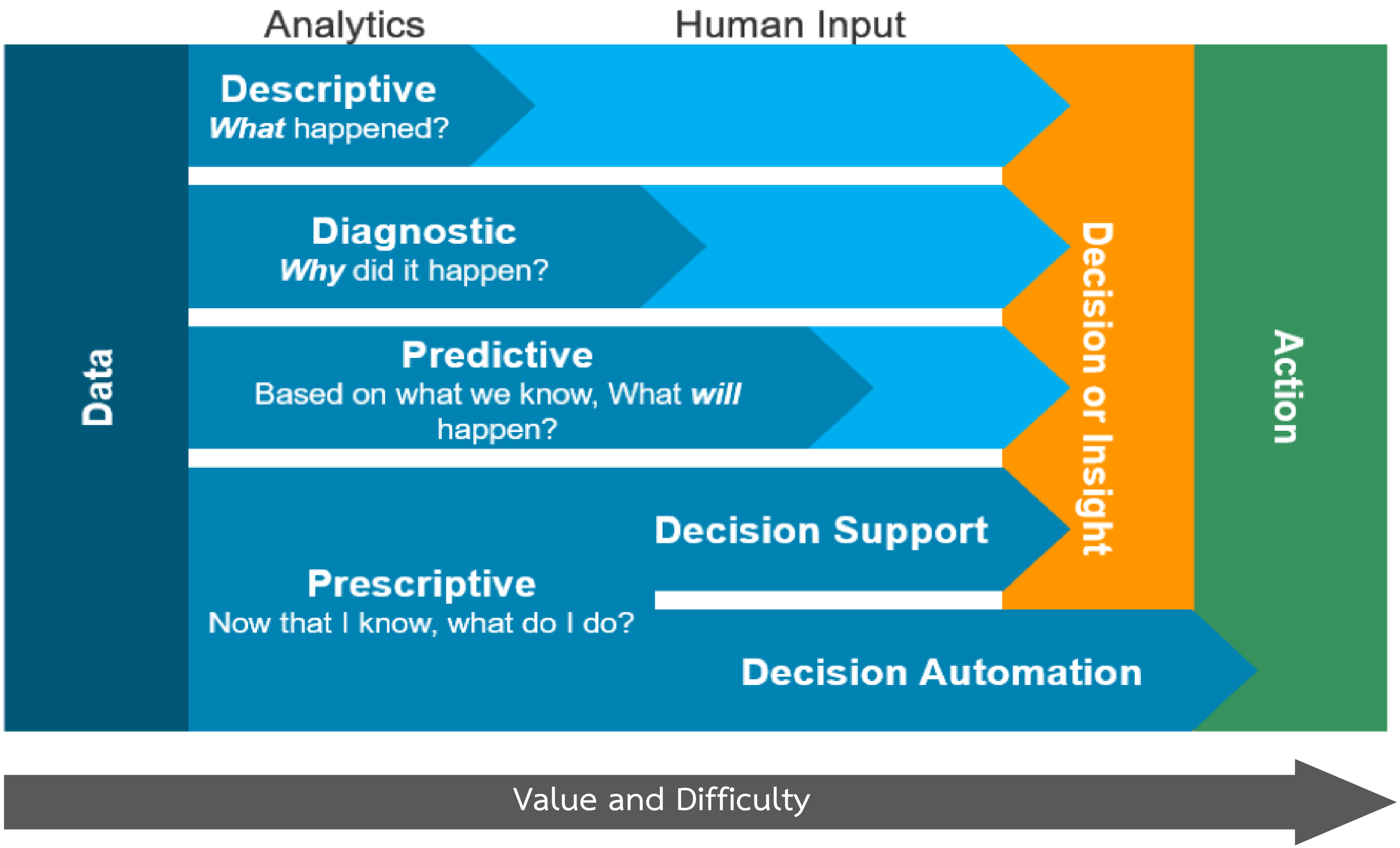
Dr. Rathachai Chawuthai

Department of Computer Engineering
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang

Agenda

- Classification
- Classifier I
- Evaluation Methods
- Classifiers II
- Regularization

Data Analytics



Ref:
- Four types of analytics capability (Gartner, 2014)
- (image) https://www.healthcatalyst.com/closed-loop-analytics-method-healthcare-data-insights

Machine Learning

Supervised Learning

Develop predictive model based on both input and output data

Unsupervised Learning

Develop predictive model based on both input and output data

Regression

- Linear Regression
- Polynomial Regression

Classification

- Decision Tree
- Logistic Regression
- Neural Network
- etc.

Clustering

- K-Means
- DB-SCAN
- etc.

Classification

Classes

Classes

Ref: • (images) http://www.ikea.com

Classes

Ref: • (images) http://www.ikea.com

Classes

Table

- 4 leg
- sit (yes)
- sleep (no)

- 4 legs
- mattress (yes)
- sleep (yes)

Bed

Sofa

- 4 legs
- mattress (yes)
- sit (yes)

Classes

Table

- 4 leg
- sit (yes)
- sleep (no)

Attributes/

Features

Class/

Label

- 4 legs
- mattress (yes)
- sleep (yes)

Bed

Sofa

- 4 legs
- mattress (yes)
- sit (yes)

Classes

Table

- 4 leg
- sit (yes)
- sleep (no)

- 4 legs
- mattress (yes)
- sleep (yes)

Bed

Sofa

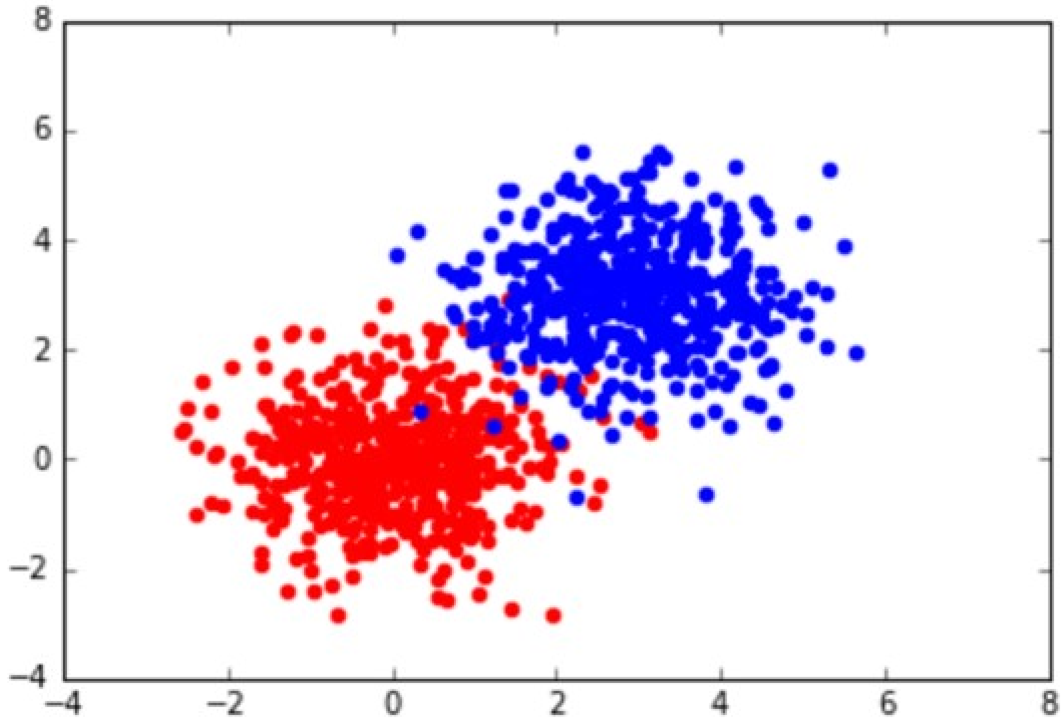
- 4 legs
- mattress (yes)
- sit (yes)

Classification

- Class named by Label
- Attributes or Features

Binary Classification

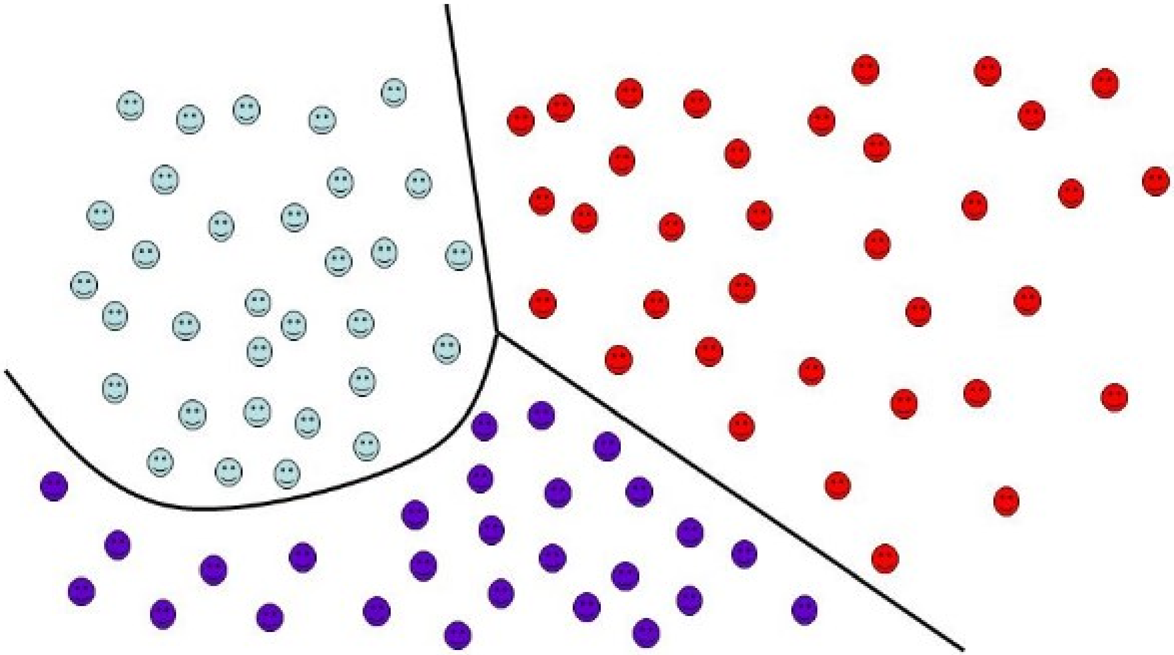
- Has 2 Classes
- We often have to deal with the simple task of Binary Classification. Some examples are:
  - Sentiment Analysis (positive/negative),
  - Spam Detection (spam/not-spam),
  - Fraud Detection (fraud/not-fraud).



Ref: • (image) http://nlpforhackers.io/classification-performance-metrics/

Multi-Class Classification

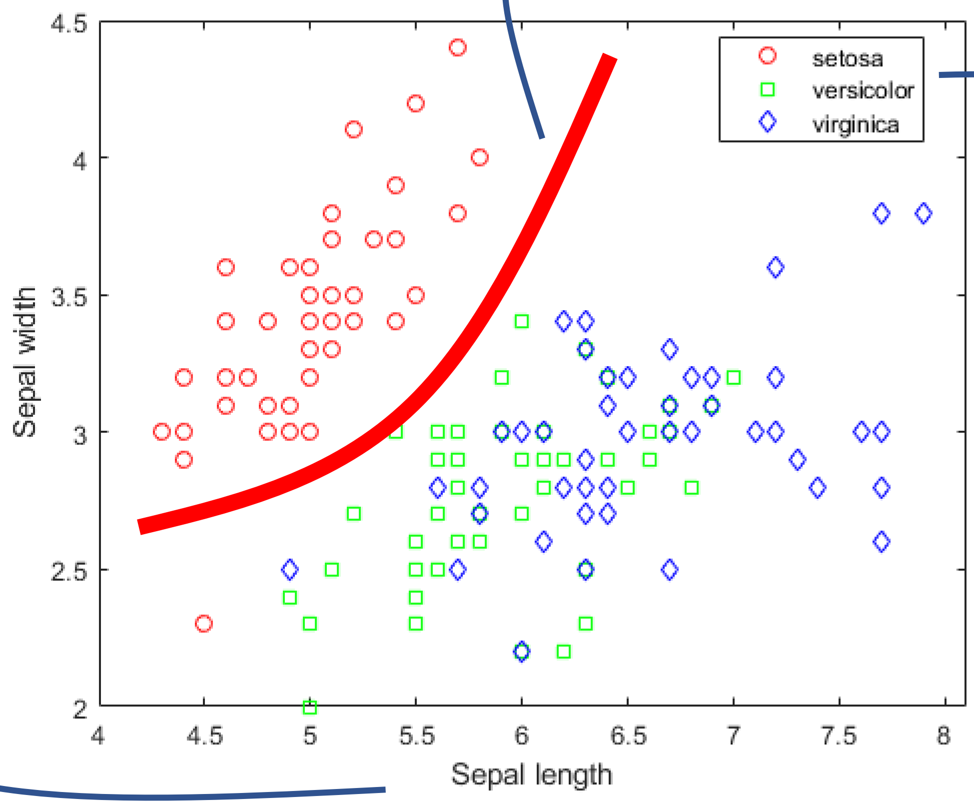
- Has more than 2 classes
- For example,
  - Genres of Movies
  - Grades
  - Sentiment Analysis (Positive, Natural, Negative)
  - etc.



