Master of Science in
Artificial Intelligence
and Data Analytics

University of Macedonia
School of Information Sciences
Department of Applied Informatics

Μηχανική Μάθηση & Επεξεργασία Φυσικής Γλώσσας

ΙΩΑΝΝΗΣ ΡΕΦΑΝΙΔΗΣ – ΝΙΚΟΛΑΟΣ ΣΑΜΑΡΑΣ

Καθηγητής **Καθηγητής**

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΠΛΗΡΟΦΟΡΙΑΣ
Τμήμα Εφαρμοσμένης Πληροφορικής

CMOR
Computational Methodologies
& Operations Research

PCA – [1]

- High--Dimensions = Lot of Features

Document classification Features per document = thousands of words/unigrams

millions of bigrams, contextual information

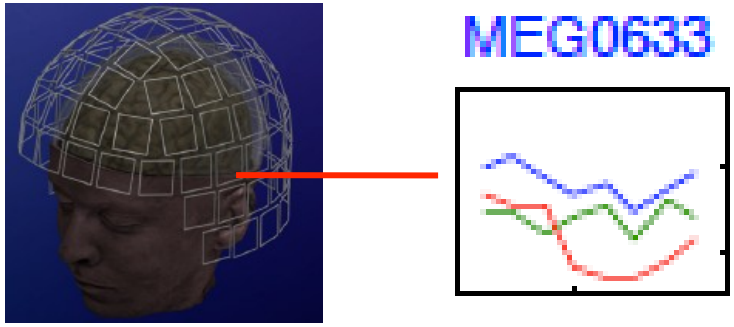
Surveys –Netflix

480189 users x 17770 movies

	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6
Tom	5	?	?	1	3	?
George	?	?	3	1	2	5
Susan	4	3	1	?	5	1
Beth	4	3	?	2	4	2

PCA – [2]

- Big & High--Dimensional Data.
- Useful to learn lower dimensional representations of the data.



PCA – [3]

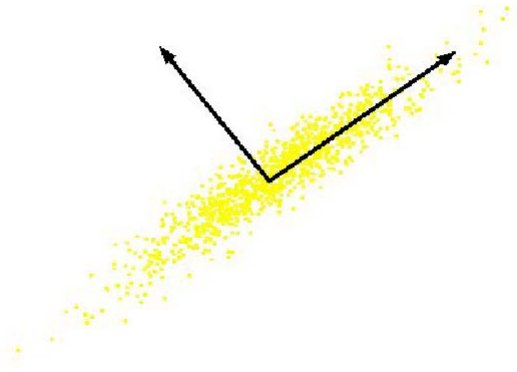
PCA, Kernel PCA, ICA: Powerful unsupervised learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets.

Useful for:

- Visualization
- More efficient use of resources (e.g., time, memory, communication)
- Statistical: fewer dimensions → better generalization
- Noise removal (improving data quality)
- Speeding machine learning algorithms

PCA – [4]

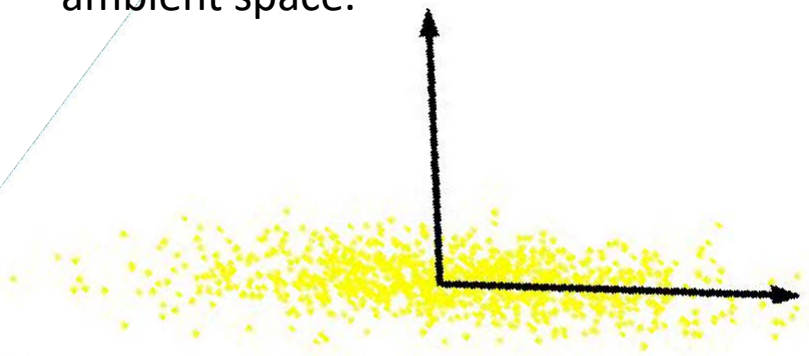
What is PCA: Unsupervised technique for extracting variance structure from high dimensional datasets.



PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.

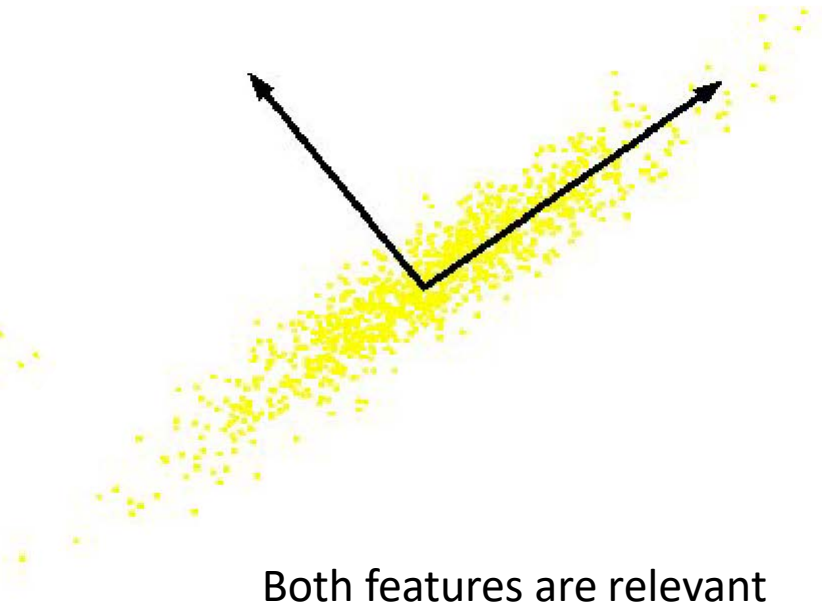
PCA – [5]

Intrinsically lower dimensional than the dimension of the ambient space.



