# Topic 1: Overview

**Predictive modelling** (or 'data mining') uses "statistics" to build mathematical models which can be used for predicting. (`https://en.wikipedia.org/wiki/Predictive_modelling`)

Terminologies of similar meaning: ***Statistical learning***, ***machine learning***, `https://en.wikipedia.org/wiki/Predictive_analytics`.

Relevant Topics (To cover: ✓; Not cover: ✗):

- Supervised Learning Models:
  - Classifiers: kNN ✓, logistic regression ✓, neural network ✗, Naive Bayes ✓, LDA ✓, linear support vector machine (SVM), ✗, kernel SVM ✗, classification trees ✓, ensemble methods ✗, ...
  - Regressors: kNN ✓, linear regression ✓, regression tree ✓, support vector regressor (SVR ✗), ...
- Unsupervised Learning Models:
  - Feature agglomeration ✗
  - Random projections ✗
  - Dimensional Reduction: PCA ✓, ...
  - Clustering: k-Means ✓, Hierarchical clustering ✓, Spectral clustering ✗, t-SNE (`https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding`) ✗, Affinity Propagation ✗, Mean Shift clustering ✗, DBSCAN ✗, OPTICS ✗, Birch ✗, ...
  - Anomaly detection ✗
  - Association: `https://en.wikipedia.org/wiki/Apriori_algorithm` ✗
  - Autoencoders ✗
- **Semi-supervised learning**: Learning from a partially labelled data (see `https://scikit-learn.org/stable/modules/label_propagation.html`).
- Time-series data mining: Time series data is time-stamped and collected over time at a particular interval (sales in a month, calls per day, web visits per hour, etc.). Time series data mining combines traditional data mining (such as sampling, clustering and decision trees) and forecasting techniques. ✗
- Self-supervised learning ✗
- Reinforcement learning ✗: Value-based, Policy-based, Model-based.

  The goal is to find a good behaviour, an action or a label for each particular situation to maximise the long-term benefits that the agent receives.
- Generative adversarial network (GAN)

## 1.1 Software, Data and References

Since predictive modelling deals with statistical models which are complex, manual calculation is mostly impossible except for some cases which will be explored in this course and in final exam.

Popular programming languages and relevant packages for data analysis and predictive modelling are shown below:

- Python + Pandas + Matplotlib + Google Tensorflow + PyTorch + ... → Anaconda Python
- R + glmnet + tree + rpart + gbm + ... → `https://cran.r-project.org/` [Chambers, 2008], [Spector, 2008]
- Java + NoSQL + Weka + SparkML + ...
- Excel, Power BI, Microsoft Access, SQL in business environment

Popular data:

- https://archive.ics.uci.edu/ml/datasets.php
- Kaggle (need registration)
- Prof. Julian McAuley has gathered datasets associated with online shopping and social network platforms at `https://cseweb.ucsd.edu/~jmcauley/datasets.html`.

Useful references for predictive modelling:

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, An Introduction to Statistical Learning: with Applications in R, 2nd ed., Springer 2021

  `https://statlearning.com/` (Second edition is available for download)

- Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, 2008 `https://hastie.su.domains/ElemStatLearn/`

- Academic journals (Web of Science, Scorpus):
  - 0004-3702,Artificial Intelligence: `https://www.sciencedirect.com/journal/artificial-intell`
  - 0160-2896,Intelligence: `https://www.sciencedirect.com/journal/intelligence`
  - 0162-8828,IEEE Transactions on Pattern Analysis and Machine Intelligence: `https://www.computer.org/csdl/journal/tp`
  - 0269-2821,Artificial Intelligence Review: `https://www.springer.com/journal/10462`
  - 0218-0014,International Journal of Pattern Recognition and Artificial Intelligence: `https://www.worldscientific.com/worldscinet/ijprai`
  - 0824-7935,Computational Intelligence: `https://onlinelibrary.wiley.com/journal/14678640`
  - 0218-2130,International Journal on Artificial Intelligence Tools
  - 0263-4503,Marketing Intelligence and Planning
  - 0268-4527,Intelligence and National Security
  - 0334-1860,Journal of Intelligent Systems
  - 0883-9514,Applied Artificial Intelligence
  - 0884-8173,International Journal of Intelligent Systems

- 0885-0607,International Journal of Intelligence and CounterIntelligence
- 0885-6125,Machine Learning
- 0890-0604,AI EDAM-Artificial Intelligence for Engineering Design, Analysis and Manufacturing https://www.cambridge.org/core/journals/ai-edam
- 0921-0296,Journal of Intelligent and Robotic Systems: Theory and Applications
- 0922-6389,Frontiers in Artificial Intelligence and Applications
- 0924-669X,Applied Intelligence
- 0924-8463,Artificial Intelligence and Law
- 0925-9902,Journal of Intelligent Information Systems
- 0933-1875,KI - Kunstliche Intelligenz
- 0933-3657,Artificial Intelligence in Medicine
- 0952-1976,Engineering Applications of Artificial Intelligence
- 0952-813X,Journal of Experimental and Theoretical Artificial Intelligence
- 0956-5515,Journal of Intelligent Manufacturing
- 0974-0635,International Journal of Artificial Intelligence
- 0974-8571,International Journal of Computational Intelligence in Control
- 0992-499X,Revue d'Intelligence Artificielle
- 1003-6059,Moshi Shibie yu Rengong Zhineng/Pattern Recognition and Artificial Intelligence http://manu46.magtech.com.cn/Jweb_prai/EN/volumn/home.shtml
- 1012-2443,Annals of Mathematics and Artificial Intelligence
- 1045-389X,Journal of Intelligent Material Systems and Structures: https://journals.sagepub.com/home/jim
- 1064-1246,Journal of Intelligent and Fuzzy Systems
- 1076-9757,Journal of Artificial Intelligence Research
- 1079-8587,Intelligent Automation and Soft Computing
- 1088-467X,Intelligent Data Analysis
- 1120-9550,Sistemi Intelligenti
- 1178-5608,International Journal on Smart Sensing and Intelligent Systems
- 1343-0130,Journal of Advanced Computational Intelligence and Intelligent Informatics: https://www.fujipress.jp/jaciii/jc/
- 1346-0714,Transactions of the Japanese Society for Artificial Intelligence
- 1469-0268,International Journal of Computational Intelligence and Applications
- 1524-9050,IEEE Transactions on Intelligent Transportation Systems
- 1532-4435,Journal of Machine Learning Research
- 1541-1672,IEEE Intelligent Systems
- 1547-2450,Journal of Intelligent Transportation Systems
- 1548-3657,International Journal of Intelligent Information Technologies
- 1550-1949,Intelligent Systems in Accounting, Finance and Management: https://onlinelibrary.wiley.com/journal/21600074
- 1556-603X,IEEE Computational Intelligence Magazine
- 1557-3958,International Journal of Cognitive Informatics and Natural Intelligence
- 1560-4292,International Journal of Artificial Intelligence in Education
- 1598-2645,International Journal of Fuzzy Logic and Intelligent Systems
- 1616-1262,Journal of Intelligence History
- 1687-5265,Computational Intelligence and Neuroscience
- 1687-9724,Applied Computational Intelligence and Soft Computing
- 1724-8035,Intelligenza Artificiale
- 1740-2832,International Journal of Technology Intelligence and Planning
- 1740-8865,International Journal of Intelligent Systems Technologies and Applications
- 1743-8187,International Journal of Business Intelligence and Data Mining
- 1745-3232,International Journal of Intelligent Enterprise