Ref: • (image) http://slideplayer.com/slide/5156750/

Classification

Classifier

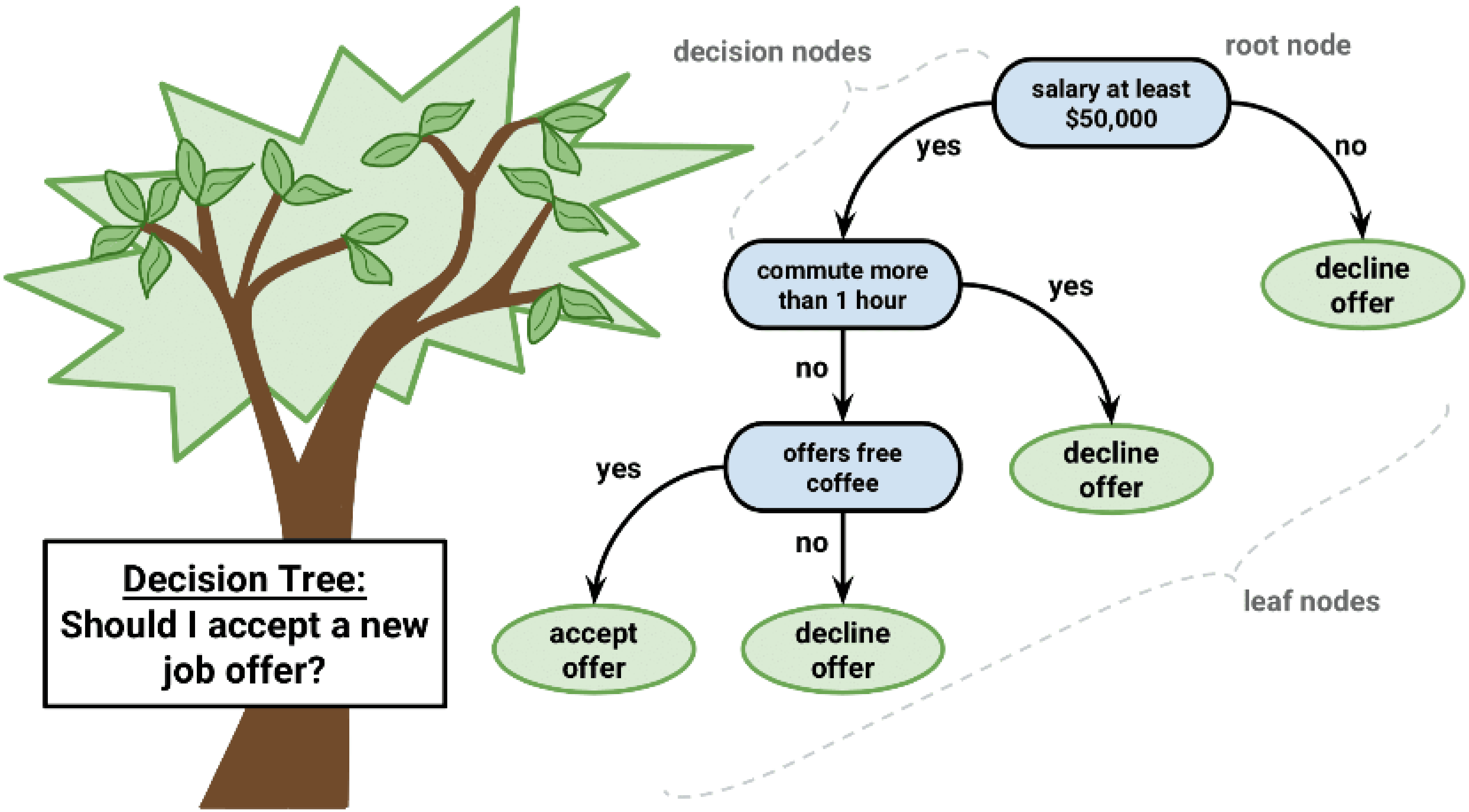


Labels

Features

Classifier I

Decision Tree



Ref: • (image) http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/

Decision Tree

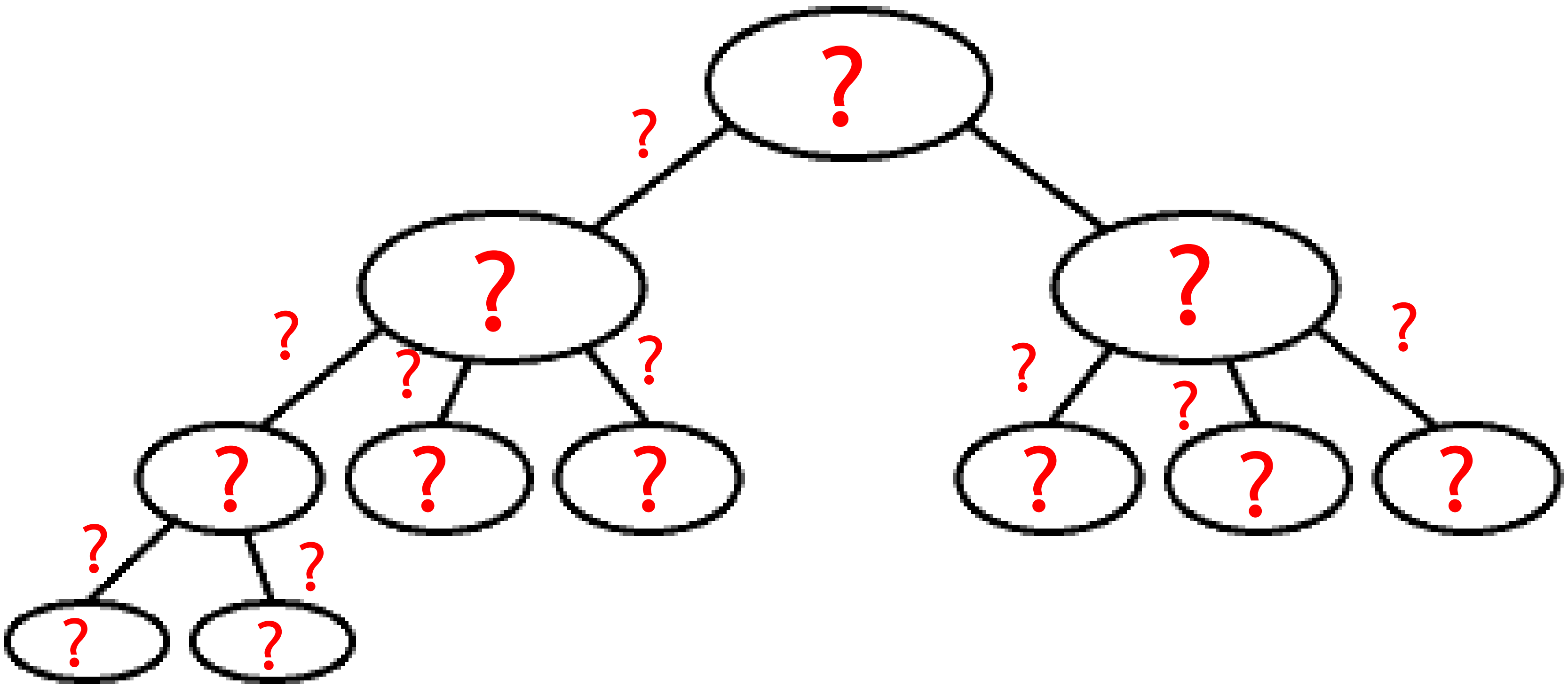
- A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.
- Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.
- A decision tree consists of three types of nodes:
  - Decision nodes – typically represented by squares
  - Chance nodes – typically represented by circles
  - End nodes – typically represented by triangles

Ref: • (content) https://en.wikipedia.org/wiki/Decision_tree

Case

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	**YES**
2	very_high	YES	YES	**YES**
3	normal	no	no	**no**
4	high	YES	YES	**YES**
5	high	no	YES	**no**
6	normal	YES	no	**no**
7	normal	no	YES	**no**

Tree



Case

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	**YES**
2	very_high	YES	YES	**YES**
3	normal	no	no	**no**
4	high	YES	YES	**YES**
5	high	no	YES	**no**
6	normal	YES	no	**no**
7	normal	no	YES	**no**

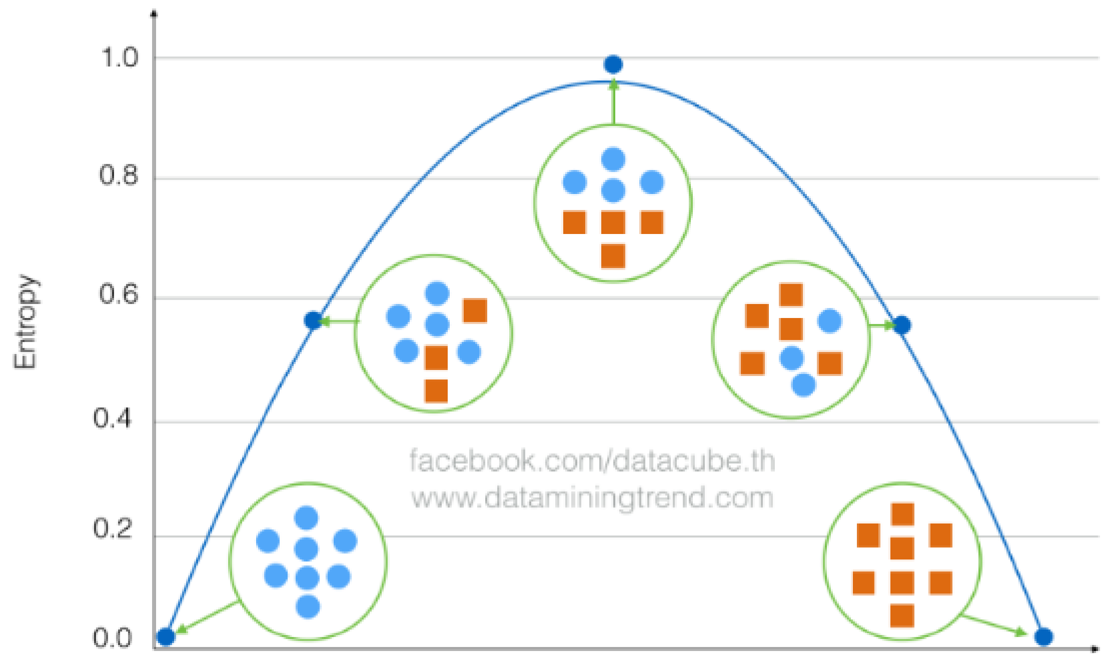
Case

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	**YES**
2	very_high	YES	YES	**YES**
3	normal	no	no	**no**
4	high	YES	YES	**YES**
5	high	no	YES	**no**
6	normal	YES	no	**no**
7	normal	no	YES	**no**

Information Gain

- Same things --> Low
- Different things -->High

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$



Ref: • (image) http://dataminingtrend.com

Decision Tree: Construction

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	YES
2	very_high	YES	YES	YES
3	normal	no	no	no
4	high	YES	YES	YES
5	high	no	YES	no
6	normal	YES	no	no
7	normal	no	YES	no

Finding entropy H of every attribute

1) Headache

Headache = YES

Headache = no

H(Flu | Headache) = - [ ] - [ ]

Decision Tree: Construction

Finding entropy H of every attribute

1) Headache

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	YES
2	very_high	YES	YES	YES
3	normal	no	no	no
4	high	YES	YES	YES
5	high	no	YES	no
6	normal	YES	no	no
7	normal	no	YES	no

Headache = YES

Headache = no

H(Flu | Headache) = - $\frac{4}{7}$ $\frac{3}{7}$

Decision Tree: Construction

Finding entropy H of every attribute

1) Headache

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	YES
2	very_high	YES	YES	YES
3	normal	no	no	no
4	high	YES	YES	YES
5	high	no	YES	no
6	normal	YES	no	no
7	normal	no	YES	no

Headache = YES   Headache = no

$$H(Flu \mid Headache) = -\frac{4}{7}\left(\underbrace{\boxed{\quad\quad}}_{Flu=YES} + \underbrace{\boxed{\quad\quad}}_{Flu=no}\right) - \frac{3}{7}\left(\underbrace{\boxed{\quad\quad}}_{Flu=YES} + \underbrace{\boxed{\quad\quad}}_{Flu=no}\right)$$

Decision Tree: Construction

Finding entropy H of every attribute

1) Headache

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	YES
2	very_high	YES	YES	YES
3	normal	no	no	no
4	high	YES	YES	YES
5	high	no	YES	no
6	normal	YES	no	no
7	normal	no	YES	no

Headache = YES

Headache = no

$$H(\text{Flu} \mid \text{Headache}) = -\frac{4}{7}\left(\boxed{\frac{3}{7}log_2\left(\frac{3}{4}\right)} + \boxed{\frac{1}{7}log_2\left(\frac{1}{4}\right)}\right) - \frac{3}{7}\left[\qquad + \qquad\right]$$

Flu=YES

Flu=no

Decision Tree: Construction

Finding entropy H of every attribute

1) Headache

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	YES
2	very_high	YES	YES	YES
3	normal	no	no	no
4	high	YES	YES	YES
5	high	no	YES	no
6	normal	YES	no	no
7	normal	no	YES	no

Headache = YES

Headache = no

H(Flu | Headache) =

$$- \boxed{\frac{4}{7}\left(\frac{3}{4}log_2\left(\frac{3}{4}\right) + \frac{1}{4}log_2\left(\frac{1}{4}\right)\right)} - \boxed{\frac{3}{7}\left(\underbrace{\boxed{\frac{0}{3}log_2\left(\frac{0}{4}\right)}}_{\text{Flu=YES}} + \underbrace{\boxed{\frac{3}{3}log_2\left(\frac{3}{3}\right)}}_{\text{Flu=no}}\right)}$$

