Data Preparation

Fátima Rodrigues
mfc@isep.ipp.pt
Departamento de Engenharia Informática (DEI/ISEP)

Motivation

In real applications the data tend to be inconsistent, incomplete and/or wrong

- What happens when the data is not correct?
- Can the knowledge extracted from the data be trusted?
- Obstacles to knowledge discovery: **poor data**

GIGO law: **Garbage In, Garbage Out**

Data Preparation

- Set of steps that may be necessary to carry out before any further analysis takes place on the available data
- It is estimated that data preparation takes 70-80% of all development effort of a data mining project
- Good data preparation is key to produce valid and reliable models

Data Preparation - Goals

- Understanding the nature of the data
- Solve problems inherent to the data
- Adapting the data according to the Data Mining algorithms
- Provide more meaningful data analysis and extract knowledge with meaning
- Know what useful information exists in a particular data set, so when random samples are formed from it, the information is preserved

Major Tasks in Data Preparation

Data Selection

- Creates the appropriate set of data to explore

Data Cleaning

- Handling missing values, smooth noisy data, identify or remove outliers and resolve inconsistencies

Data Transformation

- Normalization, data conversion

Data Reduction

- Obtains reduced representation in volume but produces the same or similar analytical results

Data Selection

Sampling

- Sampling is the main technique employed for data selection
  - It is often used for both the preliminary investigation of the data and the final data analysis

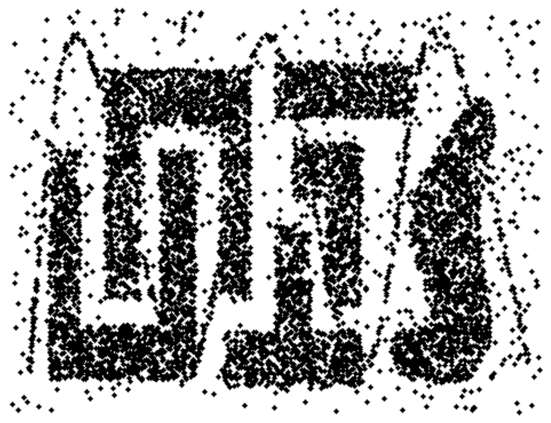
- Sampling is used in data mining because processing the entire set of data of interest is too expensive or time consuming

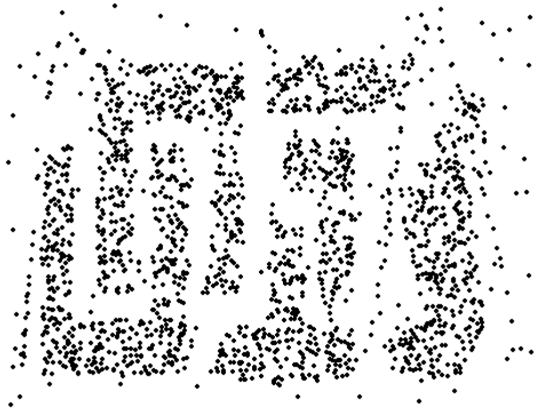
Sampling …

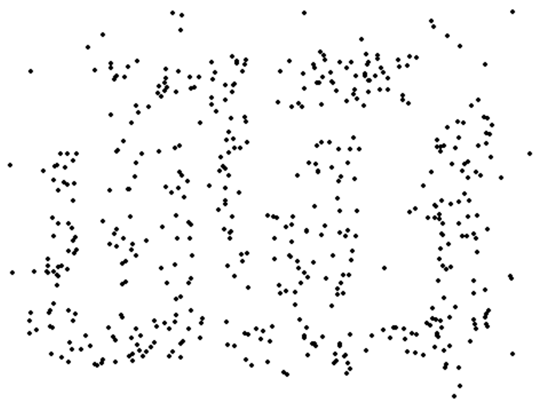
The key principle for effective sampling is the following:

– using a sample will work almost as well as using the entire data set, **if the sample is representative**

– a sample is representative if it has approximately the same properties (of interest) as the original set of data