- 1750-8975,Intelligent Buildings International
- 1751-5858,International Journal of Intelligent Information and Database Systems
- 1751-956X,IET Intelligent Transport Systems
- 1755-0386,International Journal of Advanced Intelligence Paradigms
- 1755-0556,International Journal of Reasoning-based Intelligent Systems
- 1756-378X,International Journal of Intelligent Computing and Cybernetics
- 1860-949X,Studies in Computational Intelligence
- 1861-2776,Intelligent Service Robotics
- 1864-5909,Evolutionary Intelligence
- 1868-4394,Intelligent Systems Reference Library
- 1868-5137,Journal of Ambient Intelligence and Humanized Computing
- 1868-8071,International Journal of Machine Learning and Cybernetics
- 1868-8659,International Journal of Intelligent Transportation Systems Research
- 1872-4981,Intelligent Decision Technologies
- 1875-6891,International Journal of Computational Intelligence Systems
- 1876-1364,Journal of Ambient Intelligence and Smart Environments
- 1897-8649,"Journal of Automation, Mobile Robotics and Intelligent Systems"
- 1935-3812,Swarm Intelligence
- 1935-8237,Foundations and Trends in Machine Learning
- 1939-1390,IEEE Intelligent Transportation Systems Magazine
- 1939-4608,Synthesis Lectures on Artificial Intelligence and Machine Learning
- 1941-6237,International Journal of Ambient Computing and Intelligence
- 1947-3591,International Journal of Business Intelligence Research
- 1947-9263,International Journal of Swarm Intelligence Research
- 1989-1660,International Journal of Interactive Multimedia and Artificial Intelligence
- 2001-015X,Journal of Intelligence Studies in Business
- 2049-6427,International Journal of Intelligent Unmanned Systems
- 2079-3200,Journal of Intelligence
- 2083-2567,Journal of Artificial Intelligence and Soft Computing Research
- 2089-4872,IAES International Journal of Artificial Intelligence
- 2147-6799,International Journal of Intelligent Systems and Applications in Engineering: `https://ijisae.org/index.php/IJISAE`
- 2157-6904,ACM Transactions on Intelligent Systems and Technology
- 2160-6455,ACM Transactions on Interactive Intelligent Systems
- 2185-310X,International Journal of Intelligent Engineering and Systems
- 2192-6352,Progress in Artificial Intelligence
- 2199-4536,Complex & Intelligent Systems
- 2199-4668,Journal of Reliable Intelligent Environments
- 2209-6051,Communicable diseases intelligence (2018)
- 2213-8986,"Intelligent Systems, Control and Automation: Science and Engineering"
- 2366-5971,International Journal of Intelligent Robotics and Applications
- 2379-8858,IEEE Transactions on Intelligent Vehicles
- 2380-0992,"International Journal of Intelligence, Security, and Public Affairs"
- 2405-6456,Web Intelligence
- 2414-8105,Journal of Network Intelligence
- 2442-6571,International Journal of Advances in Intelligent Informatics
- 2468-6557,CAAI Transactions on Intelligence Technology
- 2471-285X,IEEE Transactions on Emerging Topics in Computational Intelligence
- 2522-5839,Nature Machine Intelligence: https://www.nature.com/natmachintell/

## 1.2 Learning Outcomes and Assessment

The "learning outcomes" of this subject are

**CO1:** Describe the key concept of statistical learning;

**CO2:** Compare statistical models for prediction and estimation through supervised learning;

**CO3:** Identify relationship and structures from unlabelled data through unsupervised learning;

**CO4:** Demonstrate supervised and unsupervised learning with statistical software;

**CO5:** Interpret results from supervised and unsupervised learning.

Class Activities:

**Week 1:**
- Start of tutorial
- Start of practical
- Start to find assignment group members, 4–7 in a group. Individuals without any group with be either grouped the lecturer if there are 4–7, otherwise, they will be assigned to groups with 4–6 members;

**Week 3:** Everyone in class should be part of an assignment group. Lecturer will assign anyone who has not formed a group to groups with least members and deduct the marks of the student without any group by 0.2 out of the total.

**Week 4:** Assignment starts.

**Week 6 or 7:** Quiz covering topic 1 to 3(?).

**Week 11:** Submission of assignment report and computer program. Oral presentation may start if the lectures are completed, otherwise, the oral presentation will be in Week 12.

**Week 14:** Assignment markings should be out by Friday (if not, it would be in Week 15). check the marks and inform lecturer if there is any mistakes.

Assessments:

- Physical Practical Quiz (CO4): 12%.
- Assignment (38%): Report 18% (CO1 + CO2 + CO3) + Programming Code 10% (CO4) + Oral Presentation 10%
- Final Exam (100 marks $\times 0.5 \Rightarrow 50\%$): 3 + 1 Questions.
  - Q1: CO1, Supervised + Unsupervised, mainly Topic 1 and some from all other topics. 25 marks
  - Q2: CO2, Supervised Learning Models, 25 marks
  - Q3: CO3, UnSupervised Learning Models, 25 marks
  - Q4/Q5: CO5, Supervised + Unsupervised, 25 marks

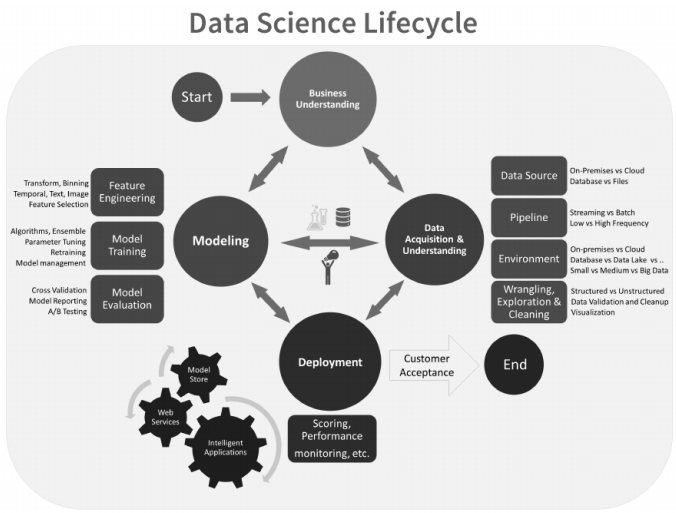
## 1.3 Data Science and CRISP-DM Framework