Decision Tree: Construction

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	**YES**
2	very_high	YES	YES	**YES**
3	normal	no	no	**no**
4	high	YES	YES	**YES**
5	high	no	YES	**no**
6	normal	YES	no	**no**
7	normal	no	YES	**no**

Finding entropy H of every attribute

1) Headache

H(Flu | Headache) =

$$- \frac{4}{7}\left[\frac{3}{4}log_2\left(\frac{3}{4}\right) + \frac{1}{4}log_2\left(\frac{1}{4}\right)\right] - \frac{3}{7}\left[\cancelto{0}{\frac{0}{3}log_2\left(\frac{0}{4}\right)} + \cancelto{0}{\frac{3}{3}log_2\left(\frac{3}{3}\right)}\right]$$

= 0.464

Decision Tree: Construction

Finding entropy H

of every attribute

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	**YES**
2	very_high	YES	YES	**YES**
3	normal	no	no	**no**
4	high	YES	YES	**YES**
5	high	no	YES	**no**
6	normal	YES	no	**no**
7	normal	no	YES	**no**

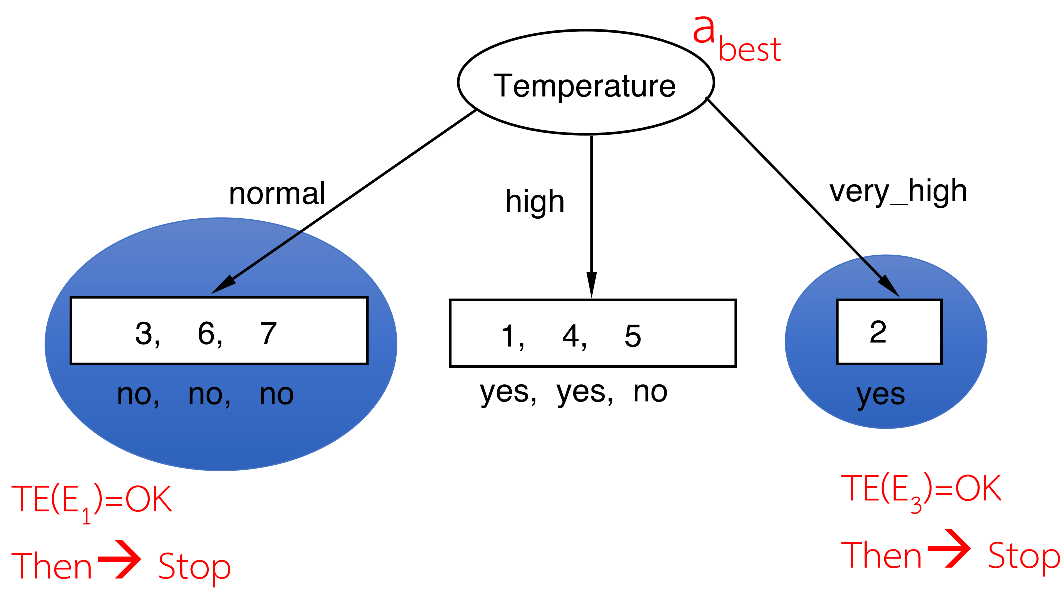
H(Flu | Headache) = 0.464

$\boxed{\text{H(Flu | Temperature) = 0.394}}$ Lowest

H(Flu | Nausea) = 0.620

Decision Tree: Construction

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	**YES**
2	very_high	YES	YES	**YES**
3	normal	no	no	**no**
4	high	YES	YES	**YES**
5	high	no	YES	**no**
6	normal	YES	no	**no**
7	normal	no	YES	**no**



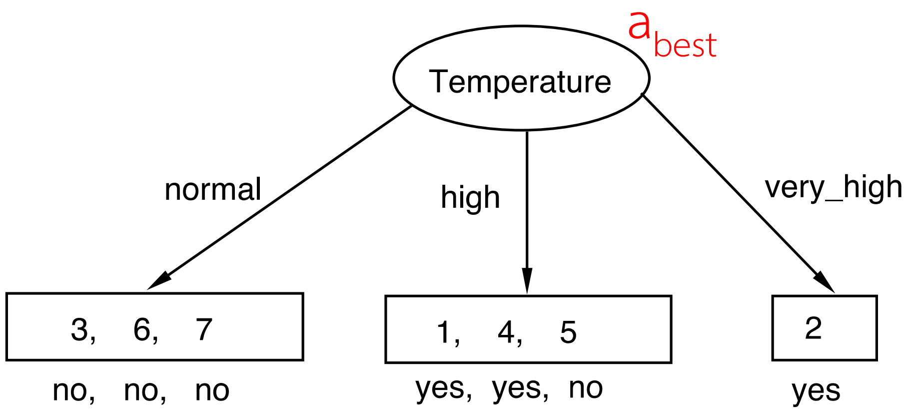
H(Flu | Headache) = 0.464

$\boxed{\text{H(Flu | Temperature) = 0.394}}$ **Lowest**

H(Flu | Nausea) = 0.620

Decision Tree: Construction

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	**YES**
2	very_high	YES	YES	**YES**
3	normal	no	no	**no**
4	high	YES	YES	**YES**
5	high	no	YES	**no**
6	normal	YES	no	**no**
7	normal	no	YES	**no**



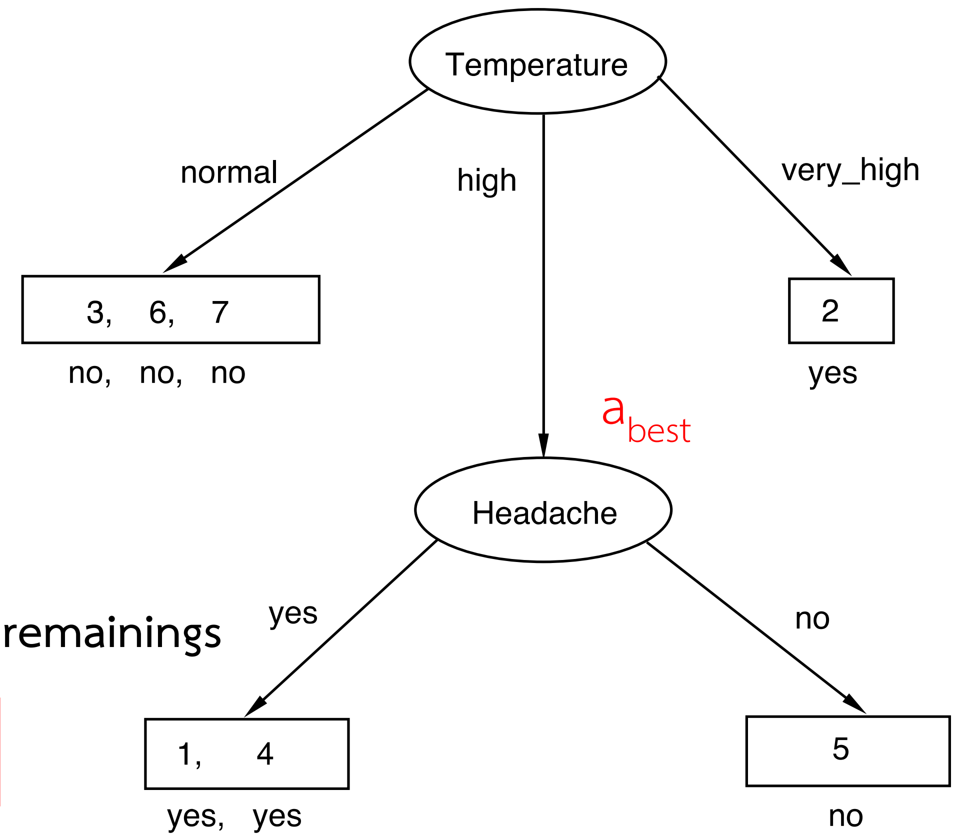
Finding entropy H of attributes from the remainings

$\boxed{H(Flu_{Temperature\ =\ high} \mid Headache) = 0}$ Lowest

$H(Flu_{Temperature\ =\ high} \mid Nausea) = 0.667$

Decision Tree: Construction

CASE	TEMPERATURE	HEADACHE	NAUSEA	Flu
1	high	YES	no	YES
2	very_high	YES	YES	YES
3	normal	no	no	no
4	high	YES	YES	YES
5	high	no	YES	no
6	normal	YES	no	no
7	normal	no	YES	no



Finding entropy H of attributes from the remainings

$$\boxed{H(Flu_{Temperature\ =\ high} \mid Headache) = 0}$$

$$H(Flu_{Temperature\ =\ high} \mid Nausea) = 0.667$$

Evaluation Methods

Accuracy

- Accuracy is the most popular performance measure used and for good reason. It’s extremely helpful, simple to compute and to understand. It is the proportion of the correctly classified samples and all the samples.

```
from sklearn.metrics import accuracy_score

print accuracy_score(y_test, model.predict(X_test))   # 0.98
```

Ref: • (content) http://nlpforhackers.io/classification-performance-metrics/

TP | TN | FP | FN

- There are other ways to measure different aspects of performance. In classic machine learning nomenclature, when we're dealing with binary classification, the classes are: **positive and negative**. Think of these classes in the context of disease detection:
  - **positive** – we predict the disease is present
  - **negative** – we predict the disease is not present.
- Let's now define some notations:
  - **TP** – True Positives (Samples the classifier has correctly classified as positives)
  - **TN** – True Negatives (Samples the classifier has correctly classified as negatives)
  - **FP** – False Positives (Samples the classifier has incorrectly classified as positives)
  - **FN** – False Negatives (Samples the classifier has incorrectly classified as negatives)

Ref: • (content) http://nlpforhackers.io/classification-performance-metrics/

TP | TN | FP | FN

	predicted: YES	predicted: NO
actual: YES	4211 *(true positive)*	181 *(false negative)*
actual: NO	94 *(false positive)*	1207 *(true negative)*

Ref: • (image) https://data36.com/predictive-analytics-101-part-2/

Evaluation Methods

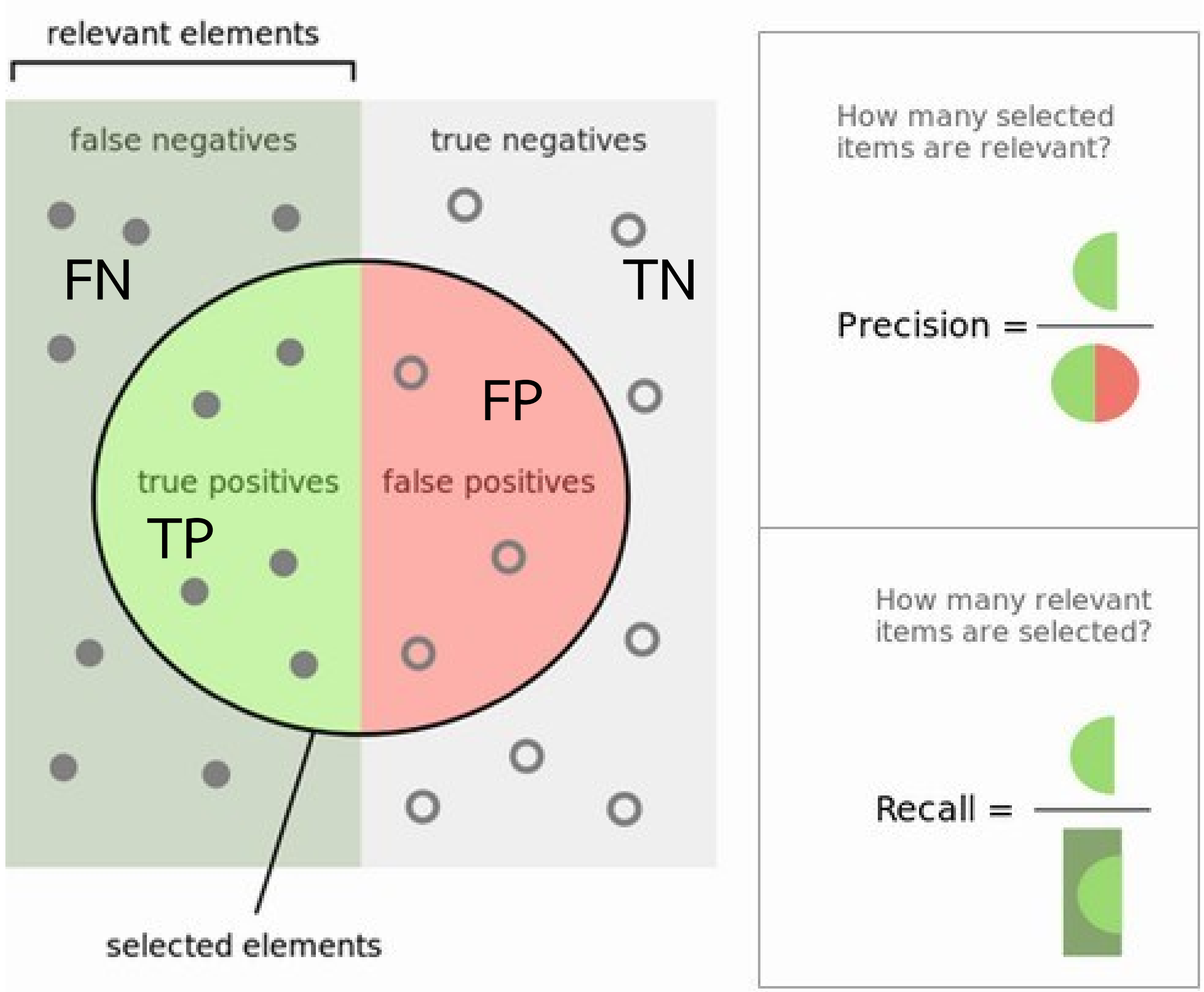
$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Evaluation Methods