8000 points


2000 Points

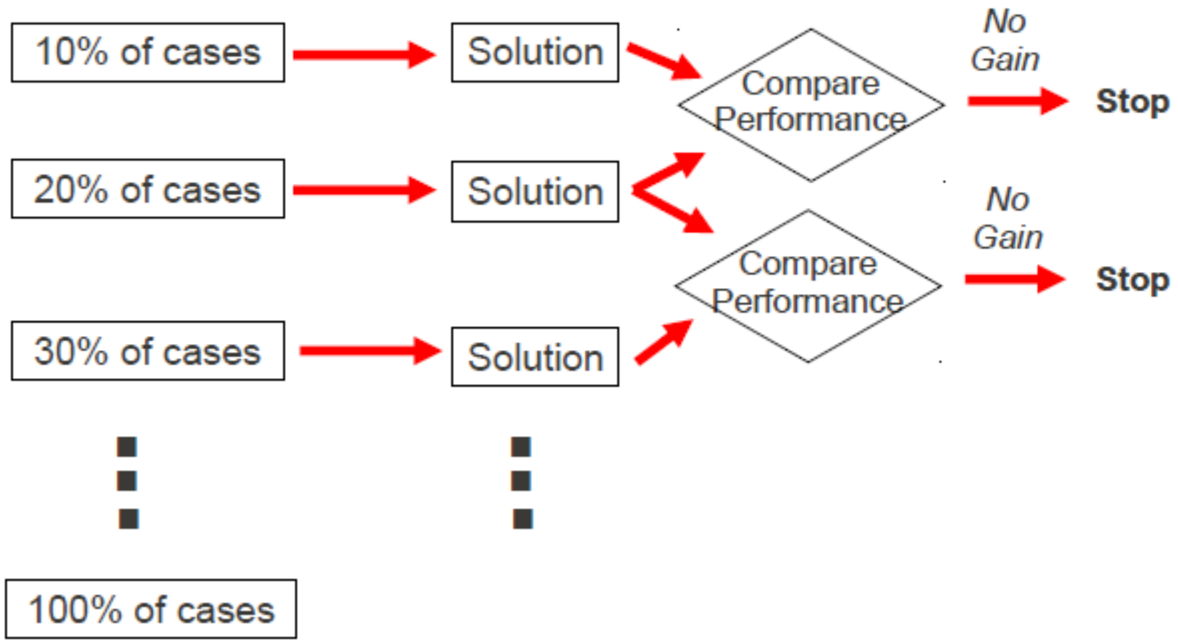

500 Points

Types of Sampling

- **Simple random sampling**
  - There is an equal probability of selecting any particular item
- **Sampling without replacement**
  - As each item is selected, it is removed from the population
- **Sampling with replacement**
  - Objects are not removed from the population as they are selected for the sample
  - The same object can be picked up more than once
- **Stratified sampling**
  - Split the data into several partitions; then draw random samples from each partition
- **Incremental sampling**

Incremental Sampling (cases)

The training is performed with random samples with an increasing number of cases



Stopping criteria:

- The error don't decreased
- The complexity of the model has increased more than the fall in the error rate
- The complexity of the current solution is acceptable

Incremental Sampling (attributes)

- **Direct Selection**
  - It starts with an empty set of attributes and iteratively select one attribute at a time, until not get any improvement in the model
  - It is more effective and more efficient in sets with few attributes

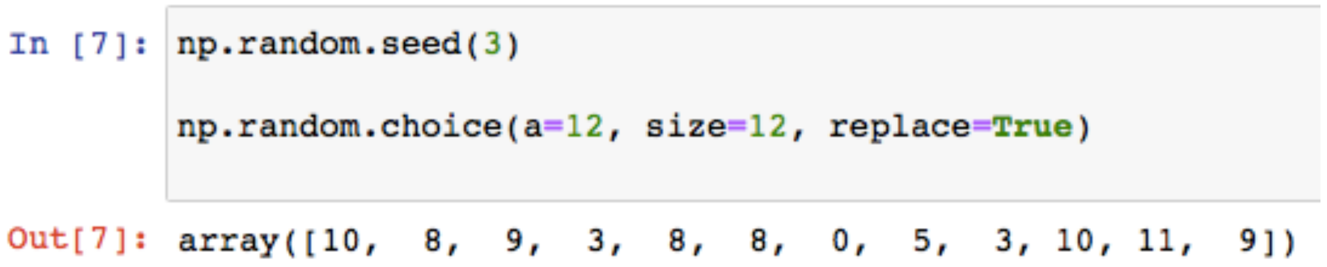
- **Reverse Elimination**
  - It starts with the set of all attributes and iteratively removes one attribute at a time, until no improvement is reached
  - It is better to discern interactions between attributes
  - best to deal with samples with a high number of attributes
  - More computationally expensive

R: Sampling

```
#Disjoint sets
set.seed(1234)      # random seed to a fixed value to be able to reproduce results

ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.65, 0.35))

 > ind
 [1] 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1 1 2 2 1 1 1 1 1
 [46] 1 1 1 1 2 1 1 2 1 1 1 1 2 1 2 2 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 2
 [91] 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 1 2 1 1 2 2 1 1 2 2 2 2 2 1 1 1 1 1 1 2 1 1 1 2
 [136] 1 2 1 1 2 1 2 1 1 1 1 2 1 2 1


 trainData <- iris[ind==1,]
 testData <- iris[ind==2,]


 #Non disjoint sets
 # take two random samples from iris dataset sample without replacement
 trainSample <- iris[sample(1:nrow(iris), 100,replace=FALSE),]
 testSample <- iris[sample(1:nrow(iris), 50,replace=FALSE),]
```

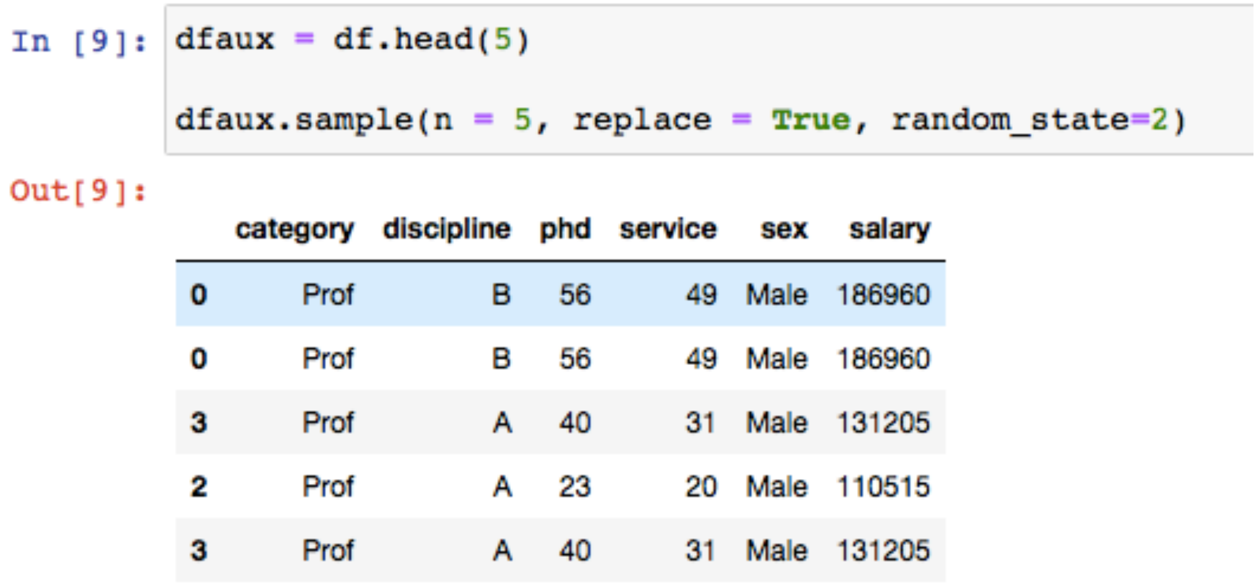
Sampling with replacement

Using NumPy



```
In [7]: np.random.seed(3)

        np.random.choice(a=12, size=12, replace=True)

Out[7]: array([10,  8,  9,  3,  8,  8,  0,  5,  3, 10, 11,  9])
```