The *CRISP-DM (Cross Industry Standard Process for Data Mining)* methodology provides a structured approach to planning a "data mining" or "data science" project. The process or methodology of CRISP-DM is described in these six major steps [Swamynathan, 2017]:

1. **Business Understanding**
    - Focuses on understanding the project objectives and requirements from a business perspective, and converts into a data mining problem definition and a preliminary plan.
2. **Data Understanding**
    - Starts with an initial data collection and proceeds with activities in order to get familiar with the data, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets to form hypotheses for hidden information.
    - Deciding whether the goal of the KDD (knowledge discovery in databases) process is classification, regression, clustering, etc.
3. **Data Preparation**
    - Covers all activities to construct the final dataset for a model from the initial raw data.
4. **Modelling**
    - Select models (and the hyperparameters) based on interpretability and/or flexibility.
    - Since some modelling techniques have specific requirements regarding the form of data, there can be a loop back here to data preparation.
    - Searching for patterns of interest in a particular representational form or a set of such representations as classification rules or trees, regression, clustering, and so forth.
5. **Evaluation**
    - Once one or more models have been built that appear to have high quality based on whichever loss functions have been selected, these need to be tested to ensure they generalise against unseen data and that all key business issues have been sufficiently considered. The end result is the selection of the champion model(s).
6. **Deployment**
    - Generally this will mean deploying a code representation of the model into an operating system to score or categorise new unseen data as it arises and to create a mechanism for the use of that new information in the solution of the original business problem.
    - The code representation must include all the data preprocessing steps leading up to modelling so that the model will treat new raw data in the same manner as during model development.

**Example 1.3.1** (Final Exam Jan 2019, Q4(a))**.** State the phases involved in cross industry standard process of data mining (CRISP-DM). (5 marks)

**Remark 1.3.2.** Most redictive modelling frameworks are similar to CRISP-DM with some variations, for example, the Microsoft Data Science Lifecycle:




https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle-business-understanding

## 1.4 Business Understanding

- Define the (business) goals that the data science techniques can target.
- Find the relevant data that helps you meet the goals / answer the questions
- Many of your seniors apply the predictive models on the data and choose the "best" model forgetting the "goal" (which is not the right direction)
- The right goal: "How do the factors influence the target?" OR "How does the best model help business?"

**Example 1.4.1.** Suppose the **goal** is to understand the factors that affect the height of a person.

- Age
- Amount of carbohidrates
- Amount of protein
- Amount of fibre

- Quantity and quality of exercises
- Hours of sleep
- etc.

Business understanding: Which factors are the easiest to perform data collection and more likely to **influence** the height?

## 1.5 Data Understanding

Data sources:

- Structured Data (EDA can be used):
  - Most companies usually stored data in structured data format in CSV/Excel, SQL database or NoSQL database.
  - SQL: Microsoft SQL (most popular), Oracle, PostgresQL, MySQL (used to be very popular in Web mangement), MariaDB, etc.
  - NoSQL: https://en.wikipedia.org/wiki/SPARQL, DataLog etc.
- Unstructured Data (EDA cannot be used):
  - Texts: Reports in Word/PDF; Twitters; etc.
  - Images (of different sizes and resolutions)
  - Biometric data
  - Songs / Lyrics
  - Time series: Stock price; Online game control sequence; Industrial robot control sequence; etc.

### 1.5.1 Unstructured data

What about **unstructured data**? There is no simple answer, we need to convert them to structured data in various ways!

- We need to convert (rescale) an image to a 2D matrix or a 3D array of a fixed shape
- We need to convert texts to 2D matrix with individual words as columns, rows as word counts. This is called the **bag of word** representation. Another more power representation is the TF-IDF (Term Frequency-Inverse Document Frequency) representation.
- Human speech signals? Musics? Videos? etc.

### 1.5.2 Structured data

To know the **structured data** — `https://en.wikipedia.org/wiki/Exploratory_data_analysis` (EDA):

- Summary statistics
- Data visualisation

Numeric statistical summary associated with a **univariate data**:

- Categorical (or nominal) data (e.g. Gender): frequency and mode. R's `factor`; Python's `astype("category")`
- Ordinal data (e.g. Student grade): percentiles(?) R's `ordered`
- Numerical data (e.g. Temperature): mean, median, quantiles, range, variance (dependent on the measurement units? E.g. 1 metre or 100 cm?), skewness

Data Visualisation associated with **univariate data**:

- Categorical and ordinal data: pie chart, barplot / bar graph
- Numerical data: histogram, boxplot, density plot

EDA tools for the univariate data analysis in R:

- R's `summary`, Python's `df.describe()`, `scipy.stats.describe`
- For (continuous) numerical data: R's `hist` (histogram), `stem` (stem-and-leaf plot), `qqplot`, Python's `plt.hist()`
- For integral data and categorical data, R's `table`.

EDA tools for the bivariate data analysis in R:

- categorical vs categorical: `barplot`, `table`,
- categorical vs numerical: `boxplot`
- numerical vs numerical:
  - Correlation: `cor(x, y, method="pearson")`, `cor.test(x, y)` (for Hypothesis testing)
  - (scatter) `plot` (calls `pairs`)
  - `heatmap`

### 1.5.3 Data Cleaning/Cleansing

Whether the real-world data comes from the 'databases' or from Excel files or Internet, there may be noise, i.e. missing values, wrong values, etc. To clean up the data:

- **Identify** missing/wrong values and **impute** if necessary
- Don't treat any column as outlier!
- Outliers are rows which are unique (e.g. the only row with a unique value. E.g. only one row with customer status VVVIP)

Predictive modelling deals with 'majority' data not a very very special / rare case.

Incorrect or inconsistent data leads to false conclusions. Therefore, understanding and cleaning data are essential to obtain a "useful" data for our predictive models. According to `https://en.wikipedia.org/wiki/Data_cleansing`, *data cleaning* or *data cleansing* is the process of detecting and correcting (or removing) corrupt or inaccurate records from the data:

- Duplicated data should be removed;
- Strings should be clean (e.g. trimming spaces, etc.)

- Missing values should be "dropped", "imputed" by replacing them with mean/mode or "regression value" or hot-deck (copy from similar row), or random valid value, etc.
- Outliers should be removed if the predictive model is sensitive to them.

## 1.6 Data Preparation / Preprocessing

Data preparation/preprocessing tries to convert the data of interest to a **structured** data or from a structured data to a properly scaled structured data.

- In R, the `caret`'s `preProcess` provides BoxCox, YeoJohnson, expoTrans, knnImpute, bagImpute, medianImpute, pca, ica, spatialSign, corr, zv, nzv, conditionalX preprocessing methods;
- In Python, preprocessing methods are available in `sklearn.preprocessing`.