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

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

Precision & Recall



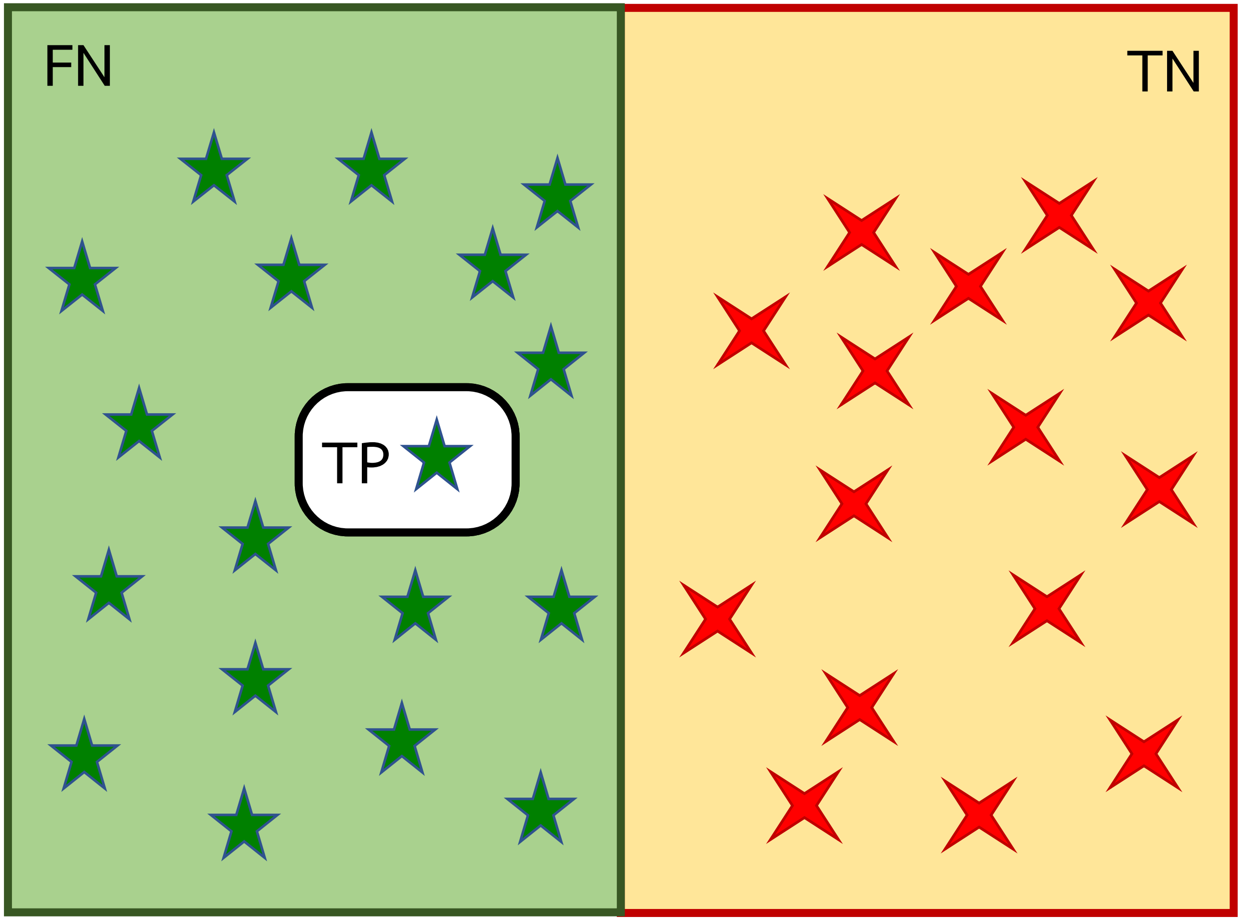
Ref: • (image)http://www.aprilzephyr.com/blog/10242016/21個經典數據科學問題及答案-轉

Question

Recall = 0.99

is it good?

Perfect



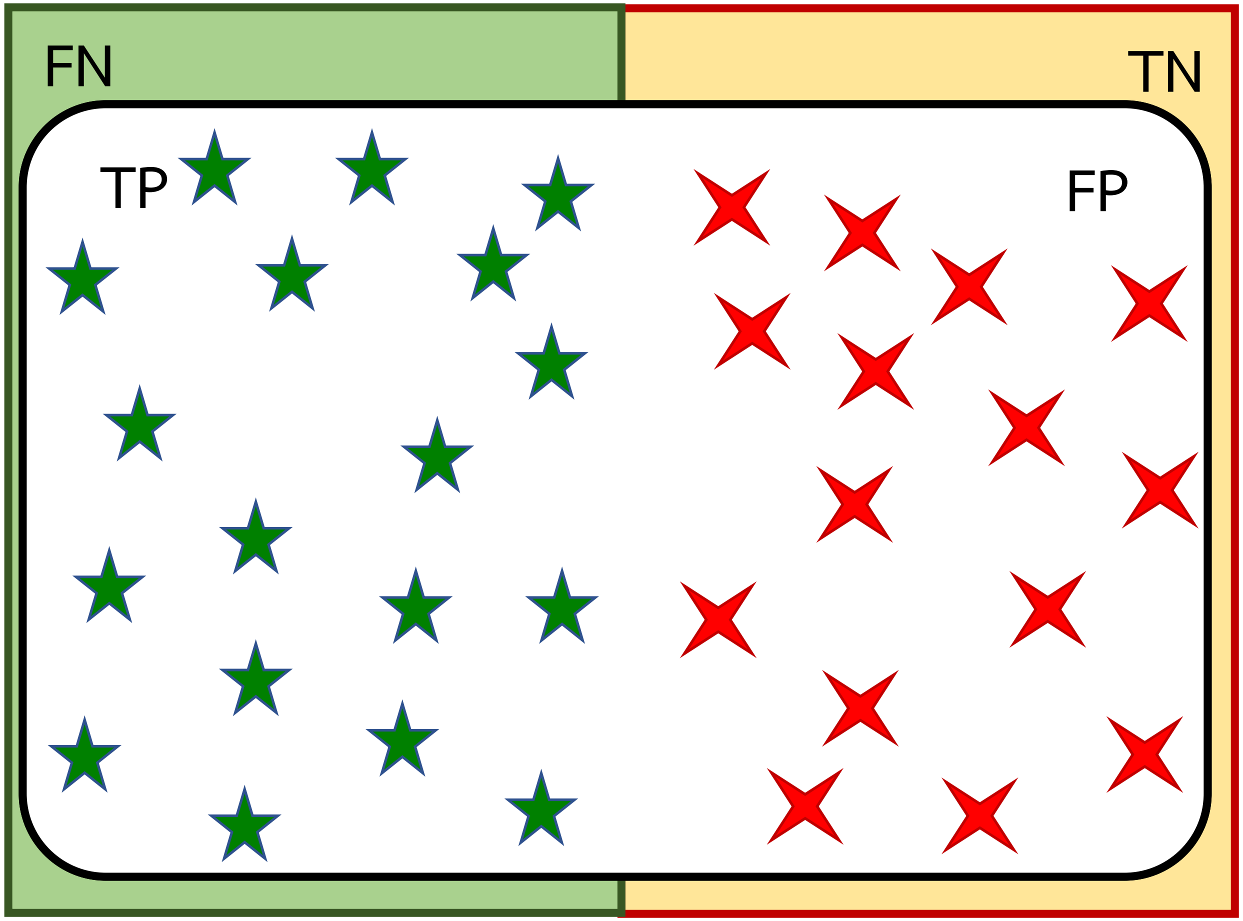
Precision = 1

Question

Recall = 0.99

is it good?

Perfect



Recall = 1

Evaluation Methods

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

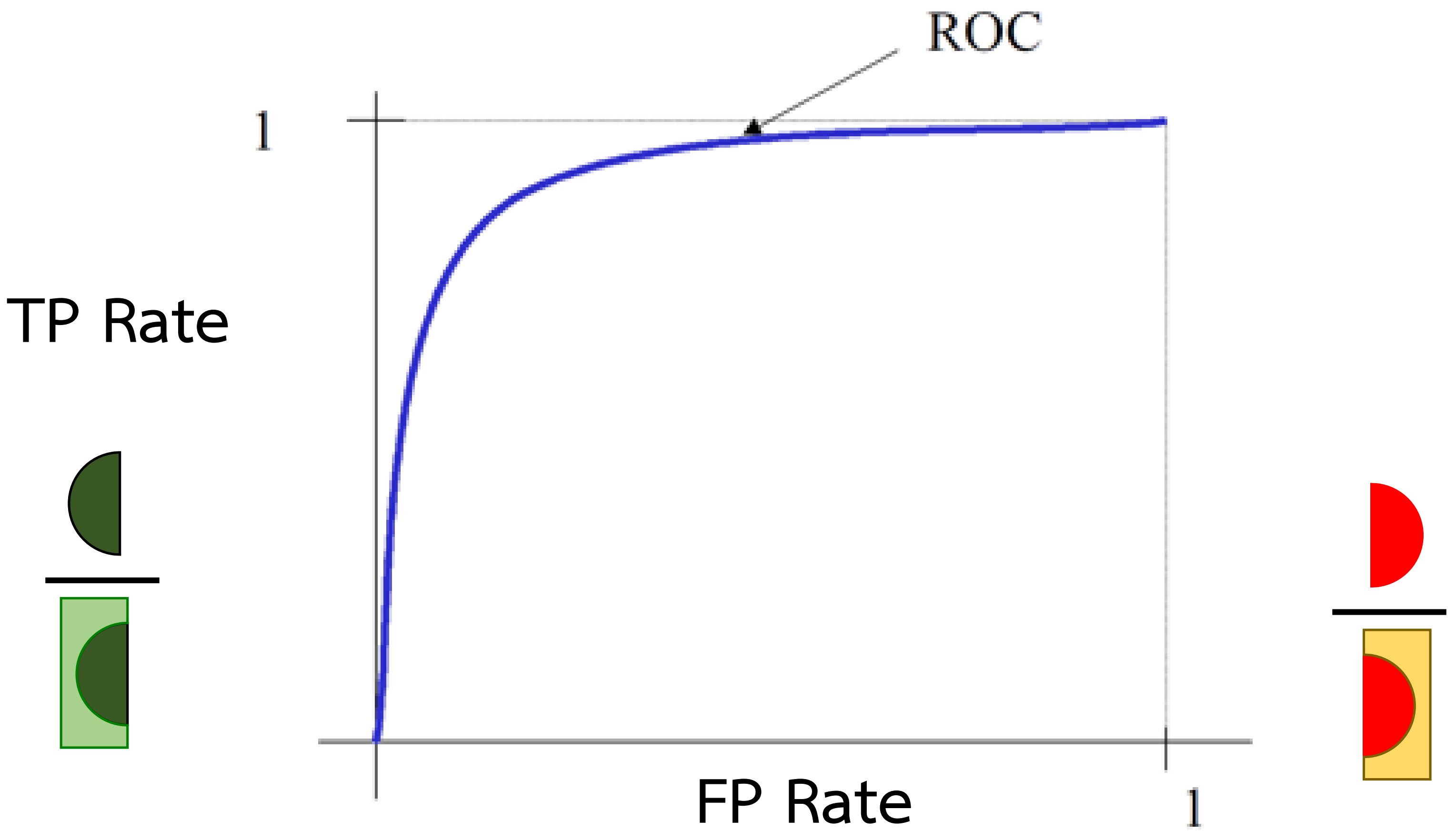
$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Perfect

ROC Curve

- Receive Operating Characteristic

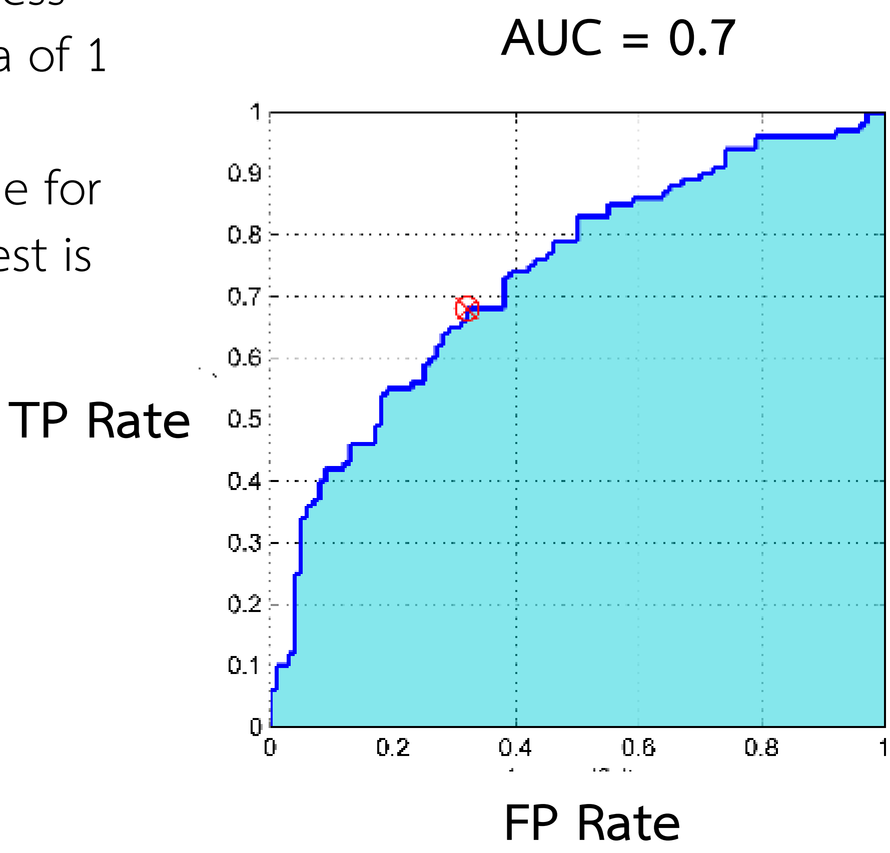


Ref: • (image) https://www.projectrhea.org/rhea/index.php/Neyman-Pearson_Lemma_and_Receiver_Operating_Characteristic_Curve