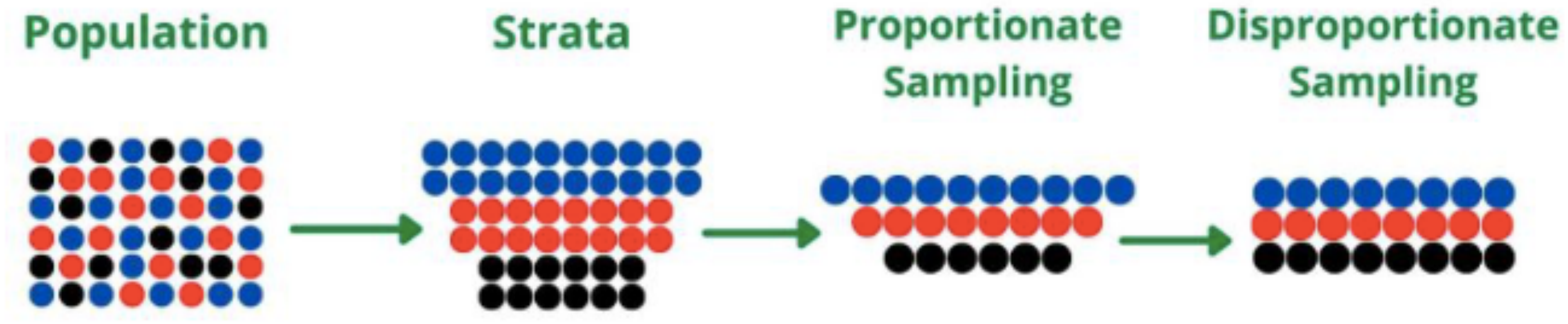
Using pandas



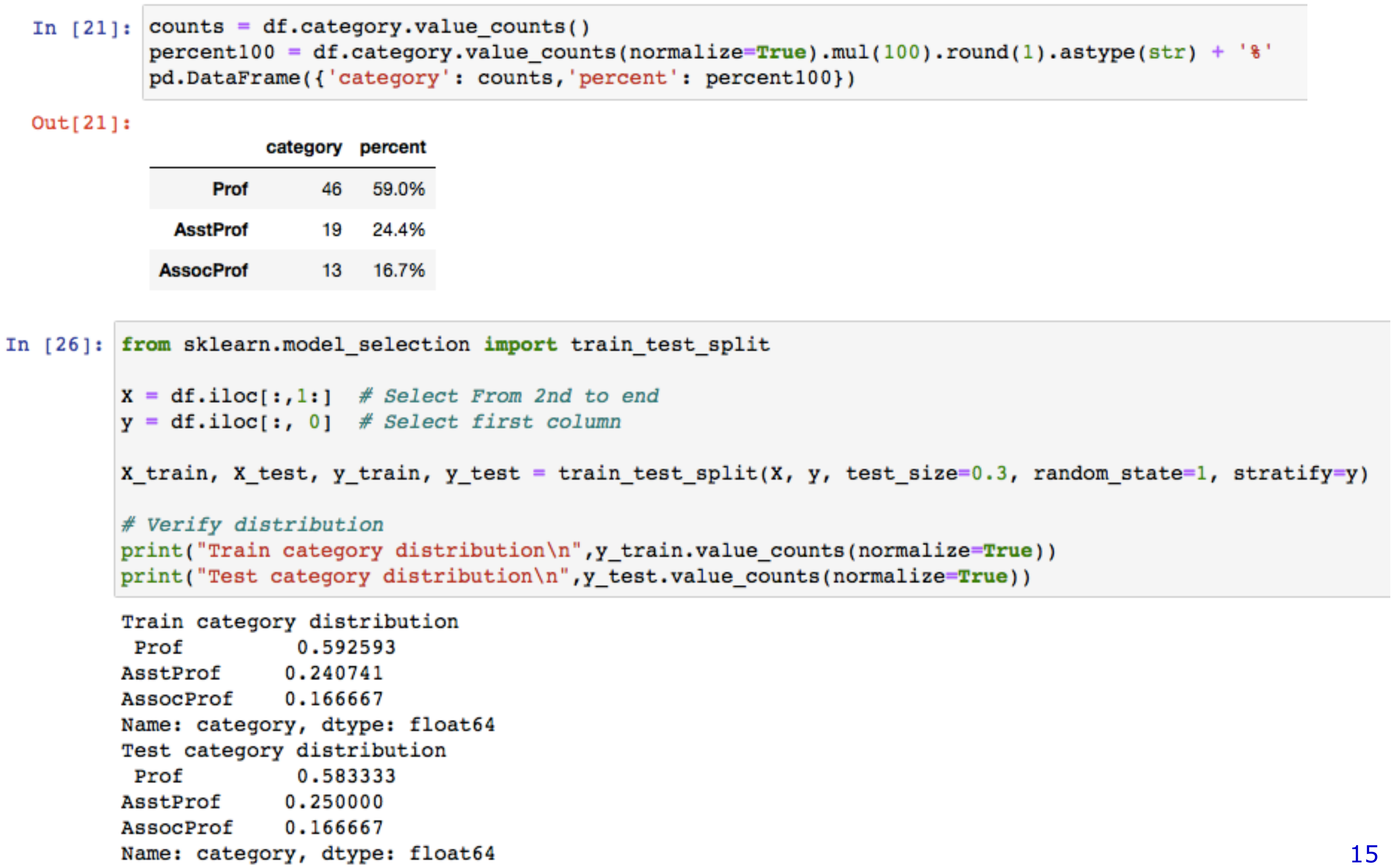
```
In [9]: dfaux = df.head(5)

        dfaux.sample(n = 5, replace = True, random_state=2)

Out[9]:
```