Only one relevant feature

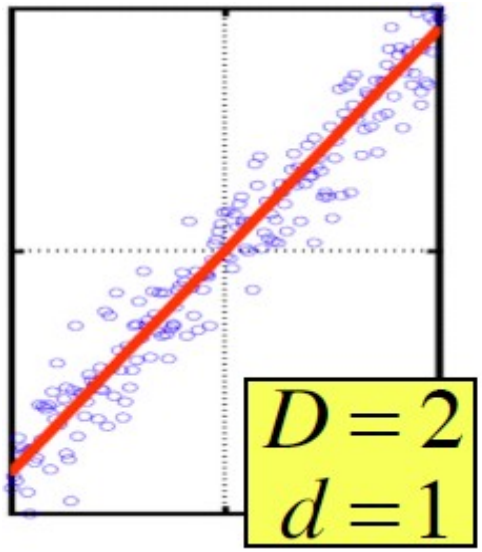
If we rotate data, again only

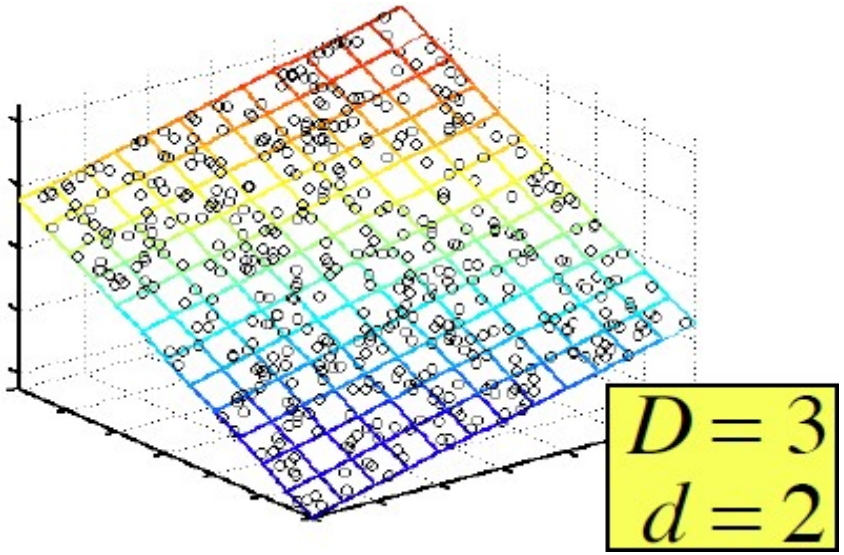


Both features are relevant

Question: Can we transform the features so that we only need to preserve one latent feature?

PCA – [6]





In case where data lies on or near a low d--dimensional linear subspace, axes of this subspace are an effective representation of the data.

Identifying the axes is known as Principal Components Analysis, and can be obtained by using classic matrix computation tools (Eigen or Singular Value Decomposition).

PCA – [7]

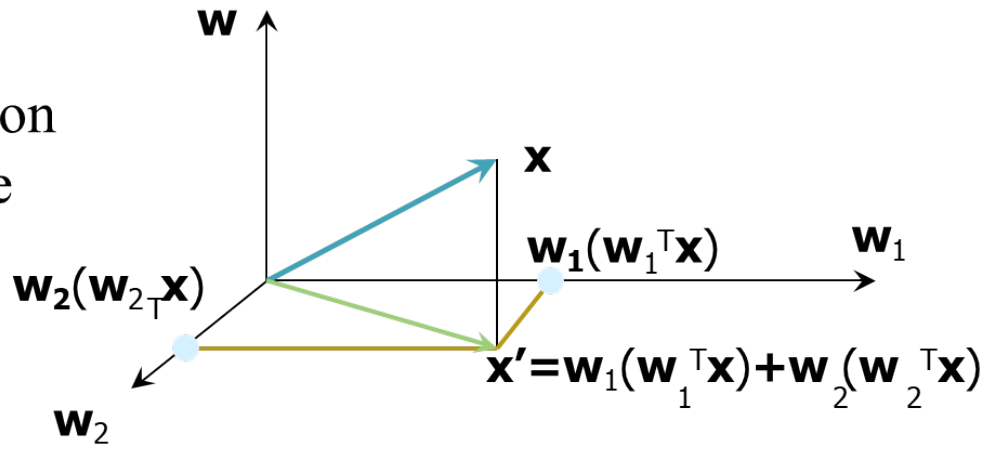
Given the **centered** data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$, compute the principal vectors:

$$\mathbf{w}_1 = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{(\mathbf{w}^T \mathbf{x}_i)^2\}$$ 1st PCA vector

We maximize the variance of projection of $\mathbf{x}$

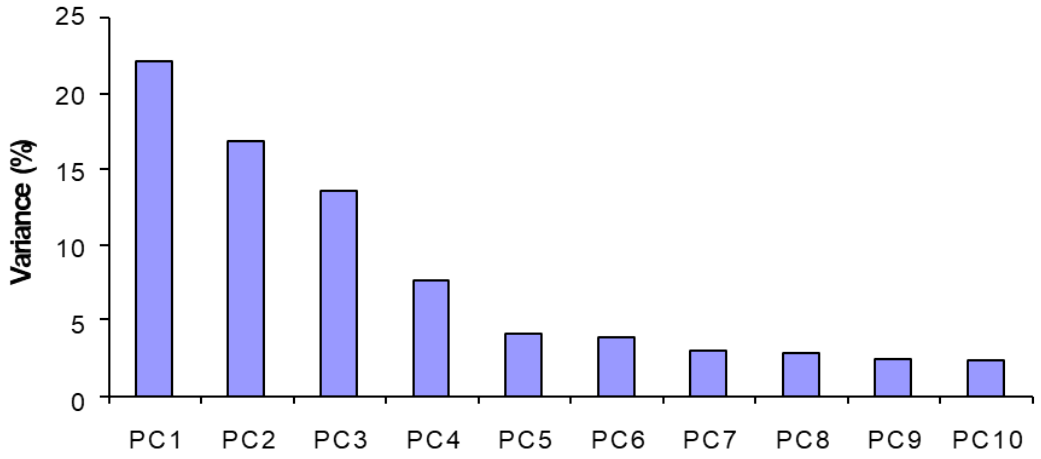
$$\mathbf{w}_k = \arg\max_{\|\mathbf{w}\|=1} \frac{1}{m} \sum_{i=1}^{m} \{[\mathbf{w}^T (\mathbf{x}_i \underbrace{- \sum_{j=1}^{k-1} \mathbf{w}_j \mathbf{w}_j^T \mathbf{x}_i}_{\mathbf{x}' \text{ PCA reconstruction}})]^2\}$$ k^{th} PCA vector

We maximize the variance of the projection in the residual subspace



PCA – [8]

- For n original dimensions, sample covariance matrix is nxn, and has up to n eigenvectors. So n PCs.
- Where does dimensionality reduction come from? Can *ignore* the components of lesser significance.



