COMP809 Data Mining and Machine Learning

Patricio Maturana-Russel

p.maturana.russel@aut.ac.nz

Department of Mathematical Sciences and Computer Science and Software Engineering Departments, Auckland University of Technology, Auckland, New Zealand

Semester 1, 2024

Contents

- Introduction: Data Mining and Machine Learning
- Attributes
- Data exploration

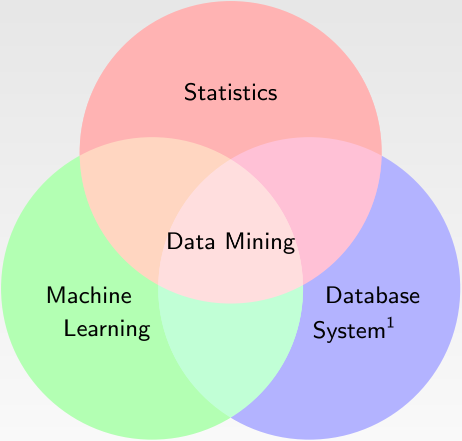
Introduction

Data Mining is the process of extracting and discovering patterns in large data sets (Wikipedia). It also includes the study and practice of data storage and data manipulation (SAS website).

Machine Learning (ML) is a method of data analysis that automates analytical model building (SAS website).

- **Unsupervised** ML methods learns patterns from untagged data.
- **Supervised** ML methods learns patterns from labelled data.

Relationship



[1] It deals with storing, retrieving, modifying, and analysing a database.

Data mining tasks

Two major categories:

- Description Methods
  - Find human-interpretable patterns (correlations, trends, clusters, trajectories and anomalies) that describe the data.
  - Often exploratory in nature, require post-processing techniques to validate and explain the results.
- Prediction Methods
  - Use some variables to predict unknown or future values of other variables.

Advances in Knowledge Discovery and Data Mining by Fayyad et.al., 1996.

Data mining tasks

- Regression [Predictive]
- Classification [Predictive]
- Clustering [Descriptive]
- Association Rule Discovery [Descriptive]
- Sequential Pattern Discovery [Descriptive]
- Deviation Detection [Predictive]

Machine Learning tasks

Data mining and machine learning have the same goal – to extract insights, patterns and relationships that can be used to make decisions – even though they have different approaches and abilities (SAS website).

Main ML tasks:

- Classification [Predictive]
- Clustering [Descriptive]

Data and Attributes

Data

It is a collection of number or pieces of information to which meaning has been attached.

The idea is turning data into information.

We can get characteristics from objects, entities, etc. $\implies$ data

Attributes

An attribute is an object's property or characteristics.

Type of attributes:

- Categorical:
  - Nominal
  - Ordinal
- Numerical:
  - Discrete
  - Continuous
    - Interval
    - Ratio

Attributes

Type of attributes:

- **Categorical**:
  - **Nominal**: label that provide only enough information to distinguish one object from another. Example: ID number, gender.
  - **Ordinal**: label that provide enough information to order the objects. Example: hardness of minerals {good, better, best}.
- **Numerical**:
  - **Discrete**: it can assume only a finite number of real values within a given interval. Example: number of students in the classroom.
  - **Continuous**: it can assume an infinite number of real values within a given interval. Example: height of a person.

Attributes

Continuous variables are classified as:

- **Interval**: numbers in which the differences between values are meaningful. Example: calendar dates, temperature in Celsius or Fahrenheit.
- **Ratio**: numbers in which both differences and ratios are meaningful. Example: temperature in Kelvin, counts, age, length.

Attributes

Properties of numbers typically used to describe attributes.

Attribute	Distinctness ($= \neq$)	Order ($< >$)	Addition ($+ -$)	Multiplication ($* /$)
Nominal	✓			
Ordinal	✓	✓		
Interval	✓	✓	✓	
Ratio	✓	✓	✓	✓

Knowing the type of an attribute allow us to avoid non-proper actions.

Attributes

Note that

- an average ID does not make sense;
- on the Kelvin scale, a temperature of 2° is, in a physically meaningful way, twice that of a temperature of 1°;
- on the Celsius scale, the above statement is not true.

Data exploration

Python cheat sheet

Numerical operators

addition	+
subtraction	-
multiplication	*
division	/
exponent	**
modulus	%
floor division	//

Conditional tests

equals	x == 38
not equal	x != 38
greater than	x > 38
or equal to	x >= 38
less than	x < 38
or equal to	x <= 38

Creating arrays in Numpy

```
import numpy as np
a=np.array([1,2])
b=np.array([(2,3.1),(5,6)],
  dtype=float) #filled by rows
```

Array information in Numpy

```
a.shape #dimensions
len(a) #length
b.ndim #number of dimensions
```

Aggregate functions in array

```
a.sum() #sum
a.min() #minimum value
a.max() #maximum value
a.cumsum()#cumulative sum
a.corrcoef()#correlation
```

Manipulating arrays with Numpy

```
a[1] #Select element at 2nd index
b[1,0] #Select element at row 1 and column 0
a[a>1] #values greater than 1
```

There are many Python cheat sheets available on Internet.