	category	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
0	Prof	B	56	49	Male	186960
3	Prof	A	40	31	Male	131205
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205

Stratified random sampling



Stratified random sampling



```
In [21]: counts = df.category.value_counts()
         percent100 = df.category.value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
         pd.DataFrame({'category': counts,'percent': percent100})
```

Out[21]:

	category	percent
Prof	46	59.0%
AsstProf	19	24.4%
AssocProf	13	16.7%

```
In [26]: from sklearn.model_selection import train_test_split

         X = df.iloc[:,1:]  # Select From 2nd to end
         y = df.iloc[:, 0]  # Select first column

         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)

         # Verify distribution
         print("Train category distribution\n",y_train.value_counts(normalize=True))
         print("Test category distribution\n",y_test.value_counts(normalize=True))
```

```
Train category distribution
 Prof          0.592593
AsstProf      0.240741
AssocProf     0.166667
Name: category, dtype: float64
Test category distribution
 Prof          0.583333
AsstProf      0.250000
AssocProf     0.166667
Name: category, dtype: float64
```

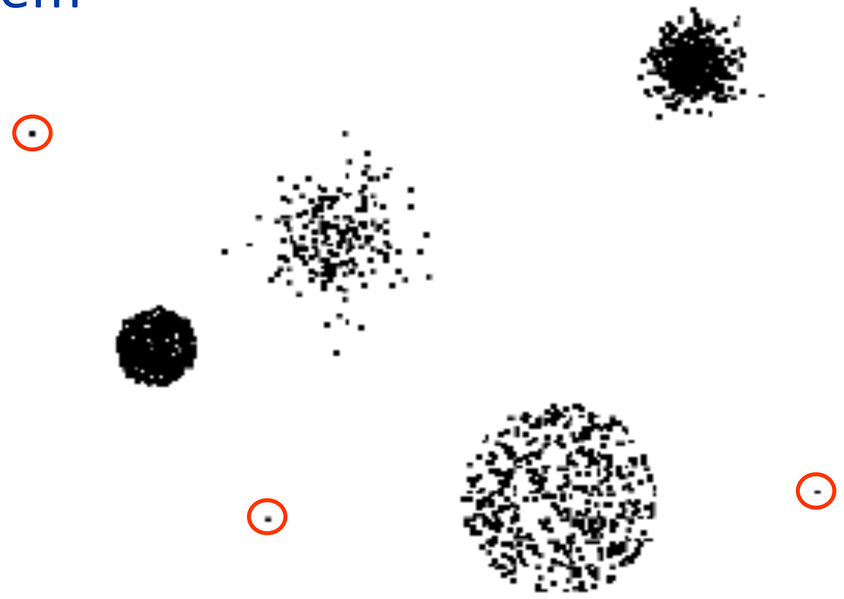
Data Cleaning

Missing Values

- Missing data can appear in several forms:
  - <empty field> “0” “.” “999” “NA” …
- Standardize missing value code(s)
- How can we dealing with missing values?
- Dealing with missing values:
  - ignore records with missing values
  - treat missing value as a separate value
  - Imputation: fill in with mean or median values

Outliers

- Outliers are values thought to be out of range
- Approaches:
  - do nothing
  - enforce upper and lower bounds
  - let binning handle the problem

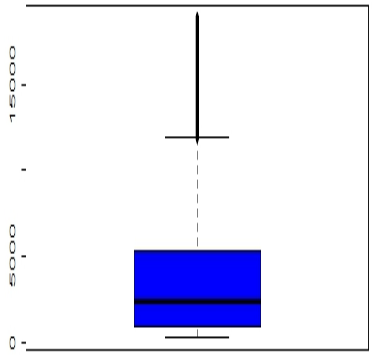


Outliers Identification

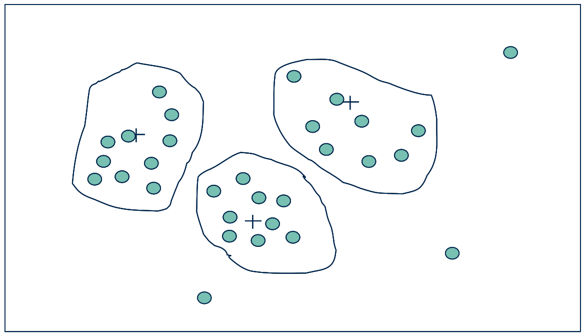
- **Graphical analysis with Boxplot**

  This method points as extreme values any values out of range:

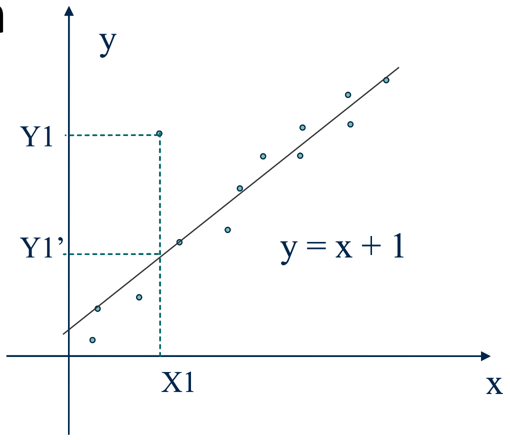
  [Q1-1.5 × IQR, ..., Q3 + 1.5 × IQR]