You do lose some information, but if the eigenvalues are small, you don't lose much

1. n dimensions in original data
2. calculate n eigenvectors and eigenvalues
3. choose only the first p eigenvectors, based on their eigenvalues
4. final data set has only p dimensions

PCA – [9]

The Breast Cancer data set is a real-valued multivariate data that consists of two classes, where each class signifies whether a patient has breast cancer or not. The two categories are: *malignant* and *benign*.
The malignant class has 212 samples, whereas the benign class has 357 samples.
It has 30 features shared across all classes: radius, texture, perimeter, area, smoothness, fractal dimension, etc.

Breast Cancer Data Exploration

```
from sklearn.datasets import load_breast_cancer
```

load_breast_cancer will give you both labels and the data. To fetch the data, you will call `.data` and for fetching the labels `.target`.

The data has 569 samples with thirty features, and each sample has a label associated with it. There are two labels in this dataset.

PCA – [10]

```
breast = load_breast_cancer()
breast_data = breast.data
```

Let's check the shape of the data.

```
breast_data.shape
(569, 30)
```

let's load the labels and check the shape.

```
breast_labels = breast.target
breast_labels.shape
(569,)
```

Now you will import `numpy` since you will be reshaping the `breast_labels` to concatenate it with the `breast_data` so that you can finally create a `DataFrame` which will have both the data and labels.

```
import numpy as np
labels = np.reshape(breast_labels,(569,1))
```

PCA – [11]

After reshaping the labels, you will concatenate the data and labels along the second axis, which means the final shape of the array will be 569 x 31.

```
final_breast_data =
np.concatenate([breast_data,labels],axis=1)
final_breast_data.shape
(569, 31)
```

Now you will import pandas to create the DataFrame of the final data to represent the data in a tabular fashion.

```
import pandas as pd
breast_dataset = pd.DataFrame(final_breast_data)
```

PCA – [12]

Let's quickly print the features that are there in the breast cancer dataset!

```
features = breast.feature_names
features
array(['mean radius', 'mean texture', 'mean perimeter',
'mean area', 'mean smoothness', 'mean compactness', 'mean
concavity', 'mean concave points', 'mean symmetry', 'mean
fractal dimension', 'radius error', 'texture error',
'perimeter error', 'area error', 'smoothness error',
'compactness error', 'concavity error', 'concave points
error', 'symmetry error', 'fractal dimension error',
'worst radius', 'worst texture', 'worst perimeter',
'worst area', 'worst smoothness', 'worst compactness',
'worst concavity', 'worst concave points', 'worst
symmetry', 'worst fractal dimension'], dtype='<U23')
```

PCA – [13]

If you note in the features array, the `label` field is missing. Hence, you will have to manually add it to the `features` array since you will be equating this array with the column names of your `breast_dataset dataframe`.

```
features_labels = np.append(features,'label')
```

Great! Now you will embed the column names to the `breast_dataset dataframe`.

```
breast_dataset.columns = features_labels
```

Let's print the first few rows of the dataframe.

```
breast_dataset.head()
```

PCA – [14]

Since the original labels are in `0, 1` format, you will change the `labels` to benign and malignant using `.replace` function. You will use `inplace=True` which will modify the `dataframe breast_dataset`.

```
breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1,
'Malignant',inplace=True)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	label
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	26.40	166.10	2027.0	0.14100	0.21130	0.4107	0.2216	0.2060	0.07115	Benign
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	38.25	155.00	1731.0	0.11660	0.19220	0.3215	0.1628	0.2572	0.06637	Benign
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	34.12	126.70	1124.0	0.11390	0.30940	0.3403	0.1418	0.2218	0.07820	Benign
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	39.42	184.60	1821.0	0.16500	0.86810	0.9387	0.2650	0.4087	0.12400	Benign

PCA – [15]

Visualizing the Breast Cancer data

You start by **`Standardizing`** the data since PCA's output is influenced based on the scale of the features of the data.

It is a common practice to normalize your data before feeding it to any machine learning algorithm.

To apply normalization, you will import `StandardScaler` module from the `sklearn` library and select only the `features` from the `breast_dataset` you created in the Data Exploration step. Once you have the `features`, you will then apply scaling by doing `fit_transform` on the `feature` data.

While applying `StandardScaler`, each feature of your data should be normally distributed such that it will scale the distribution to a mean of zero and a standard deviation of one.

PCA – [16]

```
from sklearn.preprocessing import StandardScaler
x = breast_dataset.loc[:, features].values
x = StandardScaler().fit_transform(x) # normalizing the
features x.shape (569, 30)
```

Let's check whether the normalized data has a mean of zero and a standard deviation of one.

```
np.mean(x),np.std(x)
(-6.826538293184326e-17, 1.0)
```

Let's convert the normalized features into a tabular format with the help of DataFrame.

```
feat_cols = ['feature'+str(i) for i in range(x.shape[1])]
normalised_breast = pd.DataFrame(x,columns=feat_cols)
normalised_breast.tail()
```

PCA – [17]

Now comes the critical part, the next few lines of code will be projecting the thirty-dimensional Breast Cancer data to two-dimensional **principal components**.
You will use the `sklearn` library to import the PCA module, and in the PCA method, you will pass the number of components (n_components=2) and finally call fit_transform on the aggregate data. Here, several components represent the lower dimension in which you will project your higher dimension data.

```
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=2)
principalComponents_breast = pca_breast.fit_transform(x)
```

Next, let's create a DataFrame that will have the principal component values for all 569 samples.

```
principal_breast_Df = pd.DataFrame(data =
principalComponents_breast , columns = ['principal
component 1', 'principal component 2'])
principal_breast_Df.tail()
```

PCA – [18]

ID	principal component 1	principal component 2
564	6.439315	-3.576817
565	3.793382	-3.584048
566	1.256179	-1.902297
567	10.374794	1.672010
568	-5.475243	-0.670637

Once you have the principal components, you can find the **explained_variance_ratio**. It will provide you with the amount of information or variance each principal component holds after projecting the data to a lower dimensional subspace.

PCA – [19]

```
print('Explained variation per principal component:
{}'.format(pca_breast.explained_variance_ratio_))
Explained variation per principal component:

[0.44272026 0.18971182]
```

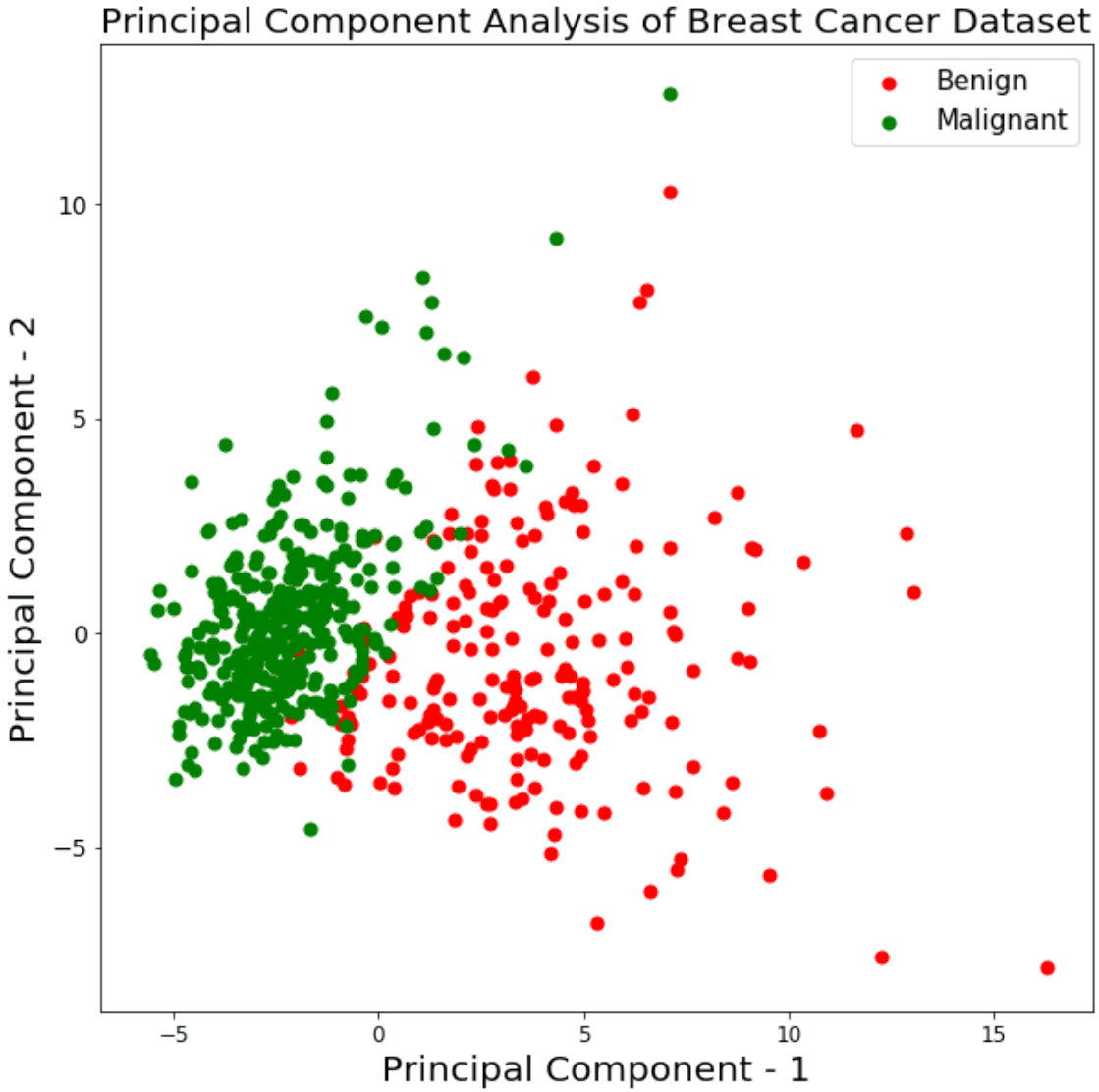
From the above output, you can observe that the principal component-1 holds 44.2% of the information while the principal component-2 holds only 19% of the information. Also, the other point to note is that while projecting thirty-dimensional data to a two-dimensional data, 36.8% information was lost.

Let's plot the visualization of the 569 samples along the principal component-1 and principal component-2 axis. It should give you good insight into how your samples are distributed among the two classes.

PCA – [20]