AUC : Area Under the ROC Curve

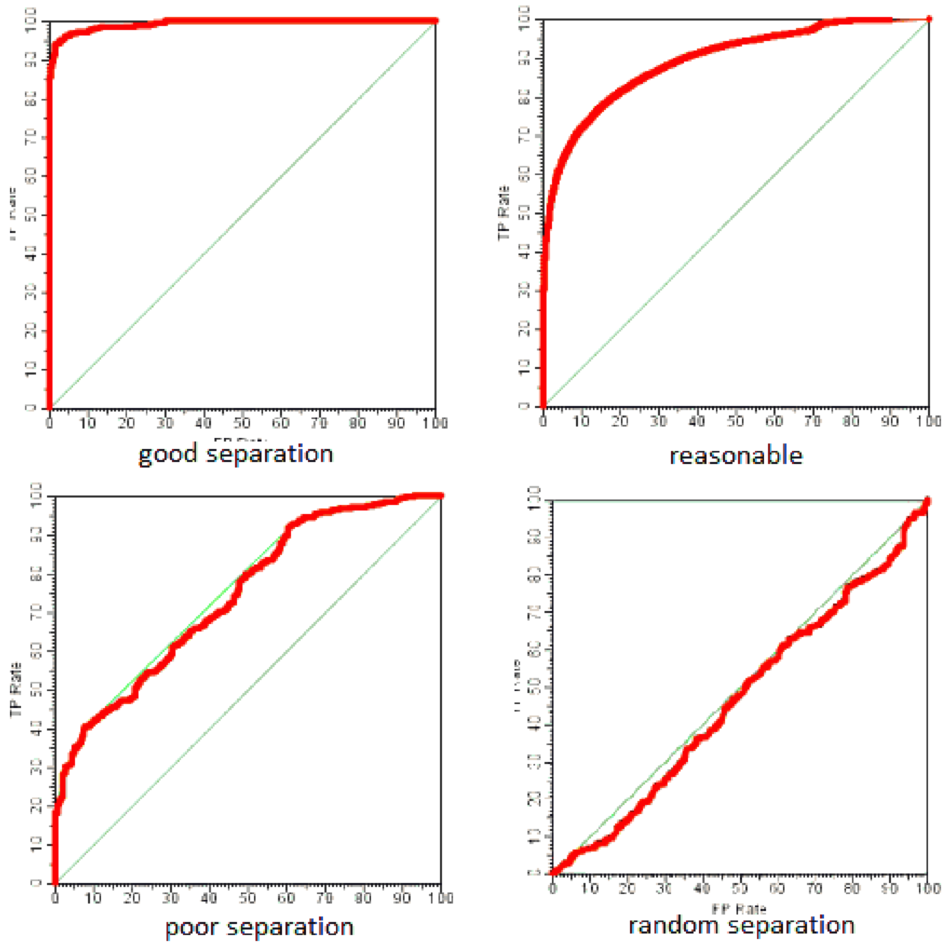
The graph at right shows three ROC curves representing excellent, good, and worthless tests plotted on the same graph. An area of 1 represents a perfect test; an area of 0.5 represents a worthless test. A rough guide for classifying the accuracy of a diagnostic test is the traditional academic point system:

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



Ref:
- (content) http://gim.unmc.edu/dxtests/roc3.htm
- (image) http://molevol.altervista.org/blog/roc-curve-area-roc-curve-auc/

AUC



Ref: • (image) http://mlwiki.org/index.php/ROC_Analysis

Confusion Matrix

Predicted Class

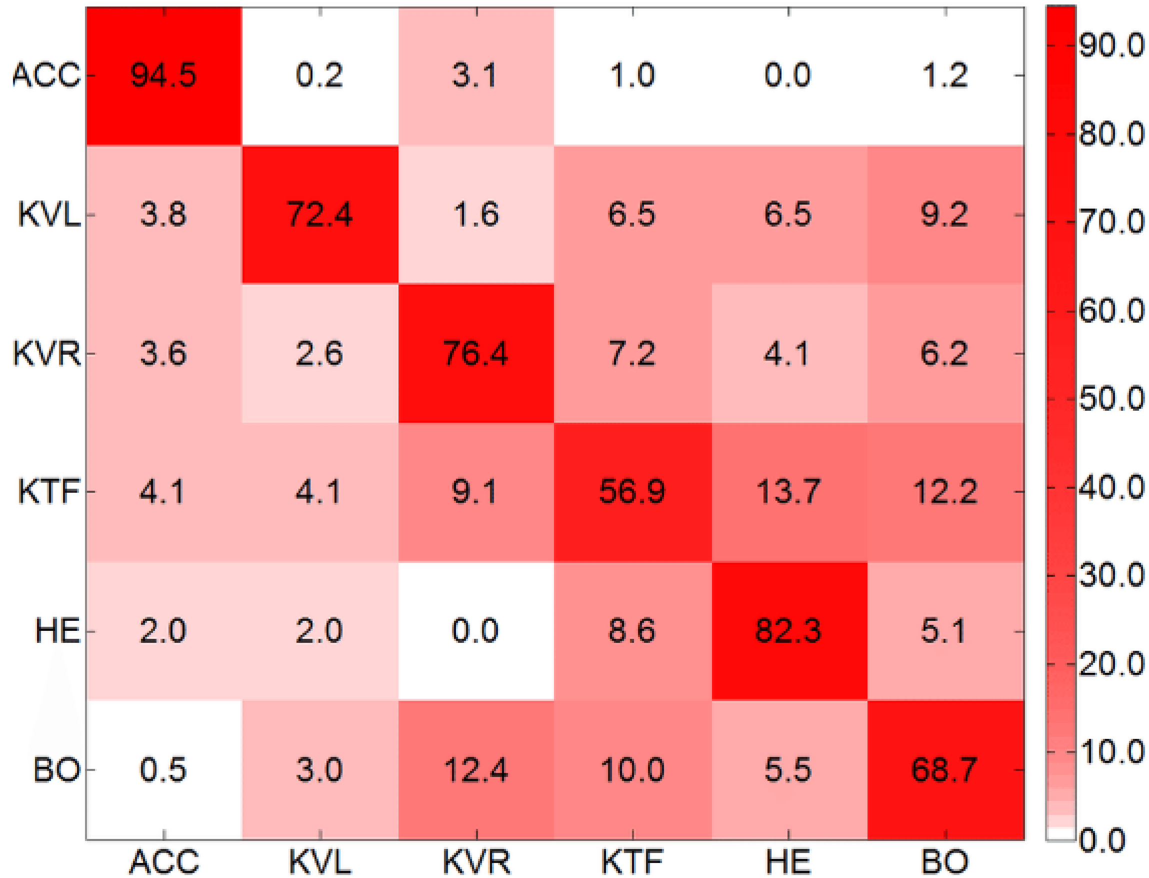
Actual Class

	Cat	Dog	Rabbit
Cat	5	2	0
Dog	3	3	2
Rabbit	0	1	11

Ref: • (image) http://python-data-science.readthedocs.io/en/latest/evaluation.html

Confusion Matrix : Heatmap

Actual Class

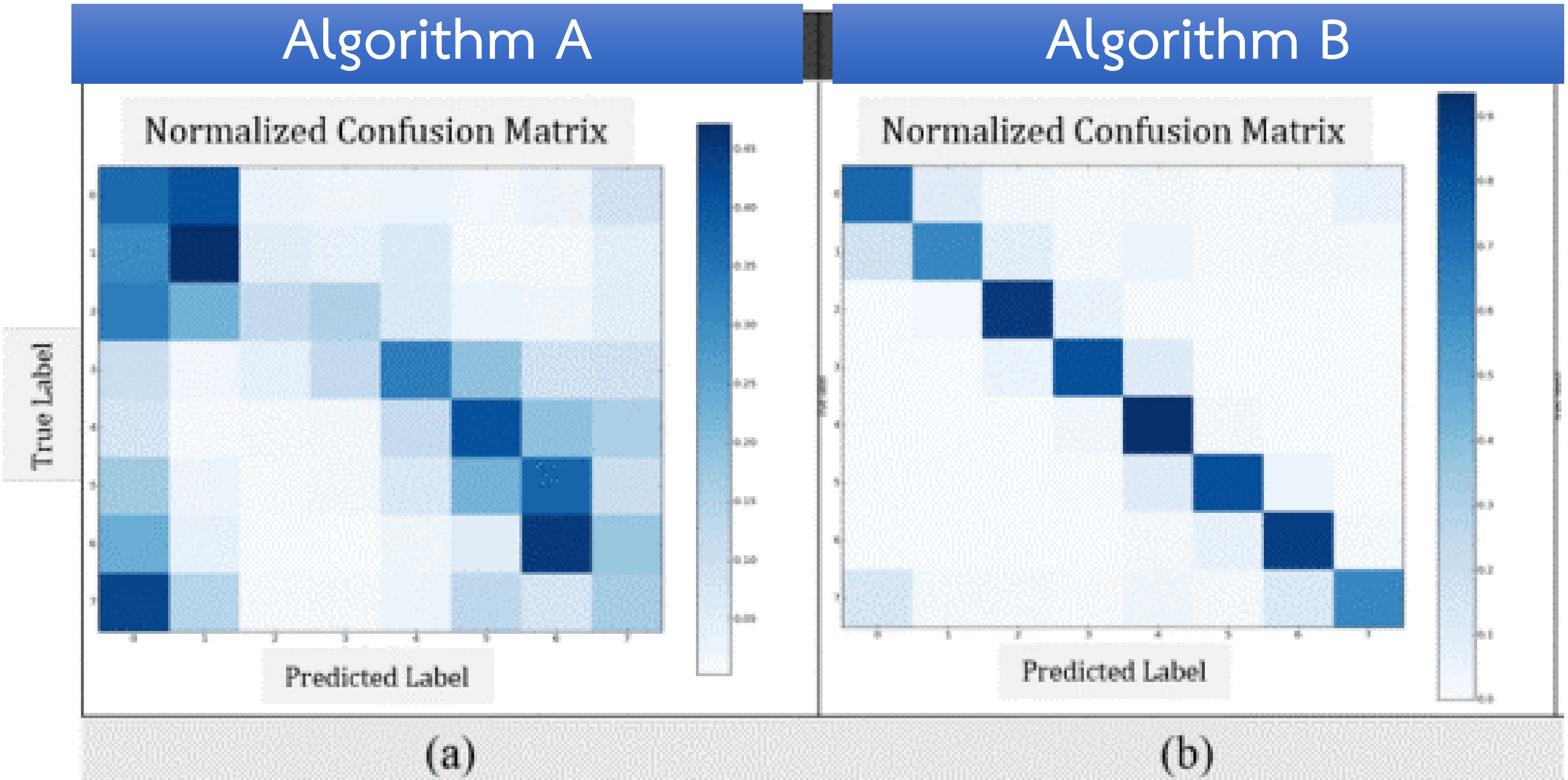


Predicted Class

Ref: • (image) https://www.researchgate.net/figure/Figure-3-Heat-map-confusion-matrix-for-multi-label-exact-deviation-classification-results-using_316316283_fig3

Confusion Matrix : Comparison

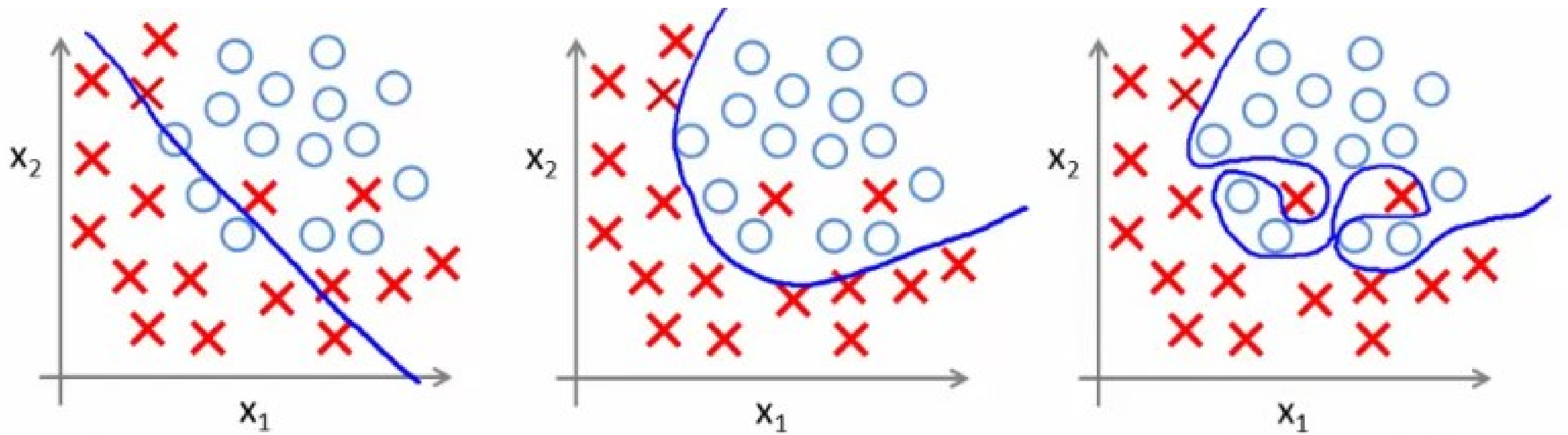
Which one is better?