- **Clustering**



- **Regression**



Field Selection

- Remove fields with no or little variability. Examine the number of distinct field values
  - *Rule of thumb: remove a field where almost all values are the same*

- *Remove Key fields*, where all the values are distinct, as they haven't any semantic associated

False Predictors or Information “Leakers”

- False predictors are fields correlated to target behavior, which describe events that happen at the same time or *after* the target behavior

Example:

The student’s final grade is a false predictor for a student’s likelihood of passing a course

False Predictors: Find “suspects”

- Build an initial decision-tree model
- Consider very strongly predictive fields as “suspects”
  - strongly predictive – if a field by itself provides close to 100% accuracy, at the top or a branch below
- Verify “suspects” using domain knowledge or with a domain expert
- Remove false predictors and build an initial model

Data Transformation

Conversion: Nominal to Numeric

- Some methods can deal with nominal values, other methods (neural nets, regression, nearest neighbor) require only numeric inputs
- To use nominal fields in such methods need to convert them to a numeric value

- Convert nominal ordered attributes (e.g. Grade) to numbers preserving *natural* order
  - to be able to use “>” and “<“ comparisons on these fields
- If no sequential order exists, it should be careful not create it, for ex.
  - male=00, female=11 and not (male=0, female=1)

Data normalization (data scaling)

- Helps prevent attributes with large ranges outweigh ones with small ranges, example:
  - income has range 3000-200000
  - age has range 10-80

- Normalization: scaled to fall within a small, specified range
  - min-max normalization
  - z-score normalization
  - normalization by decimal scaling
  - …

Min-Max Normalization

The Min-Max normalization performs a linear transformation from the original dataset to a new specific dataset (typically 0-1):

- the old minimum ($\min_1$) is mapped to a new minimum: $\min_2$
- the old maximum ($\max_1$) is mapped to a new maximum: $\max_2$

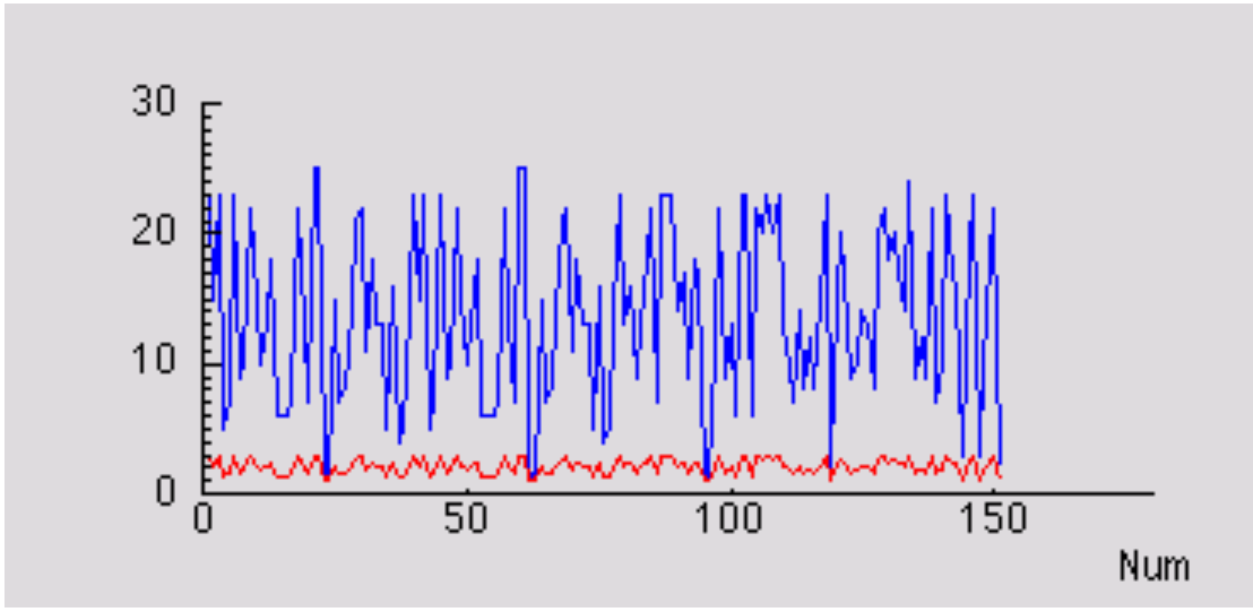
All points between these extremes are mapped to the new scale

The mathematical formula for the Min-Max normalization :

$$y' = \frac{y - \min_y}{\max_y - \min_y}$$

Advantages of the Min-Max normalization

- Preserves exactly all initial relationships of data values
- Doesn't introduce any changes in the data - the form of the histogram is maintained



- doesn't work well in samples with **isolated values**

Zscore Normalization

Also referred as medium-zero normalization or uni-variant normalization, transforms data so that:

- the average is zero
- the standard deviation is one

The formula applied is as follows:

$$x' = \frac{x - \mu}{\sigma}$$

The zscore normalization works well when:

- the sample has isolated values that dominate the normalization Min-max

Sigmoidal Normalization

Transforms the non-linear input data into the range [-1,1] using the sigmoid function

The formula applied by this type of normalization is as follows:

$$y' = \frac{1 - e^{-\alpha}}{1 + e^{-\alpha}} \qquad \alpha = \frac{1 - \mu}{\sigma}$$

The sigmoid normalization is appropriate to:

- Include isolated points in the data set to be analyzed
- Prevent the most common values to be compressed, without losing the ability to represent outliers