### 1.6.1 Preprocessing Text Data

Some preprocessing steps are:

- Removing punctuations like . , ! $ ( ) * % @
- Removing Stop words: a, an, the, is, at, what, which, on, or, and, but, how, ...
- Removing URLs (?)
- Lower casing
- Sentence, Word tokenisation: the process of splitting a large sample of text into words
- Stemming: the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. E.g. cats → cat, ate → eat, planned → plan, ...
- Lemmatization: the process of grouping together the inflected forms of a word so they can be analysed as a single item. E.g. "better" has "good" as its lemma, Similarly, "worse" & "bad", etc.

Python's Tokenisation Example

```
import nltk
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
sentence_data = "Sun rises in the east. Sun sets in the west."
sentence_tokens = nltk.sent_tokenize(sentence_data)
porter_stemmer = PorterStemmer()
wordnet_lemmatizer = WordNetLemmatizer()
for sentence in sentence_tokens:
    word_tokens = nltk.word_tokenize(sentence)
    print(word_tokens)
    for word in word_tokens:
        print(wordnet_lemmatizer.lemmatize(porter_stemmer.stem(word)))
        # crazy-> crazi, available-> avail, entry-> entri, early-> earli
```

### 1.6.2 Preprocessing Image Data

Image preprocessing steps:

- Cropping
- Rescaling (with Gaussian smoothing)

- Projective transforms: translation and rotation

Python's image processing libraries:

- PIL/pillow (Python Image Library): Very primitive
- OpenCV: Industrial standard
- Scikit-image: Works with Numpy array and interacts with OpenCV, etc.
- Mahotas: `https://mahotas.readthedocs.io/en/latest/index.html`

### 1.6.3 Preprocessing Data with Categorical Features

**Encoding categorical features**:

- one-hot encoding:

```
final_df = model.matrix(~0+col1+col2, data=d.f)
# Alternatively:
library(caret)
oneh = dummyVars( ~ ., data=d.f)
final_df = data.frame(predict(oneh, newdata=d.f))
```

- Advanced methods (non-standard): `http://contrib.scikit-learn.org/category_encoders/`

**Encoding ordered features**:

- ordinal encoding: R's as.integer(), Python's sklearn.preprocessing.OrdinalEncoder()

### 1.6.4 Preprocessing Data with Numeric Features

Preprocessing is need for some predictive models to perform better in the training process.

- **Standardisation** (column scaling): `scale(d.f)`. It is a common requirement for many machine learning estimators; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.

$$M(X_{ij}) = \frac{X_{ij} - \overline{X}_j}{s_{X,j}} \tag{1.1}$$

where

$$s_{X,j} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_{ij} - \frac{1}{N}\sum_{i=1}^{N}X_{ij})^2}.$$

- Min-max scaling

$$M(X_{ij}) = \frac{X_{ij} - X_{\text{min,j}}}{X_{\text{max,j}} - X_{\text{min,j}}}. \tag{1.2}$$

- Generating polynomial features: `poly()`
- Non-linear and custom transformation
- Dimensional reduction
- Discretization: `arules::discretize`
- **Normalisation** (row scaling): It scales individual samples to have unit norm. This process can be useful if we plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

### 1.6.5 Min-Max Normalisation (Rescaling)

Suppose we have a 2-D data $(X_{ij})_{i=1,\cdots,n;j=1,\cdots,p}$, a *min-max scaler (or normalisation)* (1.2) is a kind of standardisation which transforms all variables into the interval [0,1].

In R, we could follow the advice from `https://stackoverflow.com/questions/24520720/subtract-a-constant-vector-from-each-row-in-a-matrix-in-r`,

```
M.X = t((t(X) - sapply(X,2,min))/(sapply(X,2,max)-sapply(X,2,min)))
```

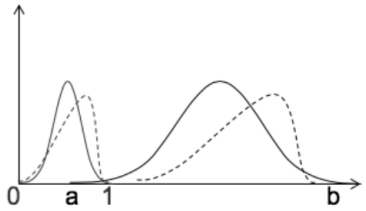
In Python, we can use numpy array with

```
M_X = (X-X.min(axis=0))/(X.max(axis=0)-X.min(axis=0))
```