```
plt.figure() plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Principal Component - 1',fontsize=20)
plt.ylabel('Principal Component - 2',fontsize=20)
plt.title("Principal Component Analysis of Breast Cancer
Dataset",fontsize=20) targets = ['Benign', 'Malignant']
colors = ['r', 'g']
for target, color in zip(targets,colors):
      indicesToKeep = breast_dataset['label'] == target
      plt.scatter(principal_breast_Df.loc[indicesToKeep,
      'principal component 1'],
      principal_breast_Df.loc[indicesToKeep,
      'principal component 2'],c = color, s = 50)
plt.legend(targets,prop={'size': 15})
```

PCA – [21]


Principal Component Analysis of Breast Cancer Dataset
Benign
Malignant
Principal Component - 2
Principal Component - 1
10
5
0
−5
−5
0
5
10
15

Rule Coverage and Accuracy

- Coverage of a rule:
  - Fraction of records that satisfy the antecedent of a rule
- Accuracy of a rule:
  - Fraction of records that satisfy both the antecedent and consequent of a rule

Tid	Refund	Marital Status	Taxable Income	Class
1	Yes	**Single**	125K	**No**
2	No	Married	100K	**No**
3	No	**Single**	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	**Single**	85K	**Yes**
9	No	Married	75K	**No**
10	No	**Single**	90K	**Yes**

(Status=Single) → No

Coverage = 40%, Accuracy = 50%

Evaluating Performance – [1]

Performance evaluation is the most critical of all the steps in the machine learning process.

Three general questions must be answered.

1. Will the benefits received from a machine learning project more than offset the cost of the machine learning process?

2. How do we interpret the results of a machine learning session?

3. Can we use the results of a machine learning process with confidence?

Evaluating Performance – [2]

Any answer to the question

1. Will the benefits received from a machine learning project more than offset the cost of the machine learning process?

requires deep knowledge about

- ❑ Business model
- ❑ State of available data
- ❑ Current resources

Evaluating Performance – [3]

- How to obtain a reliable estimate of performance?

- Performance of a model may depend on other factors besides the learning algorithm:

  – Class distribution
  – Cost of misclassification
  – Size of training and test sets

Evaluating Performance – [4]

Classification Correctness

Test set model accuracy can be summarized in a *confusion matrix*.

Suppose that the output attribute has three possible classes C_1, C_2 and C_3.

Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

A generic confusion matrix has the following form.

	C1	C2	C3
C1	c_{11}	c_{12}	c_{13}
C2	c_{21}	c_{22}	c_{23}
C3	c_{31}	c_{32}	c_{33}

Confusion Matrix at Wikipedia

Evaluating Performance – [5]

	C1	C2	C3
C1	c_{11}	c_{12}	c_{13}
C2	c_{21}	c_{22}	c_{23}
C3	c_{31}	c_{32}	c_{33}

Rule 1. Values along the main diagonal represent correct classification. The value c_{11} represents the total number of class C_1 instances correctly classified by the model.

Accuracy of Model = $$\frac{\sum_{i=1}^{n} c_{ii}}{\#\ \text{Instances}} \in [0 \quad 1]$$

Where *n* the number of possible classes

Evaluating Performance – [6]

	C1	C2	C3
C1	c_{11}	c_{12}	c_{13}
C2	c_{21}	c_{22}	c_{23}
C3	c_{31}	c_{32}	c_{33}

Rule 2. Values in row C_i represent those instances that belong actually to class C_i, i=1,…,n.

$$\#\ \text{Instances}(C_i) = \sum_{i=1}^{n} c_{ij}, \quad j = 1,\ldots,n$$

$$\#\ \text{Incorrect Instances}(C_i) = \sum_{i=1}^{n} c_{ij}, \quad i \neq j,\ j = 1,\ldots,n$$

Evaluating Performance – [7]

	C1	C2	C3
C1	c_{11}	c_{12}	c_{13}
C2	c_{21}	c_{22}	c_{23}
C3	c_{31}	c_{32}	c_{33}

Rule 3. Values found in column C_j indicate those instances that have been classified as members of class C_j, j=1,…,n.

$$\#\ \text{Instances}(C_j) = \sum_{j=1}^{n} c_{ij}, \quad i = 1,\ldots,n$$

$$\#\ \text{Incorrect Instances}(C_j) = \sum_{j=1}^{n} c_{ij}, \quad i \neq j,\ i = 1,\ldots,n$$

Total Incorrect Instances =
Incorrect Instances(C_i) + # Incorrect Instances(C_j)

Evaluating Performance – [8]

Two-class Error analysis.

	Computed Accept	Computed Reject
Accept	*True Accept*	*False Reject*
Reject	*False Accept*	*True Reject*

where

True Accept and *True Reject* represent correctly classified test set instances.

False Accept denotes accepted applicants that should have been rejected.

False Reject denotes rejected applicants that should have been accepted.

Evaluating Performance – [9]

Table 1 represent the test set error of two supervised learner models built for the credit card application problem.

Table 1.