Data Reduction

Reducing the dimension of the data set

- Some data mining methods may be unable to handle very large data sets
- The computation time to obtain a certain model may be too large for the application
- We may want simpler models

Some strategies

- Reduce the number of variables
- Reduce the number of cases
- Reduce the number of values on the variables

Discretization

The goal of discretization is to reduce the number of values a continuous attribute assumes by grouping them into a number, n, of intervals (bins)

Some supervised learning methods have their computational complexity heavily dependent on the number of values of the variables. Discretization is a preprocessing technique that may help on these situations

Unsupervised Discretization Algorithms

- Equal-with groups
- Equal-frequency groups

Supervised Discretization Algorithms

- One-level Decision Tree
- k-means method
- …

Discretization

Any discretization process consists of two steps:

- 1st, the number of discrete intervals needs to be decided

Often it is done by the user, although a few discretization algorithms are able to do it on their own

- 2nd, the width (boundary) of each interval must be determined

Often it is done by a discretization algorithm itself

Discretization - Problems

Deciding the number of discretization intervals:

- large number – more of the original information is retained
- small number – the new feature is “easier” for subsequently used learning algorithms

- Computational complexity of discretization should be low since this is only a preprocessing step

Heuristics for guessing the number of intervals

1. Use the number of intervals that is greater than the number of classes to recognize

2. Use the rule of thumb formula:

$$n_{F_i} = M / (3*C)$$

where:

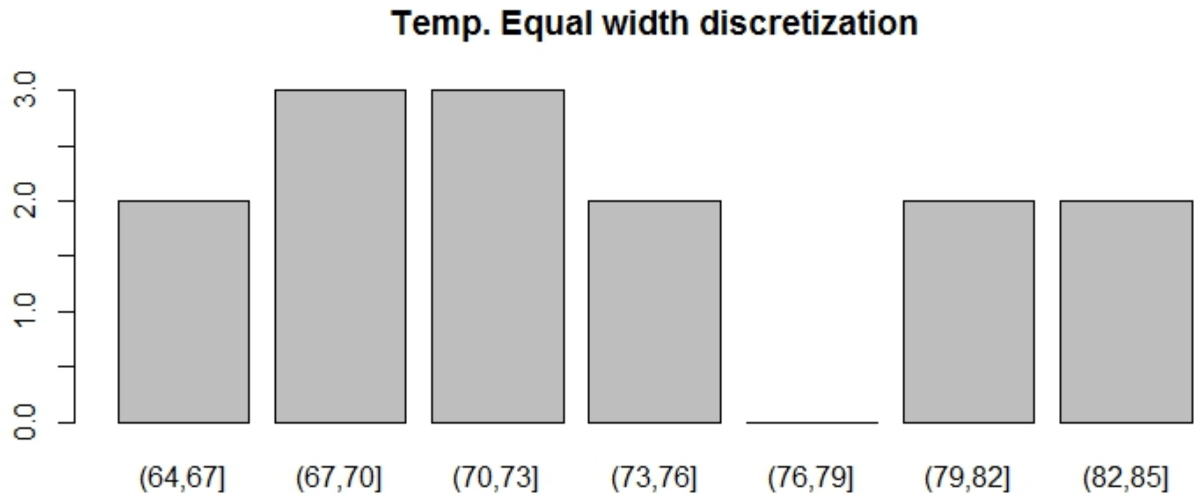
M – number of training examples/instances

C – number of classes

F_i – ith attribute

Discretization: Equal-width

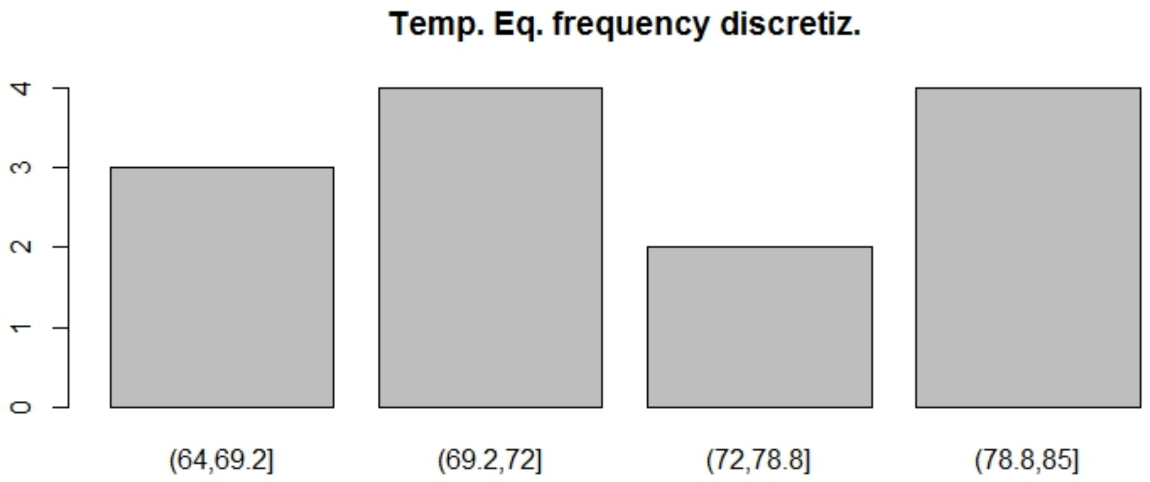
```
df.plot.box(by=‘Interval', column='tempdiscr')
```



May produce clumping

Discretization: Equal-frequency

```
df['Tempdiscr'].value_counts()

(64,67] (67,70] (70,73] (73,76] (76,79] (79,82] (82,85]
      2       3       3       2       0       2       2


df.plot.box(by='Interval', column='tempdiscr',
                            tittle="Temp. Eq. frequency discretiz.")
```