Ref: • (image) http://ieeexplore.ieee.org/document/7279167/

Classification Methods II

Underfitting vs.. Overfitting



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots$

UNDERFITTING
(high bias)

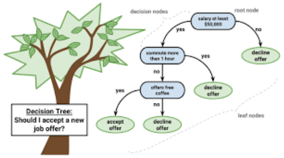
OVERFITTING
(high variance)

Ref: • (image) http://www.holehouse.org/mlclass/07_Regularization.html

Classification Techniques

- Linear classifiers
  - Fisher's linear discriminant
  - Logistic regression
  - Naive Bayes classifier
  - Perceptron
- Support vector machines
- Quadratic classifiers
- Kernel estimation
  - k-nearest neighbor
- Boosting (meta-algorithm)
- Decision trees
- Neural networks
- Learning vector quantization

Decision Tree



`DecisionTreeClassifier` is a class capable of performing multi-class classification on a dataset.

As with other classifiers, `DecisionTreeClassifier` takes as input two arrays: an array X, sparse or dense, of size `[n_samples, n_features]` holding the training samples, and an array Y of integer values, size `[n_samples]`, holding the class labels for the training samples:

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

After being fitted, the model can then be used to predict the class of samples:

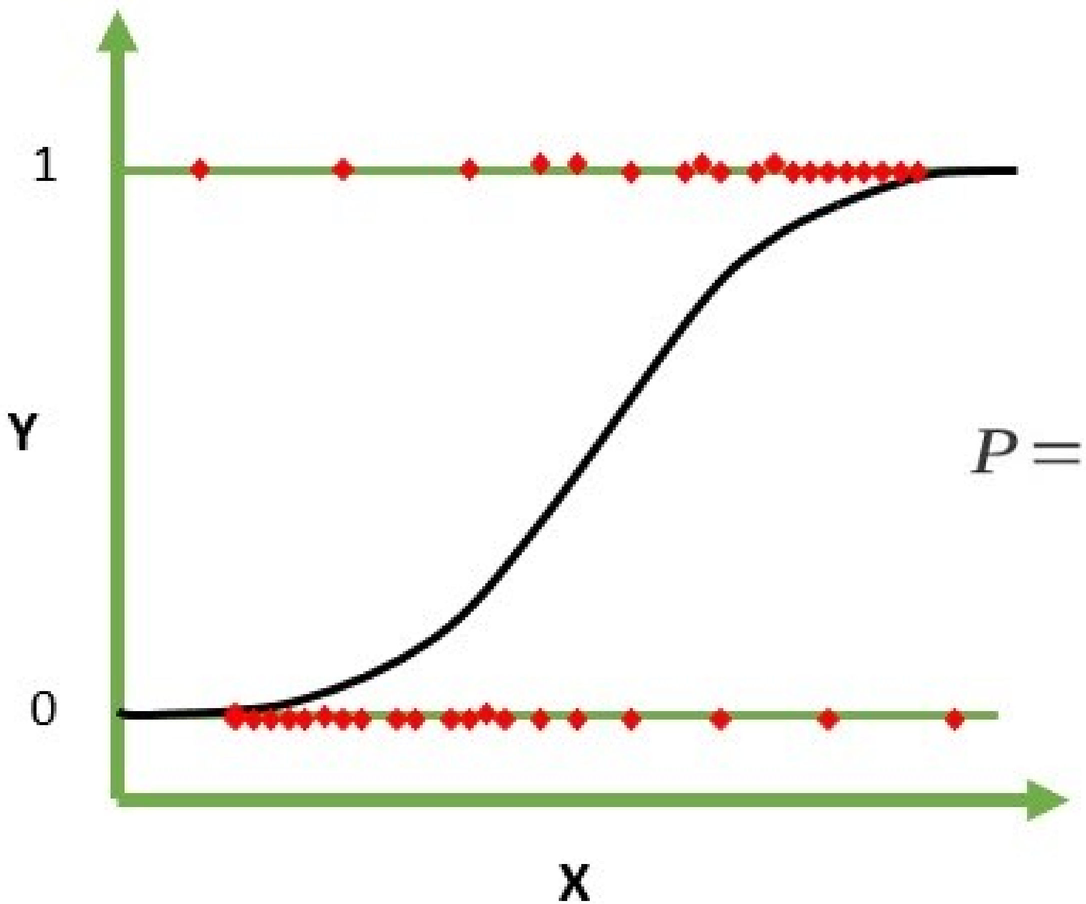
```
>>> clf.predict([[2., 2.]])
array([1])
```

Alternatively, the probability of each class can be predicted, which is the fraction of training samples of the same class in a leaf:

```
>>> clf.predict_proba([[2., 2.]])
array([[ 0.,  1.]])
```

Ref:
- (image) http://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/
- (content) http://scikit-learn.org/stable/modules/tree.html

Logistic Regression



$$P = \frac{e^{(\beta + \alpha_1 X_1 + \ldots \alpha_n X_n)}}{1 + e^{(\beta + \alpha_1 X_1 + \ldots \alpha_n X_n)}}$$

Ref: • (image) https://techdifferences.com/difference-between-linear-and-logistic-regression.html

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics


clf = LogisticRegression()
clf.fit(X_train, y_train)
```

Ref: • (content) https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Logistic Regression

```
from sklearn import model_selection
from sklearn.model_selection import cross_val_score
kfold = model_selection.KFold(n_splits=10, random_state=7)

clf = LogisticRegression()

results = model_selection.cross_val_score
       (clf, X, y, cv=kfold, scoring= 'accuracy’)

print("10-fold accuracy: %.3f" % (results.mean()))
```

Ref: • (content) https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Logistic Regression

```
from sklearn import model_selection
from sklearn.model_selection import cross_val_score
kfold = model_selection.KFold(n_splits=10, random_state=7)

clf = LogisticRegression()

results = model_selection.cross_val_score
       (clf, X, y, cv=kfold, scoring= 'accuracy’)

print("10-fold accuracy: %.3f" % (results.mean()))
```

Ref: • (content) https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Logistic Regression

```
from sklearn.metrics import confusion_matrix

confusion_matrix = confusion_matrix(y_test, y_pred)

print(confusion_matrix)
```

```
[[10872,  109]
 [ 1122,  254]]
```

The result is telling us that we have 10872+254 correct predictions and 1122+109 incorrect predictions.

Logistic Regression

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
             precision    recall  f1-score

          0       0.91      0.99      0.95
          1       0.70      0.18      0.29

avg / total       0.88      0.90      0.87
```

Ref: • (content) https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8

Naive Bayes

- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*)
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c)\,P(c)}{P(x)}$$

Posterior Probability

Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Naive Bayes

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Frequency Table		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

$$P(x \mid c) = P(Sunny \mid Yes) = 3/9 = 0.33$$

Likelihood Table		Play Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/14	

$$P(x) = P(Sunny) = 5/14 = 0.36$$

$$P(c) = P(Yes) = 9/14 = 0.64$$

Posterior Probability: $P(c \mid x) = P(Yes \mid Sunny) = 0.33 \times 0.64 \div 0.36 = 0.60$

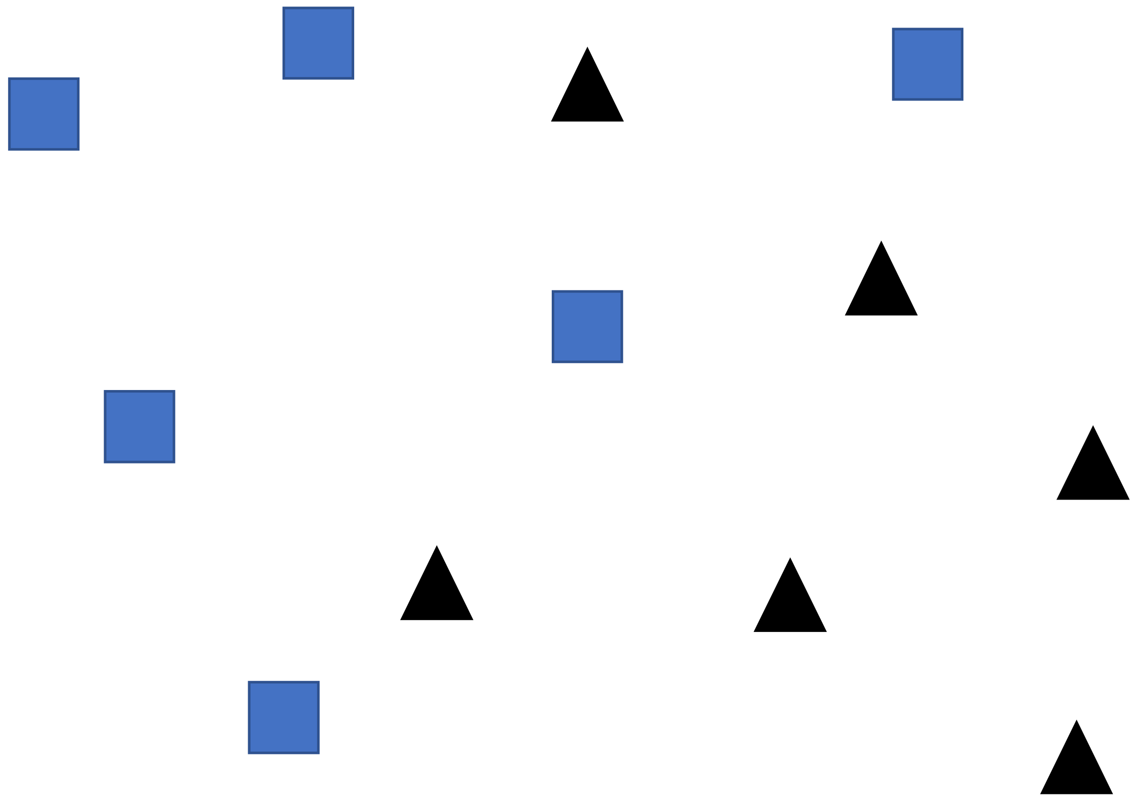
Ref: • (image) http://www.saedsayad.com/naive_bayesian.htm

Naive Bayes

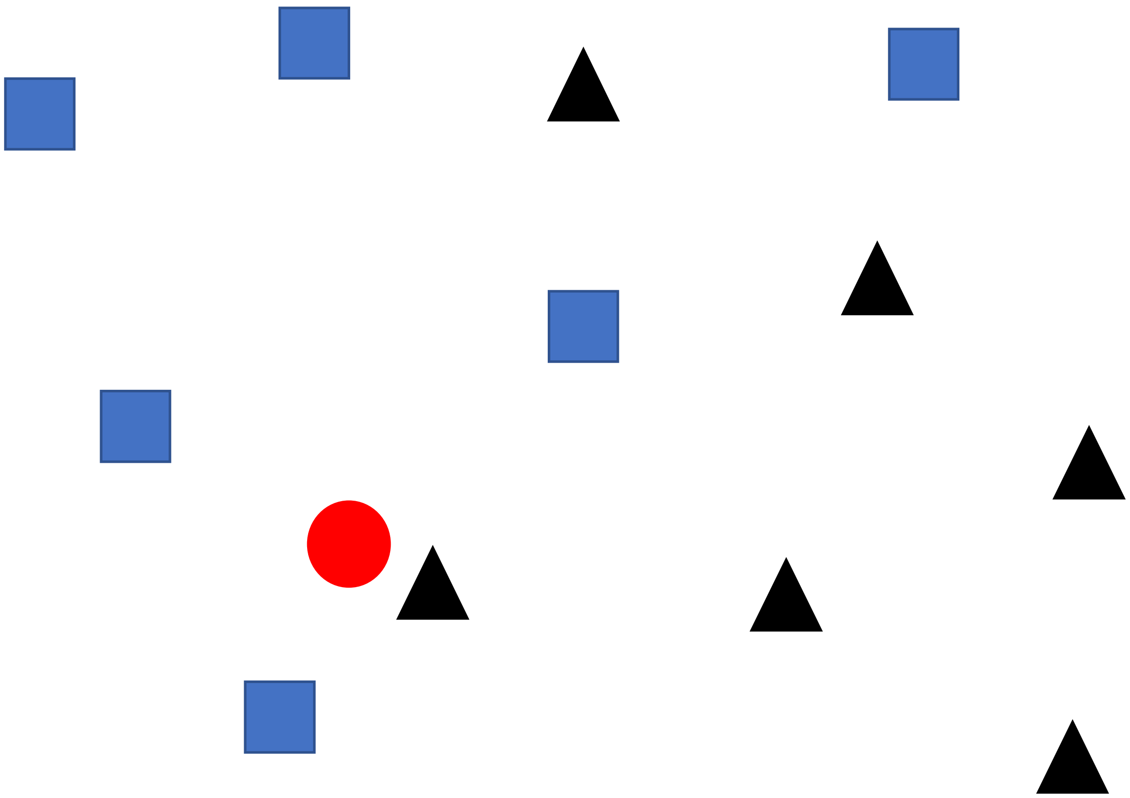
.

```
>>> import numpy as np
>>> X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
>>> Y = np.array([1, 1, 1, 2, 2, 2])
>>> from sklearn.naive_bayes import GaussianNB
>>> clf = GaussianNB()
>>> clf.fit(X, Y)
GaussianNB(priors=None)
>>> print(clf.predict([[-0.8, -1]]))
[1]
```