Model A	Computed Accept	Computed Reject	**Model B**	Computed Accept	Computed Reject
Accept	600	25	Accept	600	75
Reject	75	300	Reject	25	300

Which model is better? Why?

Model B

Evaluating Performance – [10]

Computing Test Set Confidence Intervals

$$\text{Classifier Error Rate}(E) = \frac{\#\ \text{of test set errors}}{\#\ \text{test set instances}}$$

The Error Rate gives an indication as to the likely future performance of the model.

How confident can we be that this error rate is a valid measure of actual model performance?

To answer this question we must determine a confidence interval for a computed error rate.

Evaluating Performance – [11]

A procedure for computing confidence interval.

Step 1. Given a test set sample S of size n and error rate E.

Step 2. Compute the sample variance

$$Var(E)=E(1-E)$$

Step 3. Compute the standard error

$$SE = \sqrt{Var(E)/n}$$

Step 4. Calculate an upper (lower) bound for the 95% confidence interval as

$$E \pm 2(SE)$$

Evaluating Performance – [12]

An Illustrative example

Input: S=random sample
n=100, E=10% or 0.10

Var(E)=Var(0.10)=0.10(1 – 0.10)=0.09

SE=0.03

Lower bound → 0.10 – 2(0.03)=0.04
Upper bound → 0.10 + 2(0.03)=0.16

We can be 95% confident that the actual test set error rate falls between

[0.04 0.16] → [84% 96%] (Accuracy)

Evaluating Performance – [13]

What happened if we increase the number of test set instances?

Let n=1000, E=10% or 0.10

Var(E)=Var(0.10)=0.10(1 – 0.10)=0.09

SE=0.0095

Lower bound → 0.10 – 2(0.0095)=0.081
Upper bound → 0.10 + 2(0.0095)=0.119

We can be 95% confident that the actual test set error rate falls between [0.081 0.119] → [88% 92%]

Evaluating Performance – [14]

General Comments

- The confidence interval is valid iff the test data has been randomly chosen from the pool of all possible test set instances.

- Test, training and validation data must represent disjoints sets.

- If possible, the instances in each class should be distributed in the training, validation and test data as they are seen in the entire dataset.

Evaluating Performance – [15]

If an appropriate test set is not available, we can apply a method known as cross-validation. Using this method all available data are partitioned into n fixed-size units. N-1 of them are used for training whereas the n^{th} is the test set.

This process is repeated until each of the fixed-size units has been used as test data.

Model test set correctness is computed as the average accuracy realized from n training-testing trials.

Evaluating Performance – [16]

Comparing Supervised Learner models with Categorical Output.

Comparison of two supervised learner models constructed with the same training data.

Three possible test set scenarios are:

1. The accuracy of the models is compared using two independent test sets randomly selected from a pool of sample data.
2. The same test set data is employed to compare the models. The comparison is based on a pairwise, instance-by-instance computation.
3. The same test set data is used to compare the overall classification correctness of the models.

Evaluating Performance – [17]

Hypothesis Testing

H_0: There is no significant difference in the test set error rate of two supervised learner models M_1 and M_2 built with the same training data.

H_1: There is significant difference in the test set error rate of two supervised learner models M_1 and M_2 built with the same training data.

Evaluating Performance – [18]

Comparing models with independent test data

Step 1. (Initialization). Given

- Two models, M_1 and M_2 built with the same training data.
- Two independent test sets, set A with card(A)=n_1 and set B with card(B)=n_2.
- Error rate E_1 and variance v_1 for model M_1 on test set A
- Error rate E_2 and variance v_2 for model M_2 on test set B

Step 2. Compute

$$P = \frac{|E_1 - E_2|}{\sqrt{v_1 / n_1 + v_2 / n_2}}$$

Step 3. Conclude

If P≥2, hypothesis H_1 holds. Otherwise, H_0 is true.

Evaluating Performance – [19]

Comments

$v_1=E_1(1-E_1)$, $v_2=E_2(1-E_2)$

Instead of v_1 and v_2 we can compute

$$q=\frac{E_1+E_2}{2}$$

and

$$v_q=q(1-q)$$

Hence, the computation of Step 2 is equal to

$$P=\frac{|E_1-E_2|}{\sqrt{v_q(1/n_1+1/n_2)}}$$

Evaluating Performance – [20]

An illustrative example

We test M_1 model on test set A with card(A)=100=n_1, and M_2 model on test set B with card(B)=100=n_2.

M_1 achieves an 80% classification accuracy with set A and M_2 achieves an 70% classification accuracy with set B.

We wish to know if model M_1 has performed significantly better than model M_2

Evaluating Performance – [21]

The computations are:

E_1=0.20, E_2=0.30, v_1=0.16, v_2=0.21

$$P = \frac{|0.20 - 0.30|}{\sqrt{\frac{0.16}{100} + \frac{0.21}{100}}} = \frac{0.10}{\sqrt{0.0037}} = \frac{0.10}{0.0608} \simeq 1.6447$$

Because P≈1.6447<2 there is no significant difference between models M_1 and M_2.

Evaluating Performance – [22]

If we use the average of the two error rates instead of v_1 and v_2 we take

q=(0.20+0.30)/2=0.25, v_q=0.1875

$$P = \frac{0.10}{\sqrt{0.1875\left(\frac{2}{100}\right)}} \simeq 1.633 < 2, \text{ Hence, } H_0 \text{ is true}$$

Comment. We can increase our confidence in the result by switching the two test sets and repeating the experiment. This is especially important if a significant difference is seen with the initial test set selection. The average of the two values for P is then used for the significant test.

Evaluating Performance – [23]

Pairwise comparison with a single test set

Step 1. (Initialization). Given

- Two models, M_1 and M_2 built with the same training data.
- One test set A with card(A)=n.
- Error rate E_1 for model M_1 on test set A
- Error rate E_2 for model M_2 on test set A

Step 2. Compute

$$V_{12} = \frac{1}{n-1}\sum_{i=1}^{n}[(e_{1i} - e_{2i}) - (E_1 - E_2)]^2$$

where

e_{1i} is the classifier error on the i^{th} instance for M_1

e_{2i} is the classifier error on the i^{th} instance for M_2

Evaluating Performance – [24]

Step 2. Compute (continue)

$$P = \frac{|E_1 - E_2|}{\sqrt{V_{12}/n}}$$

Step 3. Conclude

If P≥2, hypothesis H_1 holds. Otherwise, H_0 is true.

Evaluating Performance – [25]

Comparing models with a single test set

Step 1. (Initialization). Given

- Two models, M_1 and M_2 built with the same training data.
- One test set A with card(A)=n.
- Error rate E_1 for model M_1 on test set A
- Error rate E_2 for model M_2 on test set A

Step 2. Compute

$$P = \frac{|E_1 - E_2|}{\sqrt{(v_1 + v_2)/n}}$$

Step 3. Conclude

If P≥2, hypothesis H_1 holds. Otherwise, H_0 is true.

Evaluating Performance – [26]

Attribute Evaluation

A correlation coefficient (ρ) measures the degree of linear association between two numeric attributes.

$\rho \in [-1 \;\; 1]$

A positive correlation means two attributes increase or decrease in unison.

A negative correlation means that one attribute increases and the other decreases.

If ρ is very close to zero, the two attributes are not linearly correlated.

Evaluating Performance – [27]

Computing Attribute Correlations with Excel

In the Excel formula bar type

=CORREL(range1;range2)

Making Scatterplot Diagrams

A scatterplot diagram offers a two-dimensional plot of corresponding pairs of attribute values.

Cardiology Data Set

Evaluating Performance – [28]

Comparing Supervised Learner models with Numeric Output.

Comparison of two supervised learner models constructed with the same training data.

Three possible test set scenarios are:

1. The accuracy of the models is compared using two independent test sets randomly selected from a pool of sample data.
2. The same test set data is employed to compare the models. The comparison is based on a pairwise, instance-by-instance computation.
3. The same test set data is used to compare the overall classification correctness of the model.

Evaluating Performance – [29]

General Background

$$\text{Mean Absolute Error,} \quad mae = \frac{1}{n}\sum_{i=1}^{n} | a_i - c_i |$$

$$\text{Mean Squared Error,} \quad mse = \frac{1}{n}\sum_{i=1}^{n} (a_i - c_i)^2$$

$$\text{Root Mean Squared Error,} \quad rmse = \sqrt{mse}$$

where for the i^{th} instance

a_i = actual output value

c_i = computed output value

Evaluating Performance – [30]

Comparing models with independent test data

Step 1. (Initialization). Given

- Two models, M_1 and M_2 built with the same training data.
- Two independent test sets, set A with card(A)=n_1 and set B with card(B)=n_2.
- Mean Absolute Error mae_1 and variance v_1 for model M_1 on test set A
- Mean Absolute Error mae_2 and variance v_2 for model M_2 on test set B

Step 2. Compute

$$P = \frac{|\, mae_1 - mae_2 \,|}{\sqrt{v_1 / n_1 + v_2 / n_2}}$$

Step 3. Conclude

If P≥2, hypothesis H_1 holds. Otherwise, H_0 is true.

Evaluating Performance – [31]

Pairwise comparison with a single test set

Step 1. (Initialization). Given

- Two models, M_1 and M_2 built with the same training data.
- One test set A with card(A)=n.
- Mean Absolute Error mae_1 for model M_1 on test set A
- Mean Absolute Error mae_2 for model M_2 on test set A

Step 2. Compute

$$V_{12} \quad \text{and} \quad P = \frac{|\, mae_1 - mae_2 \,|}{\sqrt{V_{12}/n}}$$

Step 3. Conclude