Data exploration

It is the first step in any analysis.

A preliminary exploration of the data is used to better understand its characteristics.

Key motivations of data exploration include:

- Helping to select the right tool for preprocessing or analysis.
- Making use of humans' abilities to recognize patterns.
  - We can recognize patterns not captured by data analysis tools.
- Detect anomalies.

Data exploration

An approach/philosophy for data analysis that employs a variety of techniques (mostly graphical) is Exploratory Data Analysis (EDA), initiated by John Tukey.

- The focus was on visualization.
- Clustering and anomaly detection were viewed as exploratory techniques.
- In data mining, clustering and anomaly detection are major areas of interest, and not thought of as just exploratory.

Find more information about EDA online on

https://www.itl.nist.gov/div898/handbook/index.htm

In this paper, we will focus on

- Summary statistics.
- Visualization.

Summary Statistics

NumPy is a library for Python with mathematical functions to operate arrays and matrices. After loading this library with "`import numpy as np`", we can calculate some statistics as follows:

Measure of	Statistic	Function
Central Tendency	mean	`np.mean(x)`
	median	`np.median(x)`
		`np.quantile(x,0.5)`
Dispersion	standard deviation	`np.std(x)`
	variance	np.var(x)
	range	`np.ptp(x)`
		np.max(x) - np.min(x)
	interquartile range	`np.quantile(x,0.75) - np.quantile(x,0.25)`

`x` is an array, defined as, for instance, `x = np.array([2,4,9])`.

Summary Statistics

Pandas is a library for Python for data manipulation and analysis. It allows to calculate statistics from a data frame.

Consider the example of the Titanic, data set available on Kaggle. The data has the following variables:

- `Survived`: Survival (true or false)
- `Pclass`: Passenger Class (1 = 1st; 2 = 2nd; 3 = 3rd)
- `Name`: Name
- `Sex`: Sex (female or male)
- `Age`: Age
- `SibSp`: Number of Siblings/Spouses Aboard
- `Parch`: Number of Parents/Children Aboard
- `Ticket`: Ticket Number
- `Fare`: Passenger Fare
- `Cabin`: Cabin
- `Embarked`: Port of Embarkation (C=Cherbourg; Q=Queenstown; S=Southampton)

Summary Statistics

```
import pandas as pd
titanic = pd.read_csv("titanic.csv"); # Importing a data frame

>>> list(titanic.columns) # variables
    ["PassengerId", "Survived", "Pclass", "Name", "Sex", "Age", "SibSp",
     "Parch", "Ticket", "Fare", "Cabin", "Embarked"]

# Central tendency
>>> titanic["Age"].mean()
    29.69911764705882
>>> titanic[["Age", "Fare"]].mean()
    Age      29.699118
    Fare     32.204208
    dtype: float64
>>> titanic[["Age", "Fare"]].median()
    Age      28.0000
    Fare     14.4542
    dtype: float64
```

Summary Statistics

```
# Dispersion
>>> titanic[["Age", "Fare"]].var()
    Age       211.019125
    Fare     2469.436846
    dtype: float64
>>> titanic[["Age", "Fare"]].std()
    Age      14.526497
    Fare     49.693429
    dtype: float64
>>> titanic[["Age", "Fare"]].max() - titanic[["Age", "Fare"]].min() # range
    Age       79.5800
    Fare     512.3292
    dtype: float64
>>> titanic[["Age", "Fare"]].quantile(0.75)- \
    titanic[["Age", "Fare"]].quantile(0.25) # interquartile range
    Age      17.8750
    Fare     23.0896
    dtype: float64
```

Summary Statistics

```
>>> titanic[["Age", "Fare"]].describe()
              Age        Fare
count  714.000000  891.000000
mean    29.699118   32.204208
std     14.526497   49.693429
min      0.420000    0.000000
25%     20.125000    7.910400
50%     28.000000   14.454200
75%     38.000000   31.000000
max     80.000000  512.329200

### Other functions ###
>>> titanic.groupby("Pclass")["Age"].mean()
Pclass
1     38.233441
2     29.877630
3     25.140620
Name: Age, dtype: float64
```