K-Nearest Neighbor

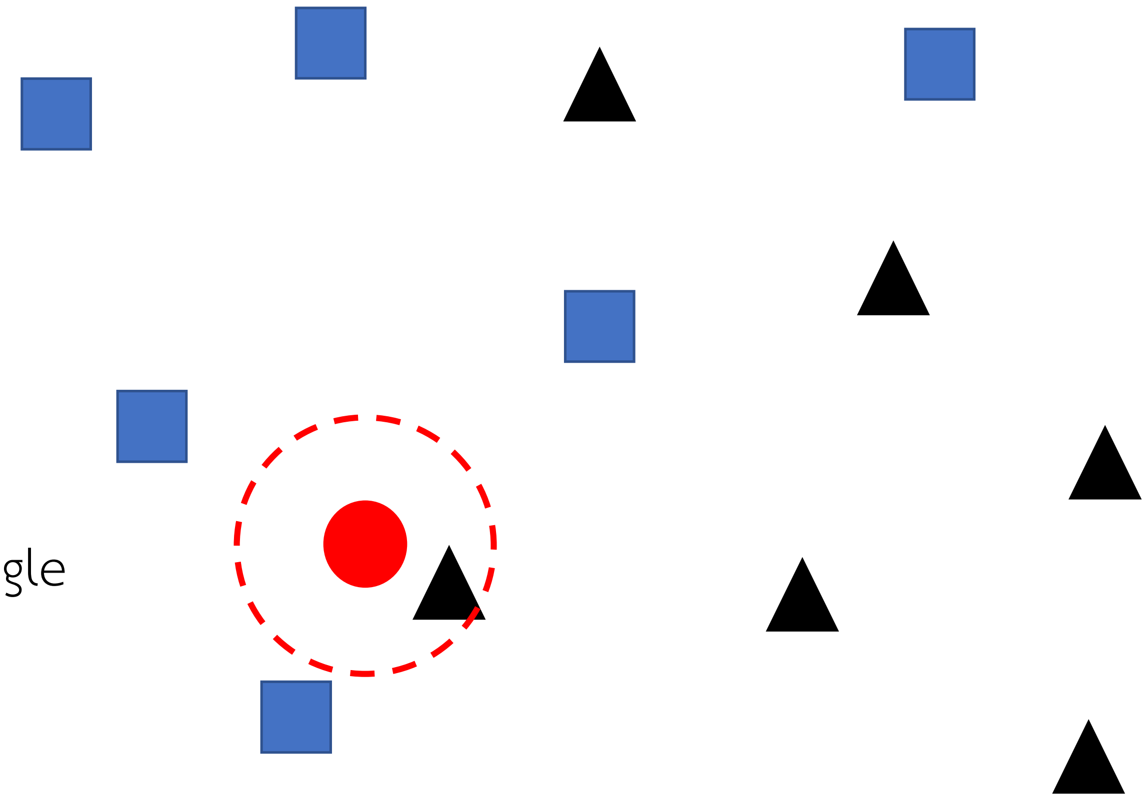


K-Nearest Neighbor



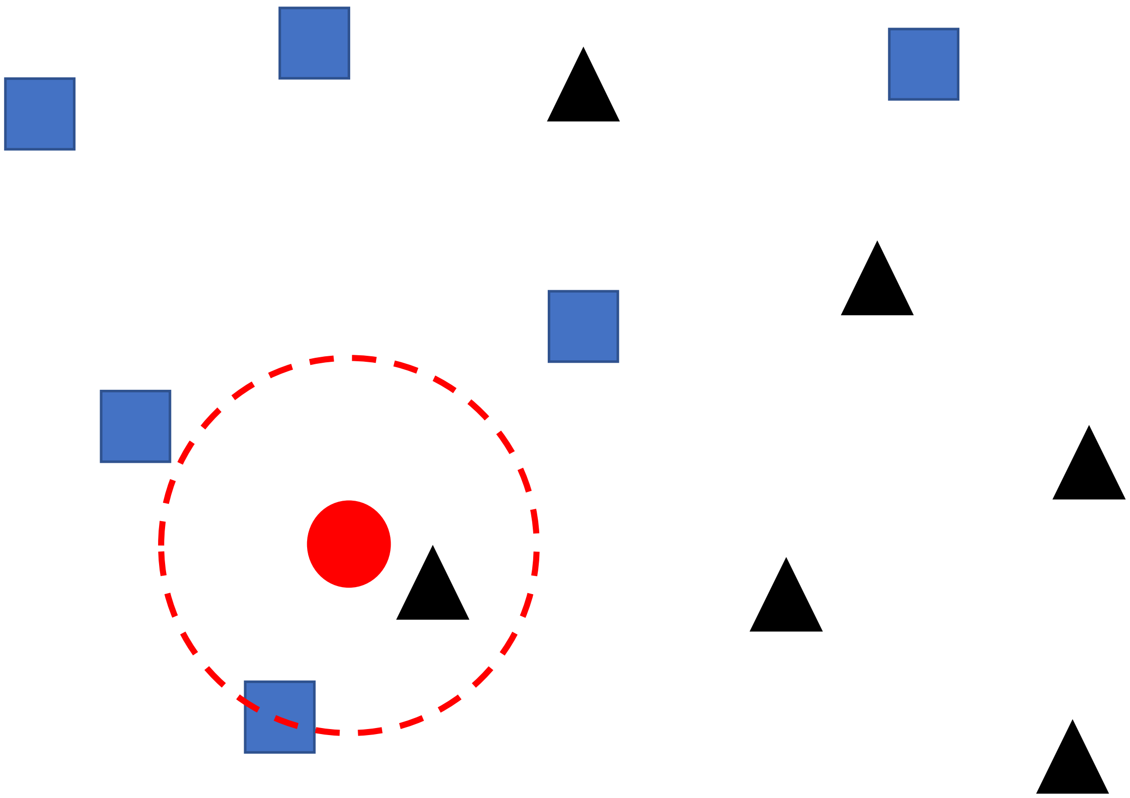
- Which class?

K-Nearest Neighbor



- k=1
- It is classified as a triangle

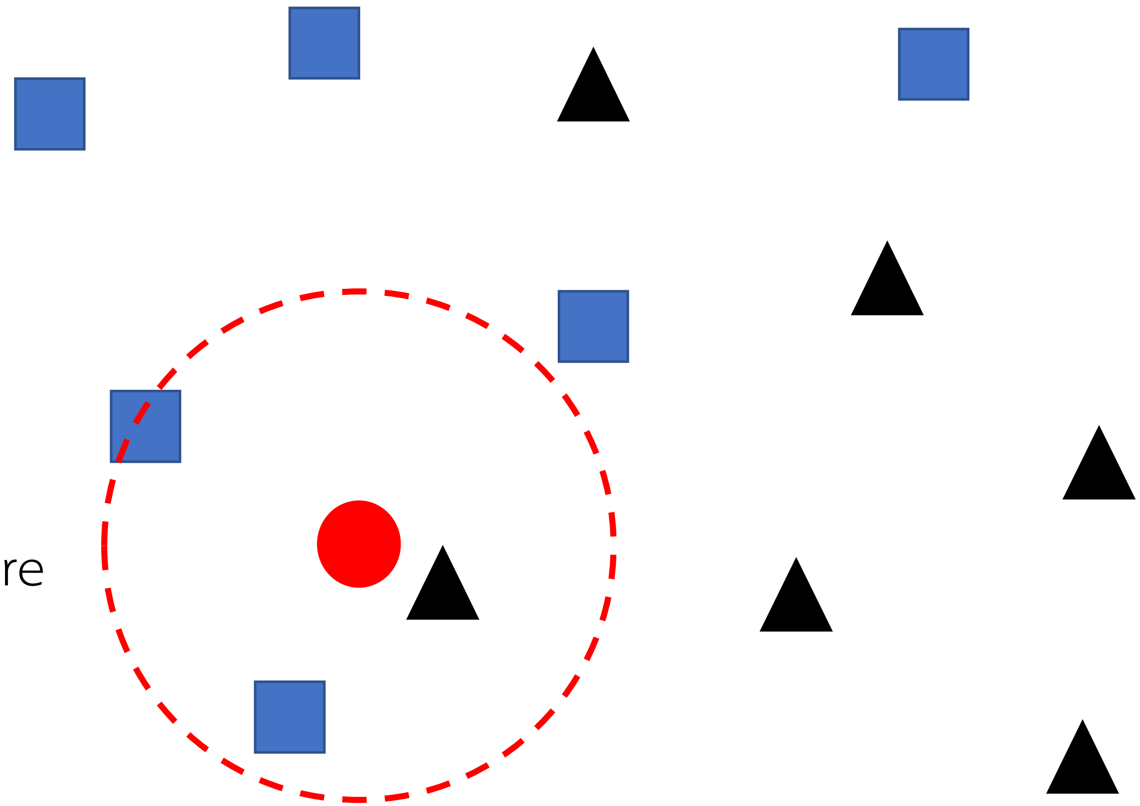
K-Nearest Neighbor



- k=2
- It is classified as ???

K-Nearest Neighbor

- k=3
- It is classified as a square

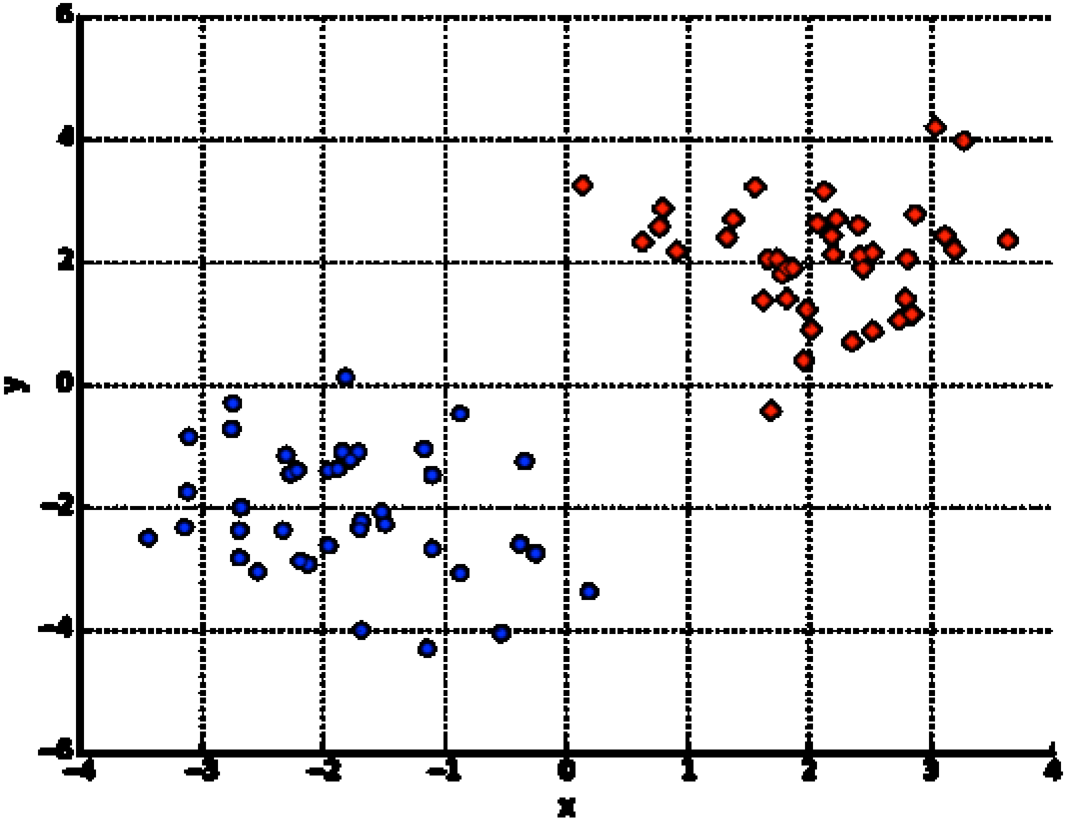


K-Nearest Neighbor

```
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsRegressor
>>> neigh = KNeighborsRegressor(n_neighbors=2)
>>> neigh.fit(X, y)
KNeighborsRegressor(...)
>>> print(neigh.predict([[1.5]]))
[ 0.5]
```