or using the pre-processing functions from sklearn:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X)        # Compute the min-max to be used for later scaling
M_X = scaler.transform(X)  # scaler.fit_transform merge these two steps
```

Min-Max normalisation provides an easy way to compare values that are measured using different scales or different units of measure. An illustration of the scaling effect for 1D data is shown below.



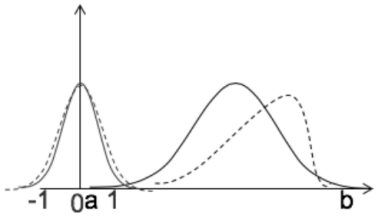

### 1.6.6 Standard Scaler/Standardisation

A *standard scaler* or a "*standardisation*" (1.1) is another method to "scale features" by transforming all variables to standard normal distribution, $N(0,1)$.

**Software Implementations**

- R: `scale(column)` (or `caret`'s `preProcess(df, method=c("center", "scale"))`
- Python: `sklearn.preprocessing`'s `StandardScaler` and `scale` (which needs to be multiplied by $\sqrt{\frac{N-1}{N}}$ to get sample statistics).

A diagram to illustrate the effect of standardisation (1.1) is shown below, i.e. it transforms normally distributed data in the range $(a,b)$ to $(-1,1)$.




**Example 1.6.1** (May 2022 Semester Final Exam, Q1(a))**.** Given the data in Table 1.1, write down the formula for standardisation and perform the standardisation pre-processing on the data in Table 1.1.

| Obs. | $x_1$ | $x_2$ |
|---|---|---|
| 1 | 3.3 | 4.4 |
| 2 | 2.4 | 3.1 |
| 3 | 0.1 | 1.9 |
| 4 | 0.3 | 2.4 |
| 5 | −0.6 | 1.1 |
| 6 | −2.9 | −0.1 |
| 7 | 4.3 | 6.4 |
| 8 | 3.4 | 5.1 |
| 9 | 1.1 | 3.9 |

Table 1.1: Two-dimensional data.

(6 marks)

**Solution:** The formula for standardisation is $x_{ij} \mapsto \dfrac{x_{ij} - \overline{x}_{\cdot j}}{\hat{\sigma}_{\cdot j}}$ ..... [1 mark]

The mean and sample standard deviation for the first and second columns are respectively

$$\overline{x}_{\cdot 1} = 1.266667, \quad \hat{\sigma}_{\cdot 1} = 2.300543; \quad \overline{x}_{\cdot 2} = 3.133333, \quad \hat{\sigma}_{\cdot 2} = 2.042670 \qquad \text{[2 marks]}$$

and the table after standardisation is

| Obs. | $x_1$ | $x_2$ |
|---|---|---|
| 1 | 0.8838 | 0.6201 |
| 2 | 0.4926 | −0.0163 |
| 3 | −0.5071 | −0.6038 |
| 4 | −0.4202 | −0.3590 |
| 5 | −0.8114 | −0.9954 |
| 6 | −1.8112 | −1.5829 |
| 7 | 1.3185 | 1.5992 |
| 8 | 0.9273 | 0.9628 |
| 9 | −0.0724 | 0.3753 |

..... [3 marks]

### 1.6.7 Imputation of Missing Values

- If there are very few missing values (e.g. less than 1%) we may throw away the rows with few missing values: `na.omit()`
- If there are too many missing values (e.g. $> 50\%$), we need to find out where are they coming from:
  - If one or two columns are the cause, we remove them
  - If the missing values are spreaded everywhere, we use missing value heatmap to try to identify some patterns and decide on how to impute the data if necessary. E.g.
    - for many predictive models such as logistic regression, naive Bayes, etc. we need to impute the data;
    - for CART decision tree, we may not need to impute the data

## 1.7 Modelling

The predictive models we are going to investigage are mostly only handles **tabular data with time-independent features** such as customer age, gender, income range, product items, etc. We assume there are $p$ features:

- the data is of the shape $\boxed{n \times p}$ with **no label**, then unsupervised learning could be applied.
- If the data is of the shape $\boxed{n \times (p+1)}$, i.e. the input $X$ is $n \times p$ and the **label** (also known as **output**, **target**) $Y$ is $n \times 1$, then unsupervised learning could be applied to 'identify' patterns in $X$ and supervised learning models may be applied on $(X, Y)$.
  - If the label is numerical / quantitative (e.g. sales figure), the modelling problem is called a **regression problem** and the model is called a **regressor**;
  - If the label is categorical / qualitative (e.g. spam, non-spam), the modelling problem is called a **classification problem** and the model is called a **classifier**.

### 1.7.1 Modelling Unlabelled Data

If the data is mostly numeric (or can be converted to numeric values) and **has no labels** such as

- new genetic information (e.g. new variation of COVID-19 viruses or other coronavirus);
- new customer/marketing data in which patterns need to be uncovered; etc.

Unsupervised Learning methods below could be tried:

- Descriptive Statistics / EDA. E.g. Data understanding (Section 1.5)
- Visualisation → Dashboard
- Dimensionality Reduction. E.g. PCA (Chapter 8)
- Clustering. E.g. k-means, HC (Chapter 9)
- Unfolding the graph/network structures
- `https://en.wikipedia.org/wiki/Association_rule_learning`
- `https://en.wikipedia.org/wiki/Anomaly_detection`

### 1.7.2 Modelling 'Continuous' Labelled Data — Regression Problems

In a predictive model, if the target variable (or response) $Y$ is numerical or *quantitative*, the problem is called a *regression problem* and the predictive model is called a ***regressor***.

The popular regressors are mostly discriminative, therefore, it is probably more common to classify regressors by "linearity":

- Linear regression models:

$$Y = \sum_i w_i X_i \Rightarrow y_i = w_0 + w_1 x_{1i} + \cdots + w_p x_{pi}$$

- Nonlinear regression models:

$$Y = \sum_i w_i \phi_i(X_1, ..., X_p) \Rightarrow y_i = w_0 + w_1 \phi_1(x_{1i}, \cdots, x_{pi}) + \cdots + w_p \phi_p(x_{1i}, \cdots, x_{pi}).$$

**Exercise 1.7.1** (Simple Linear Regression)**.** If the linear regression (a kind of supervised learn-

ing method) is used to model the relation between $y$ and $x$ as follows.

| | | | | | |
|---|---|---|---|---|---|
| $y$ | 23.82 | 47.16 | 66.66 | 88.39 | 110.54 |
| $x$ | 1 | 2 | 3 | 4 | 5 |
| $y$ | 131.1 | 174.15 | 214.72 | 233.9 | 252.14 |
| $x$ | 6 | 8 | 10 | 11 | 12 |

Predict the value at $x = 7$ using the linear regression model. [Ans: $\hat{y} = 150.93$]

**Example 1.7.2.** (Simple model training and prediction in R)

```
#install.packages("SomePackageName")
#library(SomePackageName)
model = lm(y ~ ., data=Xy)
# Some models can be used for statistical inference
print(model)  # or print(summary(model))
# Deployment: prediction
predicted = predict(model, newdata=data.frame(x1=...,x2=...))
```

**Example 1.7.3.** (Simple model training and prediction in Python)

```
from sklearn.linear_model import LinearRegression
lrobj = LinearRegression()
model = lrobj.fit(Xy.iloc[:,3:4],Xy.iloc[:,4])
newdata = pd.DataFrame({'x1':..., 'x2':...})
predicted = model.predict(newdata)
```

### 1.7.3 Modelling 'Discrete' Labelled Data — Classification Problems

Modelling means to use a mathematical model $Y = h(X)$ to fit the observed data:

$$(\boldsymbol{x}_1, y_1),\ (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n).$$

such that the total errors $\sum_i diff(y_i, h(\boldsymbol{x}_i))$ is acceptably small. Note that $(\boldsymbol{x}_i, y_i)$ can come be marketing or scientific data obtained through surveys or observations and they are usually stored in a tabular form which have been cleaned.

Note that the $\boldsymbol{x}_i$ are usually called **inputs** / **attributes** / **features** / **columns** / **independent variables** / **explanatory variables** / **predictors** / **explanatory factors**, etc. [Afifi and Clark, 1997] and may have more than one components:

$$\boldsymbol{x}_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,p}).$$

The $y_i$ are usually called the **output** / **label** / **target** / **response** / **dependent variable** / **outcome**, etc. and is usually a single component.

The mathematical model to fit the observed data $(\boldsymbol{x}_i, y_i)$ is called a **predictive model**:

$$\underbrace{Y}_{\text{output}} = h(\underbrace{X_1, \cdots, X_p}_{input}) =: h(X). \tag{1.3}$$

The model $h$ may be used for

- Prediction: If we just want to know "for a given input $(x_1, \cdots, x_p)$, what is the value of $y$?"
- Inference: Is the model correct? How the output $Y$ is changing w.r.t. the input $X_i$? E.g. What factors "improves" sales?

An approach to classify the predictive models (classifiers in particular) based on the 'Bayesian statistics' point of view:

- **Discriminative models**:

$$h(X) = \text{argmax}_j \, \mathbb{P}(Y = j | X_1, \cdots, X_p)$$

  E.g. linear regression, kNN, logistic model, etc.

- **Generative models**

$$h(X) = \text{argmax}_j \, \mathbb{P}(X_1, \cdots, X_p | Y = j) \mathbb{P}(Y = j)$$

  E.g. Naive Bayes, LDA, etc.

In the discriminative modelling, one aims to learn a predictor given observations; In the generative modelling, one aims to learn a joint distribution over all variables.

We usually use a 'family' of models $h(X)$ with 'internal' parameters. The characteristic of the "number of parameters" allows us to classify models into

- **Parametric models**:
  - Models with **fixed** set of parameters
  - "Training" tries to find the most suitable parameter values to minimise "errors"
  - E.g. logistic regression
- **Nonparametric models**:
  - Models without fixed set of parameters. Internal "representation" **grows as data increases**.
  - "Training" tries to "fit" the data into the model!
  - E.g. kNN

**Example 1.7.4** (Churn Prevention — Classification Problem)**.** You have just landed a great analytic job with MegaTelCo, a telecommunication firm. They are having a major problem with customer retention in their wireless business. In a particular state, 20% of cell phone customers leave when their contracts expire, and it is getting increasingly difficult to acquire new customers.

Since the cell phone market is now saturated, the huge growth in the wireless market has tapered off. Communications companies are now engaged in battles to attract each other's customers while retaining their own. Customers switching from one company to another is called **churn**.

You are asked to reveal the problem and to devise a solution. Your task is to device a precise, step-by-step plan for how your team should use MegaTelCo's vast data resources to decide which customers should be offered the special retention deal prior to the expiration of their contract.

Concrete programming on churn analysis can be found in the following links:

- `https://lukesingham.com/how-to-make-a-churn-model-in-r/`,
- `https://github.com/susanli2016/Data-Analysis-with-R/blob/master/customer_churn.Rmd`, etc.

### 1.7.4 Model Training and Prediction

Training / fitting a model with the preprocessed batch data can be complex. We need

$$\textbf{training algorithm}\Big(\textbf{batch data}, \textbf{model specification}\Big) \xrightarrow{?} \textbf{trained model}.$$

Note that the training algorithm **can fail** giving us nothing or a wrong model.

If the training is fine, we will get a well-trained model and the following are how R and Python implements the training process.

- R's training and prediction process:

```
lrmodel = lm(y ~ ., data=X)
# lm is the training algorithm for linear regression
# X  is the data
# y ~ . is the model specification linear regression
Yhat = predict(lrmodel, newdata=data.frame(x1=...,x2=...))
# We can put in new data and ask the trained model to predict
```

- Python's training and prediction process:

```
from sklearn.linear_model import LinearRegression
lrobj = LinearRegression()   # Set up the training algorithm
lrmodel = lrobj.fit(df.iloc[:,3:4],df.iloc[:,4])
                             # Do the training with data
newdata = pd.DataFrame({'x1':..., 'x2':...})
Yhat = lrmodel.predict(newdata)
```

Note that in reality, we have other training algorithms which we are not exploring:

- Event-driven training algorithms: With the proliferation of IoT devices, sensors, and applications emitting bytes around the clock, data scientists are faced with the task of prioritisation. The majority of data is insignificant and does not require a model to be retrained; however, when an **atypical** event does occur, AI can kick in and administer best next steps. This type of processing is valuable for businesses to automate things like inventory-control, or for AI to know when someone has arrived at home or departed.
- Real-time training algorithms: Useful when time is of the essence in realising value from the model. For example, a bank needs a fraud model to score credit card transactions within milliseconds to quickly deny likely fraudulent transactions.

### 1.7.5 Flexibility vs Interpretability

Interpretability is usually related relatively simple mathematical formulation such as the linear regression:

$$y = ax + b.$$

We can say that the input $x$ will influence $y$ at a rate $a$!

When the mathematical formulation becomes a bit more complex, like being quadratic

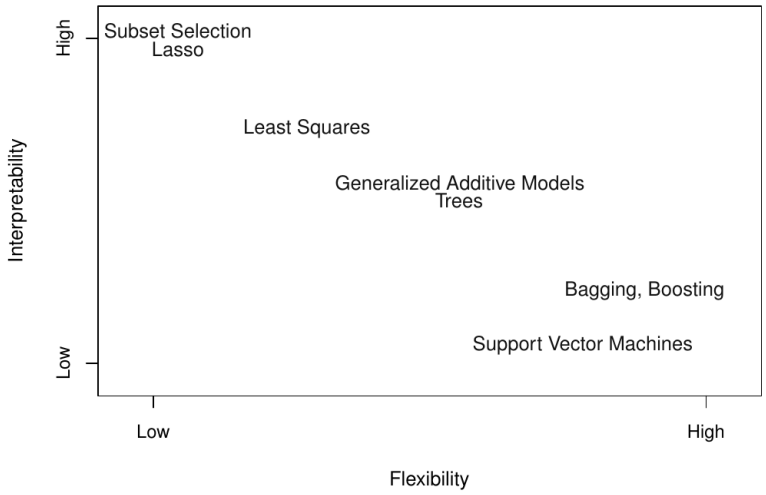
$$y = ax^2 + bx + c.$$

The input $x$ will influence $y$ at a quadratic manner when $|x| > 1$ but when $|x| \ll 1$, then $x$ will be influencing $y$ in a linear rate because $x^2 \approx 0$.

In general,

- when the predictive model is more nonlinear, it is more **flexible** — they can better fit data which are not on a line/hyperplane;
- when the predictive model has a less complex mathematical formulation, it is more **interpretable** — how each input $X_i$ affects the output $Y$ may be accounted from the mathematical formulation

Flexibility and interpretability are usually inverse to each other and the business requirement will decide on the right predictive model based on the balance of these two aspects. An illustration of some predictive models based on flexibility and interpretability are given below James et al. [2013]:




**Example 1.7.5** (Final Exam Jan 2019, Q5(a))**.** Discuss on the trade-off between model interpretability and prediction accuracy. (5 marks)

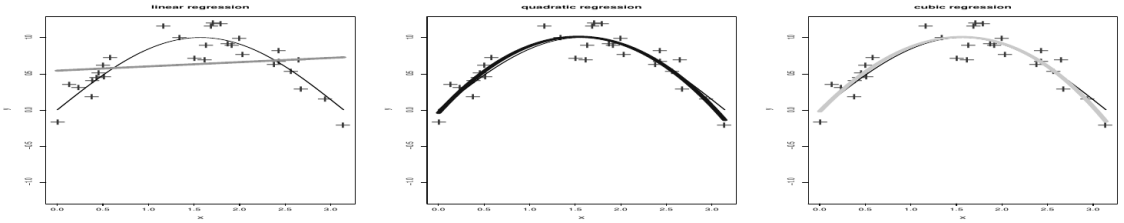
**Example 1.7.6.** Suppose the output $y$ is governed by the input $x$ following the equation:

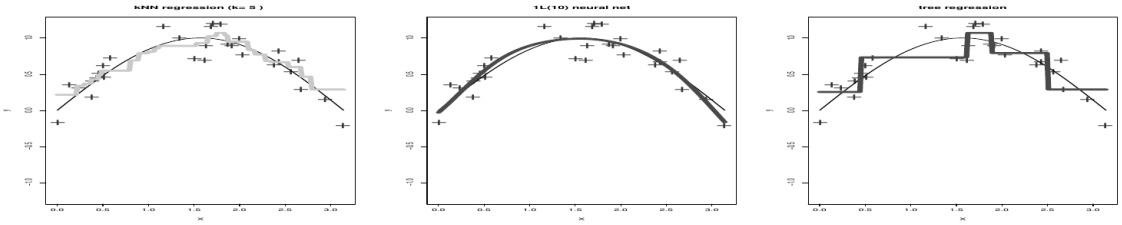
$$y = \sin(x) + R, \quad R \sim Normal(0, 0.2^2)$$

for the range $x \in [0, \pi]$. Try the following regression models:

- Linear regression: $y = ax + b + \epsilon$
- Quadratic regression: $y = a_2x^2 + a_1x + b + \epsilon$
- Cubic regression: $y = a_3x^3 + a_2x^2 + a_1x + b + \epsilon$
- kNN (Topic 2)
- Neural Network with 1 hidden layer 10 nodes ($= 1 \times 10 + 10 + 10 \times 1 + 1 = 31$) parameters
- Regression tree

and determine which model is the best based on the interpretability and flexibility.

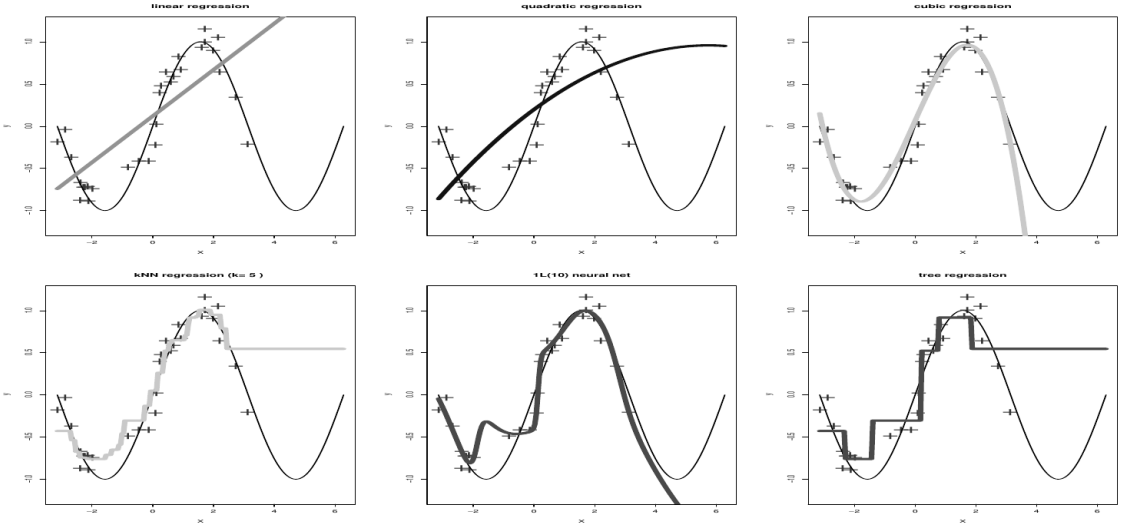





Things to consider: Flexibility (more parameters, model more complex) vs Interpretability (less parameters, model simpler)

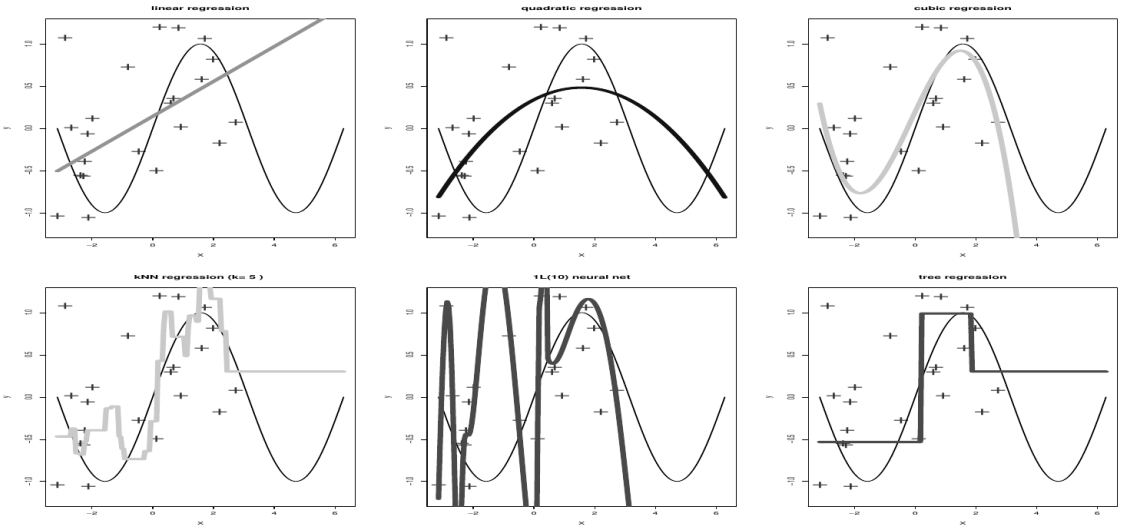
- Inflexible ⇒ Simpler math formula ⇒ Poorer Predictibility, better inference(?)
- Flexible ⇒ Complicated math formula ⇒ Good Predictibility, poorer inference(?) □

**Example 1.7.7** (Influence of the **domain**)**.** In the previous example, we have looked at the model for the range $x \in [0, \pi]$, now, let us look at $x \in [-\pi, 2\pi]$ with the same formula: $y = \sin(x) + R, \quad R \sim Normal(0, 0.2^2)$.




**Example 1.7.8** (Influence of the **noise**)**.** For a model with large 'noise' (i.e. $\sigma$ increases from 0.2 to 1.2):

$$y = \sin(x) + R, \quad -\pi \leq x \leq 2\pi, \quad R \sim Normal(0, 1.2^2).$$




Neural network is not performing well when the noise is large! □

## 1.8 Evaluation

**Unsupervised Learning**:

- There is **NO** standard measures for evaluation;
- For **pattern identification**: regular patterns (e.g. special shapes), cluster patterns, random pattern (particular probability distribution) are some special evaluation measures.

---

**Supervised Learning**:

- We have output $Y$ associated with an input $X$, so we can put in the input of an observation $\boldsymbol{x}_i$ to the predictive model $h(\boldsymbol{x}_i)$ and **compare** it to the actual observed label $y_i$;
- The 'theoretical' measurement of the **difference** between a true model $Y$ and a predictive model $h(X)$ is called a **generalisation error**.
- The evaluation metrics for supervised regression and supervised classification are listed below:

  **Regression:**

  - Residue Sum of Squares,

$$RSS = \sum_{i=1}^{n} \Big( y_i - f(x_1, \cdots, x_p) \Big)^2$$