If P≥2, hypothesis H_1 holds. Otherwise, H_0 is true.

Evaluating Performance – [32]

Comparing models with a single test set

Step 1. (Initialization). Given

- Two models, M_1 and M_2 built with the same training data.
- One test set A with card(A)=n.
- Mean Absolute Error mae_1 for model M_1 on test set A
- Mean Absolute Error mae_2 for model M_2 on test set A

Step 2. Compute

$$P = \frac{|\, mae_1 - mae_2 \,|}{\sqrt{(v_1 + v_2)/n}}$$

Step 3. Conclude

If P≥2, hypothesis H_1 holds. Otherwise, H_0 is true.

Evaluating Performance – [33]

Unsupervised Model Evaluation

More difficult task than supervised evaluation. Why?

All unsupervised clustering techniques compute some measure of cluster quality. A common technique is to calculate the summation of squared error differences between the instances of each cluster and their corresponding cluster center.

Smaller values for sums of squared error differences indicate clusters of high quality.

Evaluating Performance – [34]

A procedure for Unsupervised Model Evaluation

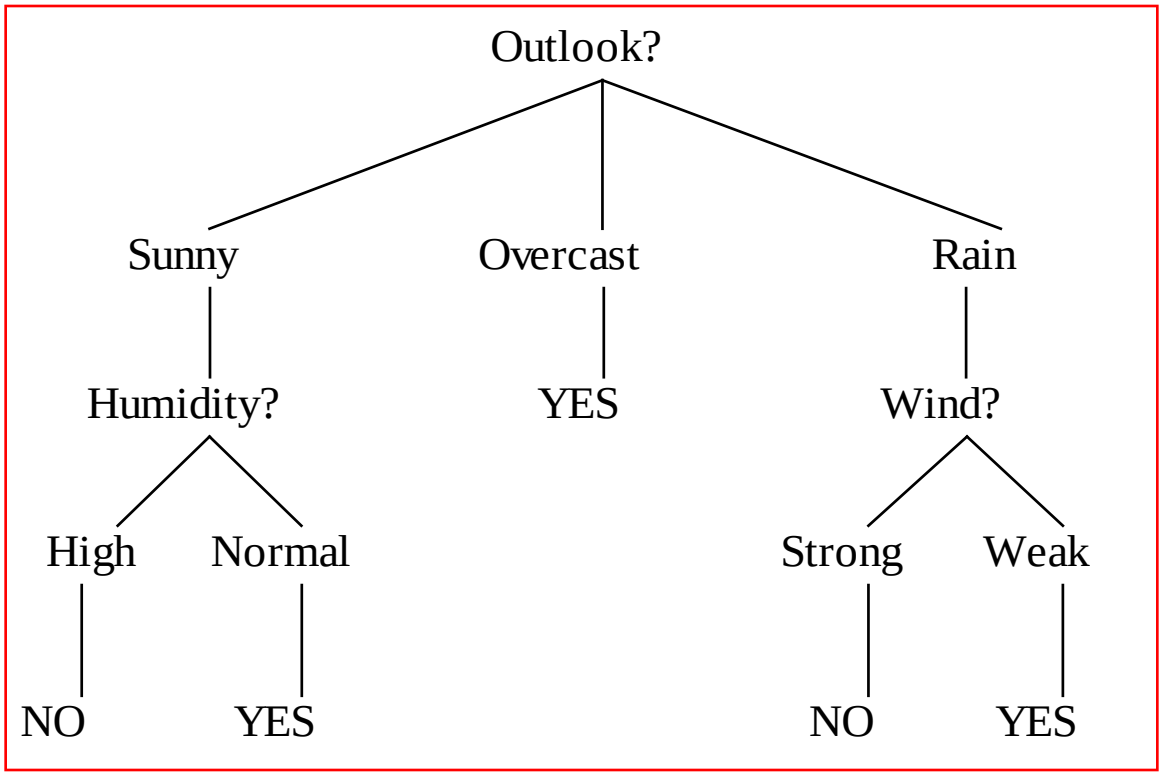
Step 1. Perform an unsupervised clustering. Designate each cluster as a class and assign each cluster an arbitrary name.

Step 2. Choose a random sample of instances from each of the classes formed as a result of the instance clustering. Each class should be represented in the random sample in the same ratio as it is represented in the entire dataset.

Step 3. Build a supervised learner model with class name as the output attribute using the randomly sampled instances as training data. Use the remaining instances to test the supervised model for classification correctness.

Decision Trees – [1]

Definition. A Decision Tree (D.T.) is a simple structure where non-terminal nodes represents tests on one or more input attributes and terminal nodes reflect decision outcomes.



Is today a good day to play tennis?

Decision Trees – [2]

Advantages

1. Easy to understand

2. Can be translated easily into production rules

3. Work well experimentally

Requirements

1. Value of output attribute is a logical combination of input attributes (no sums, logs, weights, etc.)

2. If necessary continuous variables can be converted to discrete and vice versa.

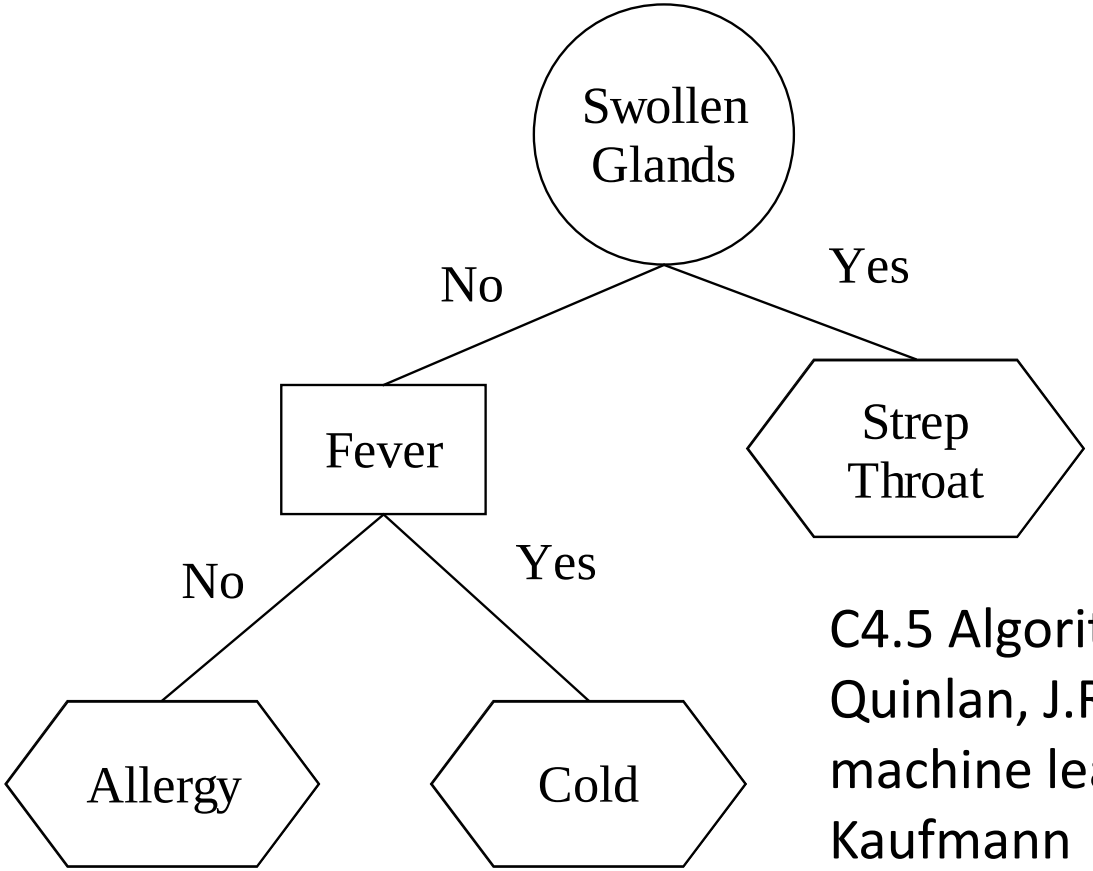
Decision Trees – [3]

Table 1. Hypothetical training data for disease diagnosis

Patient ID	Sore Throat	Fever	Swollen Glands	Congestion	Headache	Diagnosis
1	Yes	Yes	Yes	Yes	Yes	Strep
2	No	No	No	Yes	Yes	Allergy
3	Yes	Yes	No	Yes	No	Cold
4	Yes	No	Yes	No	No	Strep
5	No	Yes	No	Yes	No	Cold
6	No	No	No	Yes	No	Allergy
7	Yes	No	Yes	No	No	Strep
8	No	No	No	Yes	Yes	Allergy
9	Yes	Yes	No	Yes	Yes	Cold
10	No	Yes	No	Yes	Yes	Cold

Decision Trees – [4]

The D.T. generalizes the data in **Table 1.**



C4.5 Algorithm
Quinlan, J.R. (1993). Programs for machine learning. CA: Morgan Kaufmann

Decision Trees – [5]

The D.T. (created by data of **Table 1**) tell us that we can diagnose a patient in this dataset. The D.T. has generalized the data and provided us with attribute relationships important for an accurate diagnosis.

- If a patient has *Swollen Glands*, the diagnosis is *Strep Throat*
- If a patient doesn't have *Swollen Glands* and has a *Fever*, the diagnosis is a *Cold*
- If a patient doesn't have *Swollen Glands* and a *Fever*, the diagnosis is an *Allergy*

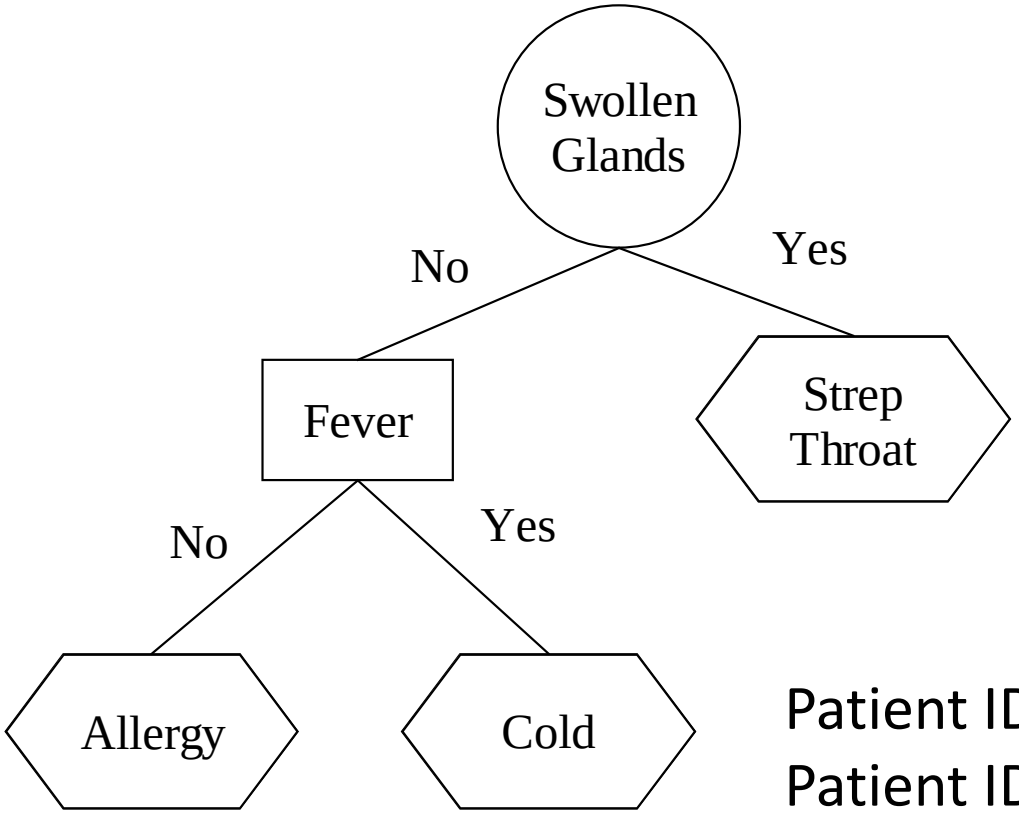
Decision Trees – [6]

The instances used to create the D.T. model are known as *Training Data*.

How well the produced model is able to be of general use????
In order to test the accuracy of the model we use a *test set*. The instances of the test set have a known classification.

Patient ID	Sore Throat	Fever	Swollen Glands	Congestion	Headache	Diagnosis
11	No	No	Yes	Yes	Yes	??
12	Yes	Yes	No	No	Yes	??
13	No	No	No	No	Yes	??

Decision Trees – [7]



Patient ID: 11 → Strep Throat
Patient ID: 12 → Cold
Patient ID: 13 → Allergy

Decision Trees – [8]

Any D.T. can be translated into a set of production rules. In general, production rules are rules of the form

IF *antecedent conditions*
THEN *consequent conditions*

Where
Antecedent conditions detail values or value ranges for input attributes

Consequent conditions specify the values or value ranges for the output attribute

Decision Trees – [9]

Mapping a D.T. to a set of production rules.

Start from a root node

FOR each path of the decision tree

Put the value(s) of input attributes in antecedent conditions

IF more than one input attributes exist **THEN**

join them using logical AND operator

Put the value at the terminal or leaf node in consequent conditions