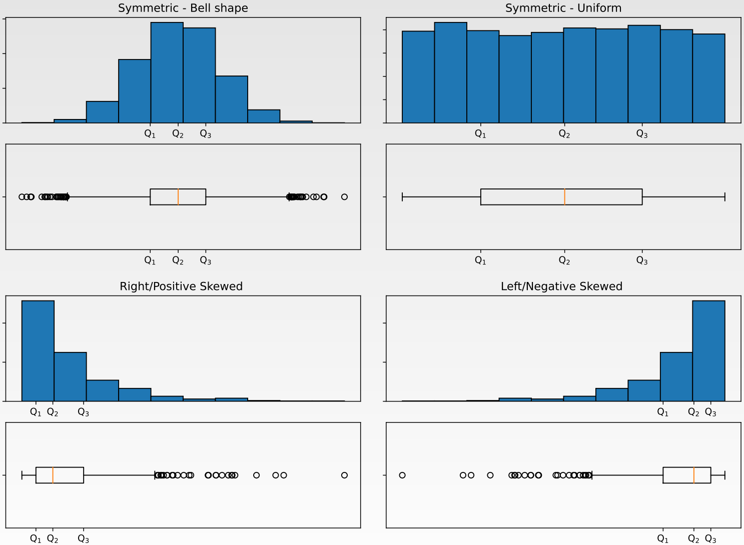
Summary Statistics

```
>>> titanic.agg({"Age":  ["min", "max", "median", "skew"],
                 "Fare": ["min", "max", "median", "mean"],})
              Age        Fare
    min      0.420000    0.000000
    max     80.000000  512.329200
    median  28.000000   14.454200
    skew     0.389108         NaN
    mean          NaN   32.204208

>>> titanic.groupby("Survived")["Survived"].count()
    Survived
    0    549
    1    342
    Name: Survived, dtype: int64
```

For more information visit https://pandas.pydata.org

Visualization



Visualization

Comments on the distribution of the data must include:

- Centre
- Dispersion
- Shape
- Outliers

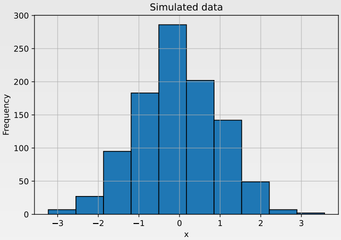
Visualization

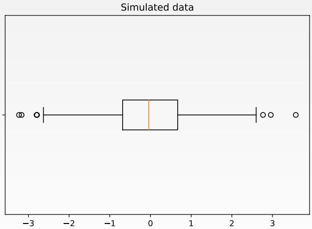
```
import numpy as np
from matplotlib import pyplot as plt

np.random.seed(123)
x=np.random.normal(size=1000,loc=0,scale=1)

plt.hist(x, bins=10, edgecolor="black")
plt.title("Simulated data")
plt.xlabel("x")
plt.ylabel("Frequency")
plt.grid(axis="y", alpha=0.75)
plt.grid(axis="x", alpha=0.75)
plt.show()

plt.clf() # clean window
plt.boxplot(x,vert=0)
plt.title("Simulated data")
ax = plt.gca()
ax.set_yticklabels([])
plt.show()
```





Data quality

Data quality problems:

- Noise and outliers
- Missing values
- Duplicate data

Data Quality

Causes:

- Incomplete data may come from
  - "Not applicable" data value when collected.
  - Different considerations between the time when the data was collected and when it is analyzed.
  - Human/hardware/software problems.
- Noisy data (incorrect values) may come from
  - Faulty data collection instruments.
  - Human or computer error at data entry.
  - Errors in data transmission.
- Inconsistent data may come from
  - Different data sources.
  - Functional dependency violation (e.g., modify some linked data).
- Duplicate records also need data cleaning.

Data Preprocessing

- Aggregation
- Missing values
- Duplicate data
- Data errors
- Outliers
- Data Normalization and Standardization
- Data Balancing
- Feature selection

Data Preprocessing

Aggregation: combining two or more attributes (or objects) into a single attribute (or object).

Purpose:

- Data reduction
  - Reduce the number of attributes or objects
- Change of scale
  - Cities aggregated into regions, states, countries, etc
- More "stable" data
  - Aggregated data tends to have less variability

Data Preprocessing

Missing values

Reasons:

- Information is not collected (e.g., people decline to give their age and weight)
- Attributes may not be applicable to all cases (e.g., annual income is not applicable to children)

Handling missing values:

- Eliminate Data Objects
- Estimate Missing Values
- Ignore the Missing Value During Analysis
- Replace with all possible values (weighted by their probabilities)
- Bayesian solution: replace them by probability distributions.