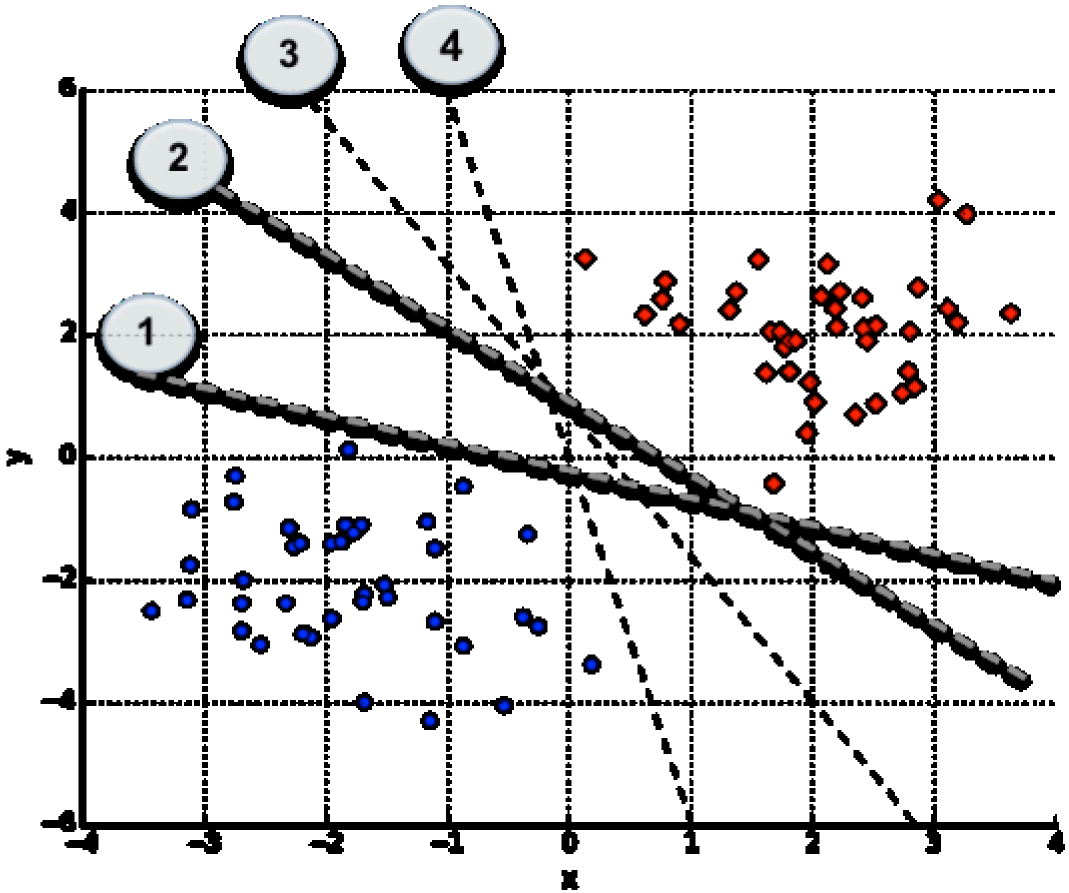
Ref: • (content) http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

Support Vector Machines (SVM)



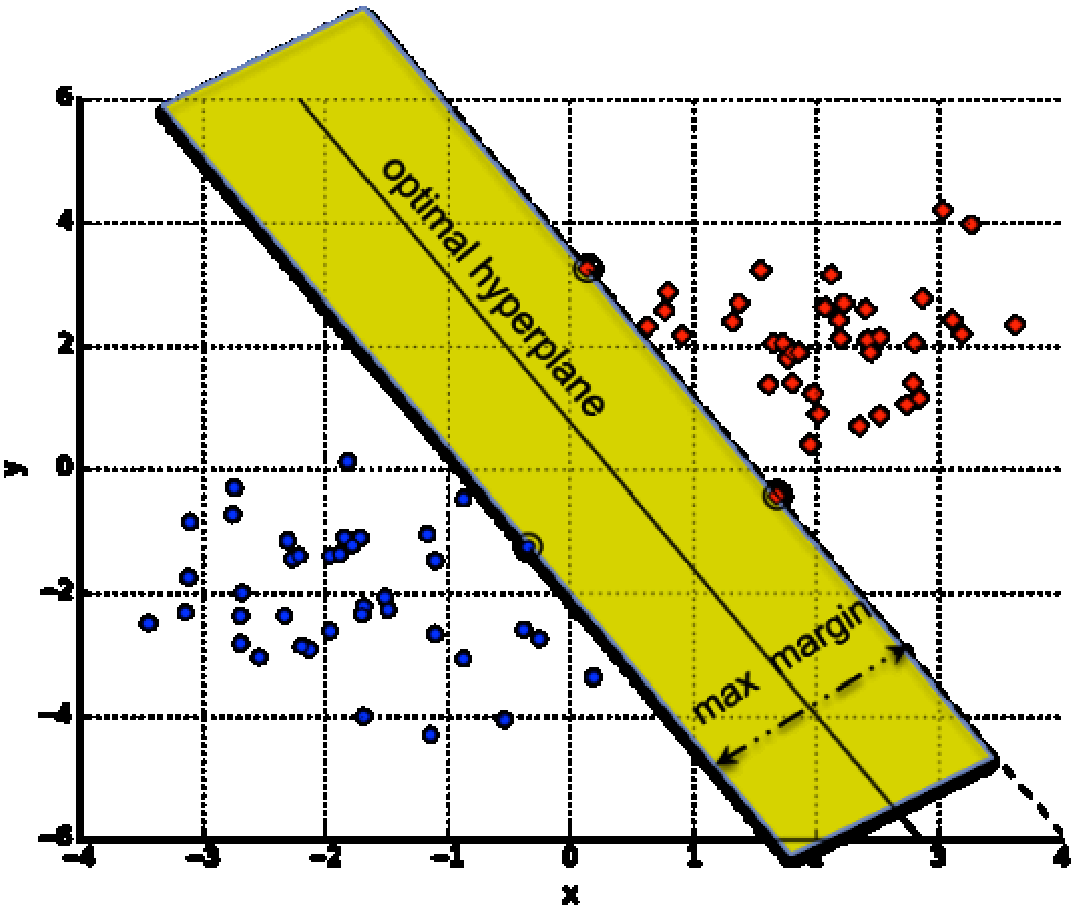
Ref: • (image) http://beta.cambridgespark.com/courses/jpm/05-module.html

Support Vector Machines (SVM)



Ref: • (image) http://beta.cambridgespark.com/courses/jpm/05-module.html

Support Vector Machines (SVM)



Ref: • (image) http://beta.cambridgespark.com/courses/jpm/05-module.html

Support Vector Machines (SVM)

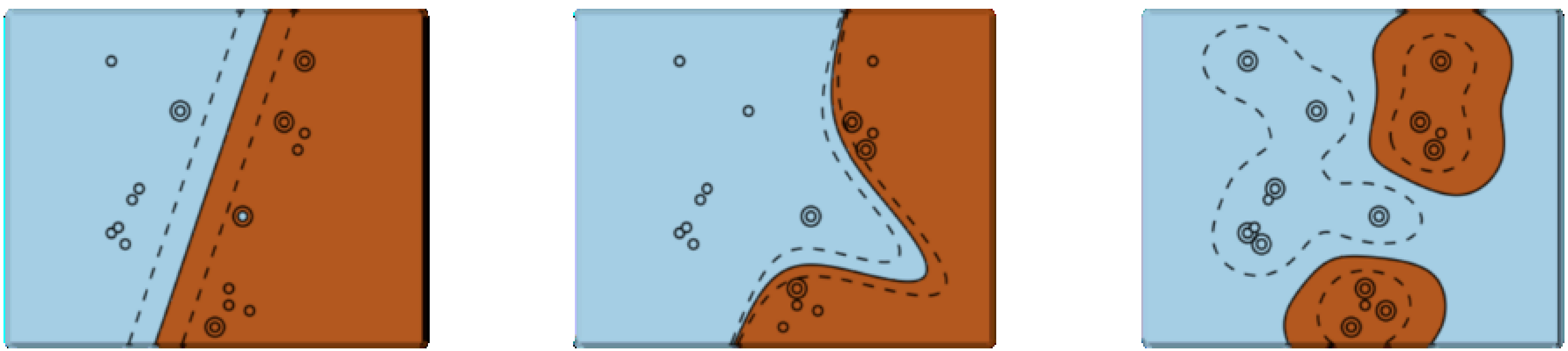
- Kernels

  - **Gaussian Radial Basis Function (RBF)**: $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$
  - **Polynomial**: $K(x_i, x_j) = (\gamma x_i^T x_j + c)^d$
  - **Sigmoid**: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + c)$

Ref: • (image) http://beta.cambridgespark.com/courses/jpm/05-module.html

Support Vector Machines (SVM)

Linear Kernel | **Polynomial Kernel** | **RBF Kernel**



C hyperparameter | *C plus gamma, degree and coefficient hyperparameters* | *C plus gamma hyperparameter*

Support Vector Machines (SVM)

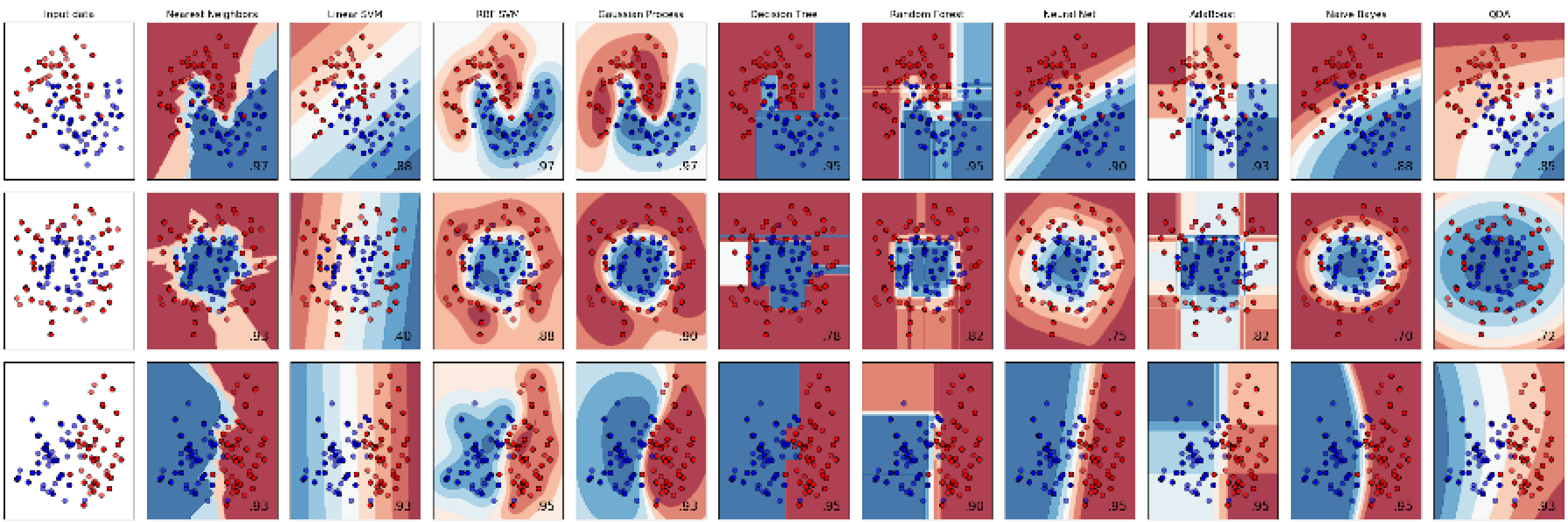
```
>>> from sklearn import svm
>>> X = [[0, 0], [1, 1]]
>>> y = [0, 1]
>>> clf = svm.SVC()
>>> clf.fit(X, y)
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

Ref: • (image) http://scikit-learn.org/stable/modules/svm.html

Python Code

- Go to

http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html



Regularization

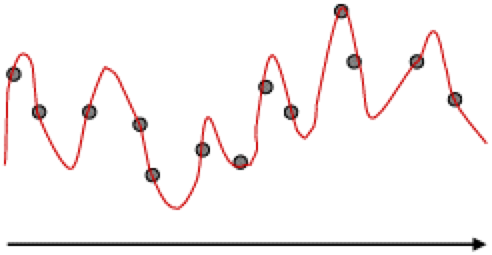
Regularization

– The minimization

$$\min_f |Y_i - f(X_i)|^2$$

may be attained with zero errors.

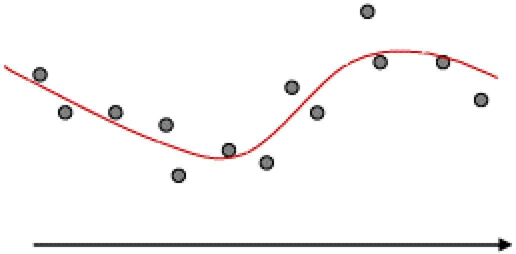
But the function may not be unique.



– Regularization

$$\min_{f \in H} \sum_{i=1}^{n} |Y_i - f(X_i)|^2 + \lambda \|f\|_H^2$$

- Regularization with smoothness penalty is preferred for uniqueness and smoothness.
- Link with some RKHS norm and smoothness



L1 vs. L2

L1 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} |w_i|}$$

L2 regularization on least squares:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \sum_j \left(t(\mathbf{x}_j) - \sum_i w_i h_i(\mathbf{x}_j) \right)^2 + \boxed{\lambda \sum_{i=1}^{k} w_i^2}$$

L2 regularization	L1 regularization
Computational efficient due to having analytical solutions	Computational inefficient on non-sparse cases
Non-sparse outputs	Sparse outputs
No feature selection	Built-in feature selection

Ref: • (image) http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/

In the spirit of science, there really is no such thing as a 'failed experiment.' Any test that yields valid data is a valid test.

Adam Savage

つづく