Discretization: Equal-frequency advantages

- Generally preferred because avoids clumping
- In practice, “almost-equal” height binning is used which avoids clumping and gives more intuitive breakpoints

- Additional considerations:
  - don’t split frequent values across bins
  - create separate bins for special values (e.g. 0)
  - readable breakpoints (e.g. round breakpoints)

Discretization considerations

- Equal Width is simplest, good for many classes
  - can fail miserably for unequal distributions
- Equal frequency gives better results
- Class-dependent can be better for classification
  - Note: decision trees build discretization on the fly
  - Naïve Bayes requires initial discretization
- Many other methods exist …

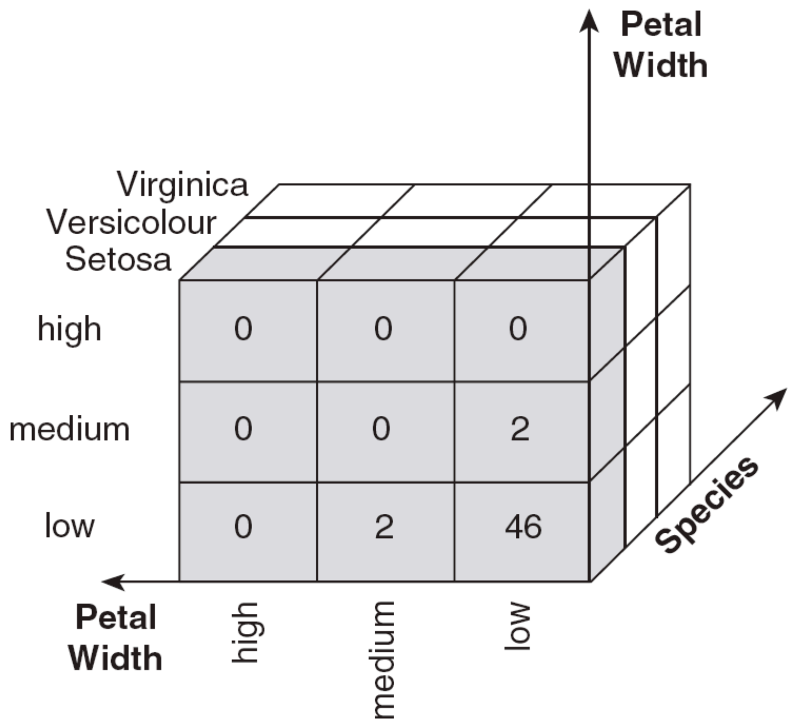
Example: Iris dataset

The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant: setosa, virginica, versicolour, and four attributes: sepal width, sepal length petal width, petal length

sepal length	sepal width	petal length	petal width	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
6.3	2.9	5.6	1.8	Iris-virginica

Pearson's Correlation	petal length	petal width	sepal length	sepal width
petal length	–	Strong 0.963	Strong 0.872	weak
petal width		–	Strong 0.818	weak
sepal length			–	weak
sepal width				–

Example: Iris dataset



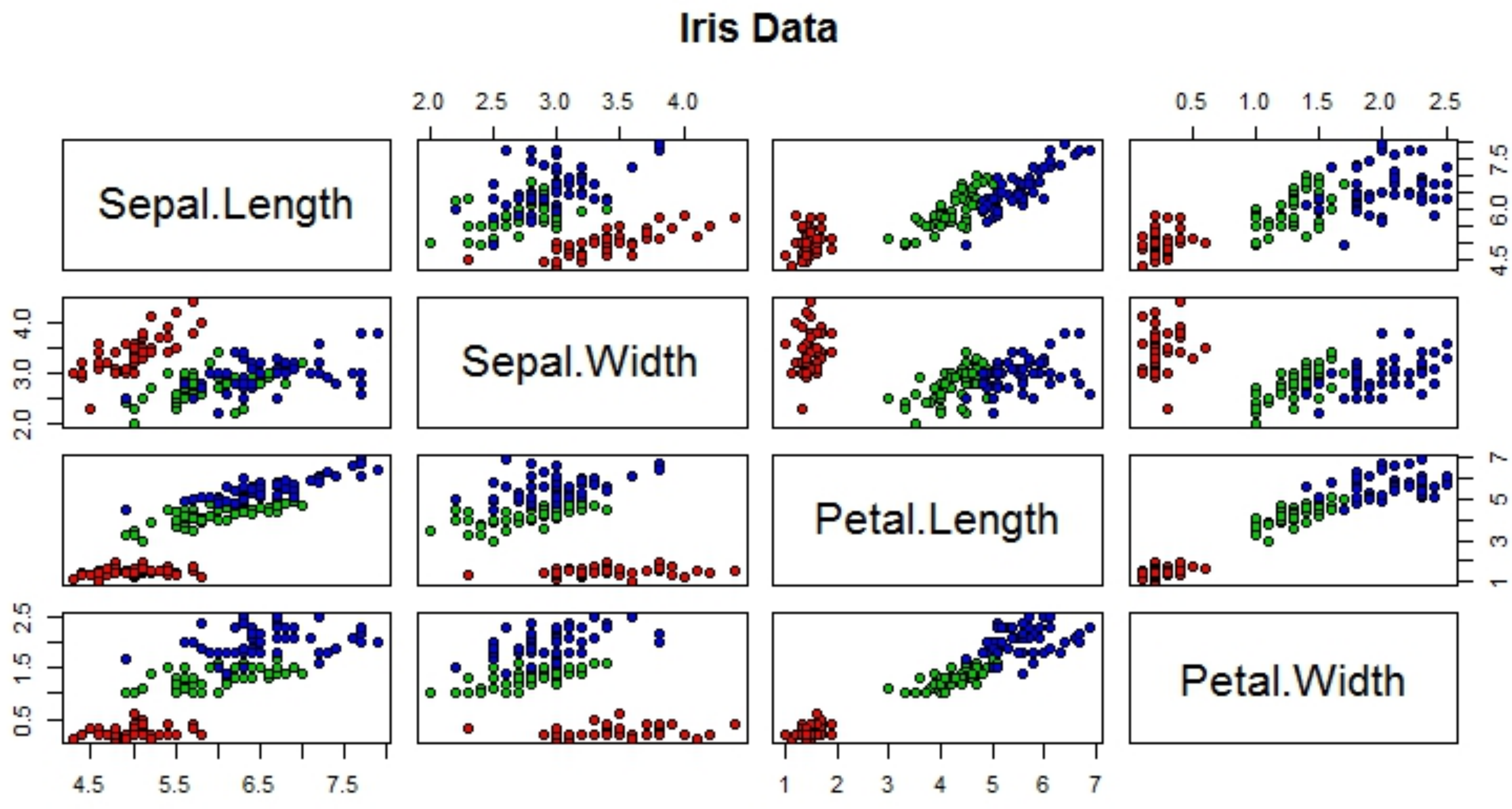
multidimensional array slices
petal length × petal with:

		Width	
Setosa	low	medium	high
Length low	46	2	0
medium	2	0	0
high	0	0	0

		Width	
Versicolor	low	medium	high
Length low	0	0	0
medium	0	43	3
high	0	2	2

		Width	
Virginica	low	medium	high
Length low	0	0	0
medium	0	0	3
high	0	3	44

Scatter Plot Array of Iris Attributes



Iris Initial Model

```
default: Iris-setosa
except
 if petal-length>=2.45 and petal-length<5.355 and petal-width < 1.75
 then Iris-versicolor
 except if petal-length >= 4.95 and petal-width < 1.55
        then Iris-virginica
        else
            if sepal-length < 4.95 and sepal-width >= 2.45
            then Iris-virginica
            else
                if petal-length >= 3.35
                then Iris-virginica
                except if petal-length < 4.85
                          and sepal-length < 5.95
                       then Iris-versicolor
```

Iris Attributte Discretization

- petal width: [0, 0.75) *low*, [0.75, 1.75) *medium*, [1.75, ∞) *high*
- petal length : [0, 2.5) *low*, [2.5, 5) *medium*, [5, ∞) *high*

Petal Length	Petal Width	Species Type	Count
low	low	Setosa	46
low	medium	Setosa	2
medium	low	Setosa	2
medium	medium	Versicolour	43
medium	high	Versicolour	3
medium	high	Virginica	3
high	medium	Versicolour	2
high	medium	Virginica	3
high	high	Versicolour	2
high	high	Virginica	44

Iris Final Model

```
PetLenght = low => Setosa
PetLenght = high => Virginica
PetLenght = medium
       PetWidht = low => Versicolor
       PetWidht = high
              sepal width <= 3.1 => Virginica
              sepal width   > 3.1 => Versicolor
       PetWidht = medium  => Versicolor
```

Attribute Combination

Combining one or more independent variables in a single variable

Example

Height	Length	Width	Box	Height	Length	Width	Box
2	12	2	Class1	12	4	2	Class2
6	4	2	Class1	4	12	2	Class2
3	8	2	Class1	8	6	2	Class2
4	4	3	Class1	4	8	3	Class2

Model:

```
Height <= 3: class1 (2)
Height > 3:
:...Length <= 4: class1 (3/1)
    Length > 4: class2 (3)
```

Propositional rules

Attribute Combination

New attribute - Volume: Height x Length x Width

Height	Length	Width	Volume	Box	Height	Length	Width	Volume	Box
2	12	2	48	Class1	12	4	2	96	Class2
6	4	2	48	Class1	4	12	2	96	Class2
3	8	2	48	Class1	8	6	2	96	Class2
4	4	3	48	Class1	4	8	3	96	Class2

Model:

```
Volume <= 48: class1 (4)
Volume > 48: class2 (4)
```

Relational rules

Relational rule extraction is only achieved with proper preprocessing - combination of variables

Principal Component Analysis (PCA)

General Idea

- Substitute the set of variables by a new (smaller) set where most of the “information” on the problem is still expressed

Goal

- Find a new set of axes onto which we will project the original data points

Data Reduction

- Use a smaller set of variables that contain the relevant information that is in the complete data
- The goal is to distinguish what is similar/different from the data using a smaller set of attributes

Principal Component Analysis (PCA)

- With PCA a new set of axes $y_1, y_2, \ldots, y_q$ are formed by linear combinations of the original variables $x_1,x_2,\ldots,x_n$ with $q < n$

$$
\begin{aligned}
y_1 &= a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n \\
y_2 &= a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n \\
&\ldots \\
y_q &= a_{q1}x_1+a_{q2}x_2+\ldots+a_{qn}x_n
\end{aligned}
$$

- We search for the linear combinations that "explain" most of the variability that existed among the data points on the original axes
- If we are "lucky" with a few of these new axes (ideally two for easy data visualization), we are able to explain most of the variability on the original data
- Each original observation is then "projected" into these new axes

Principal Component Analysis (PCA)

Algorithm

- Find a first linear combination which better captures the variability in the data

- After finding this linear combination (the first direction), PCA looks for a second linear combination that is orthogonal to the first one, and tries to capture the variability not explained by the first one, and so on..

- Continue until the set of new variables explains most of the variability (frequently 90% is considered enough)

Data Preparation Key Ideas

- Use meta-data
- Inspect data for anomalies and errors
- Eliminate “false positives”
- Develop small, reusable software components
- Plan for verification - verify the results after each step