Data Preprocessing

```
>>> titanic.isnull().values.any() # any NaN
    True
>>> titanic.isnull().sum() # Number of NaN per attribute
    PassengerId      0
    Survived         0
    Pclass           0
    Name             0
    Sex              0
    Age            177
    SibSp            0
    Parch            0
    Ticket           0
    Fare             0
    Cabin          687
    Embarked         2
    dtype: int64

>>> titanic.fillna("XX") # replace NA values
>>> titanic["Age"].fillna(titanic["Age"].mean()) # replacing NaN with mean
>>> titanic.dropna() # Drop all rows with NaN values
```

For more information visit https://pandas.pydata.org

Data Preprocessing

Duplicate Data

Data set may include data objects that are duplicates, or almost duplicates of one another

- Major issue when merging data from heterogeneous sources

Examples:

- Same person with multiple email addresses

Data cleaning

- Process of dealing with duplicate data issues

Data Preprocessing

To check duplicates

```
>>> duplicate = titanic[titanic.duplicated()]
>>> duplicate
    Empty DataFrame
    Columns: [PassengerId, Survived, Pclass, Name, Sex, Age,
              SibSp, Parch, Ticket, Fare, Cabin, Embarked]
    Index: []
```

or we can check the ID number

```
>>> titanic["PassengerId"][titanic["PassengerId"].duplicated()]
    Series([], Name: PassengerId, dtype: int64)
```

To remove duplicates we use

```
titanic.drop_duplicates() #in case there is any in this dataframe
```

`X.duplicated()` returns TRUE or FALSE values.

Data Preprocessing

Data Error

We can check errors as follows:

```
>>> (titanic["Age"] < 0).values.any() # are all the values positive?
    False
>>> (titanic["Age"] > 100).values.any()
    False
>>> ((titanic["Age"] < 0) | (titanic["Age"] > 100)).values.any()
    False

>>> from pandas.api.types import is_numeric_dtype
>>> is_numeric_dtype(titanic['Age']) # are all the values numeric?
    True
```

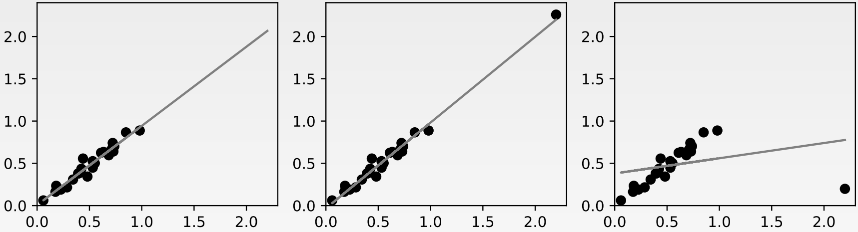
Data Preprocessing

Outliers

- Extreme values close to the limits of the data range or do not follow the trend of the remaining data.
- May represent errors occurred during data acquisition.
- May have a negative impact on the data mining method.
- They can be detected by visual inspection.

Data Preprocessing

Outliers



Data Preprocessing

Data Normalization

Problem: The ranges of certain variables may differ greatly from each other which can have a negative effect on the data mining technique. Variables with greater ranges have stronger impact on the results than others.

id	mpg	cylinders	cubic inches	hp
1	14	8	350	165
2	31.9	4	89	71
3	51.7	8	302	140
4	15	8	400	150
5	30.5	4	144	116.55
6	23	4	350	125

Data Preprocessing

Solution: Normalize the data to standardize the scale of each variable.
Example: Min-Max Normalization

$$X^* = \frac{X - X_{min}}{X_{max} - X_{min}}$$

id	mpg*	cylinders*	cubic inches*	hp*
1	0	1	0.84	1
2	0.47	0	0	0
3	1	1	0.68	0.73
4	0.03	1	1	0.84
5	0.44	0	0.18	0.48
6	0.24	0	0.84	0.57

Data Preprocessing

Normalization vs **Standardization**

- Normalization rescales the values into a range of [0,1].
- This might be useful in some cases where all parameters need to have the same positive scale.
- However, the outliers from the data set are lost.

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- Standardization rescales data to have a mean (μ) of 0 and standard deviation (σ) of 1 (unit variance)

$$X_{changed} = \frac{X - \mu}{\sigma}$$

Data Preprocessing

Data Balancing

- Often real world data is imbalanced.
- For a credit card stream, 99% of transactions are genuine while only 1% are in fraud.
- In such cases any machine learning algorithm will have difficulty in learning to find patterns that correlate to the "fraud" class.
- In such cases performance can be improved by either scaling down the majority ("genuine") class or creating new data for the minority ("fraud") class. These techniques are called:
  - Undersampling,
  - Oversampling,

  respectively.

Feature selection

It selects a subset of features among the set of all features. The idea is to find the optimal set of features.

Some of the methods used for this are:

- Pearson's correlation.
- ANOVA.
- Chi-Square.

Also known as *variable selection*, *feature reduction*, *attribute selection*, and *variable subset selection*.

End