  - R-squared is a statistical measure that represents the proportion of the variance for a dependent variable (the output) that is explained by an independent variable in (1.3), i.e.

$$R^2 = 1 - \frac{\sum (Y_{actual} - Y_{predicted})^2}{\sum (Y_{actual} - Y_{mean})^2}$$

  **Classification:**

  - Error rate: The "proportion of mistakes" that are made if we apply our estimate $\hat{f}$ to the training data:

$$\frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

    where $I$ is an indicator function where $I(true) = 1$ and $I(false) = 0$; $y_i$ is the actual class label while $\hat{y}_i = \hat{f}(\boldsymbol{x}_i)$ is the predicted class label.
  - Accuracy: The "proportion of correct prediction" that are made if we apply our estimate $\hat{f}$ to the validation data (see Section 1.8.2):

$$\frac{1}{n} \sum_{i=1}^{n} I(y_i = \hat{y}_i)$$

In R, the caret (Classification And REgression Training) package provides a good selection of performance measures. In Python, the `sklearn.metrics` provides standard performance measures Bowles [2015].

There are hundreds of standard test datasets that can be used to practice and get better at machine learning. Most of them are hosted for free on the UCI Machine Learning Repository (`https://archive.ics.uci.edu/ml/index.php`). These datasets are useful because they are well understood, they are well behaved and they are small. Some of them are available in CRAN's `datasets`, `mlbench`, `AppliedPredictiveModeling` [Kuhn and Johnson, 2013] and `ISLR` packages.

### 1.8.1 Performance Evaluation for Regression Problems

We will be using the estimation of the **regression problem**'s theoretical MSE (related to the empirical SSE above by MSE=SSE/$n$):

$$\mathbb{E}[(f(X)+\epsilon-h_D(X))^2]$$

as an example for accessing model accuracy.

Note that MSE is a statistic for measuring the difference between the **true/theoretical model**

$$Y=f(X)+\epsilon$$

and the **predictive model**

$$\hat{Y}=h_D(X).$$

The problem we are facing in assessing model accuracy is this: we **don't know the true model**

$$Y=f(X)+\epsilon$$

we only have the data (from the true model):

$$y_i=f(\boldsymbol{x}_i)+\epsilon_i,\quad i=1,2,...,n.$$

If we use all data $(\boldsymbol{x}_i,y_i)$, $i=1,2,...,n$ for training a predictive model $h_D(X)$ and if we use the mean squared error to **pick** the models, we may pick the models that **overfit to the noise**.

**Example 1.8.1** (Calculating the **Empirical Regression Error**)**.** Given the data with

| $X$ | Actual $Y$ | Prediction $\hat{Y}$ |
|---|---|---|
| 4.21 | 11.64 | 10.96 |
| 2.85 | 7.47 | 7.77 |
| 2.13 | 5.96 | 6.09 |
| 0.69 | 2.36 | 2.71 |
| 0.82 | 3.21 | 3.02 |
| 4.68 | 10.80 | 12.04 |
| 2.69 | 7.15 | 7.38 |
| 4.99 | 13.11 | 12.77 |
| 2.39 | 7.38 | 6.71 |
| 3.87 | 10.52 | 10.15 |

Calculate the MSE, MAE and $R^2$ error.

**Solution**: The **default** empirical regression error is usually the mean square error that we are familiar with:

$$\begin{aligned}MSE&=\frac{1}{10}((11.64-10.96)^2+...+(10.52-10.15)^2)\\&=0.30231\end{aligned}$$

Occationally, empirical MAE will be used:

$$\begin{aligned} MAE &= \frac{1}{10}(|11.64 - 10.96| + ... + |10.52 - 10.15|) \\ &= 0.453 \end{aligned}$$

The empirical $R^2$ error calculation is

$$\begin{aligned} R^2 &= 1 - \frac{(11.64 - 10.96)^2 + \cdots + (10.52 - 10.15)^2}{(11.64 - 7.96)^2 + \cdots + (10.52 - 7.96)^2)} \\ &= 1 - \frac{3.0198}{113.8392} = 0.9734731 \end{aligned}$$

where the means of the actual data and prediction are both 7.96.

**Example 1.8.2** (May 2022 Semester Final Exam, Q1(d))**.** Write down the mathematical formulas for two popular empirical accuracy measures associated with the generalisation error of regression problems and comment on the range of the two measures when the regressor fits the theoretical model well and when the regressor fits the theoretical model poorly. (5 marks)

### 1.8.2 Performance Evaluation for General Classification Problems — Confusion Matrix

The performance of a predictive model or a "classifier" can be checked based on their prediction accuracy. Suppose that we seek to estimate $f$ on the basis of training observations $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)\}$, where $y_1, \cdots, y_n$ are **qualitative**. The most common approach for quantifying the accuracy of the "estimated" predictive model $\hat{f}$ is the ***test error rate***, which is the proportion of mistakes that are made if we apply $\hat{f}$ to the testing observations

$$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} I(y_i \neq \hat{y}_i) \tag{1.4}$$

where $n_{test}$ is the number of test observations. An "acceptable" classifier is one for which the test error (1.4) is smallest.

The ***confusion matrix*** (or ***contingency table***) is a matrix or table that summarises the number of correct and incorrect classification.

**Example 1.8.3.** Suppose that 75% of the iris data is used as training data and the remainder 25% iris data is used as testing data. For a particular sample, the following confusion matrix is obtained using the kNN($k = 5$) predictive model:

| | **actual** | | |
|---|---|---|---|
| **predicted** | setosa | versicolor | virginica |
| setosa | 14 | | |
| versicolor | | 10 | 1 |
| virginica | | 1 | 12 |

Using an R script, we have the following test result:

```
             | iris_test[, M]
iris_predict |     setosa | versicolor |  virginica |  Row Total |
-------------|------------|------------|------------|------------|
      setosa |         14 |          0 |          0 |         14 |
             |      1.000 |      0.000 |      0.000 |      0.368 |
             |      1.000 |      0.000 |      0.000 |            |
             |      0.368 |      0.000 |      0.000 |            |
-------------|------------|------------|------------|------------|
  versicolor |          0 |         10 |          1 |         11 |
             |      0.000 |      0.909 |      0.091 |      0.289 |
             |      0.000 |      0.909 |      0.077 |            |
             |      0.000 |      0.263 |      0.026 |            |
-------------|------------|------------|------------|------------|
   virginica |          0 |          1 |         12 |         13 |
             |      0.000 |      0.077 |      0.923 |      0.342 |
             |      0.000 |      0.091 |      0.923 |            |
             |      0.000 |      0.026 |      0.316 |            |
-------------|------------|------------|------------|------------|
Column Total |         14 |         11 |         13 |         38 |
             |      0.368 |      0.289 |      0.342 |            |
-------------|------------|------------|------------|------------|
```

In each cell,

- the first number is the count with 'prediction vs actual';
- the second number is the ratio of count against prediction class count;
- the third number is the ratio of count against actual class count;
- the last number is the percentage of the count out of total number of data.

**Example 1.8.4** (May 2022 Semester Final Exam, Q1(b))**.** Assuming the inputs of the data are all categorical and the output has **three classes**.

1. Give an example of the nonparametric discriminative supervised learning model which can handle three classes. (2 marks)

2. Give an example of the parametric generative supervised learning model which can handle three classes. (2 marks)

3. For the confusion matrix below:

| | Actual | | |
|---|---|---|---|
| Prediction | A | B | C |
| A | 44 | 3 | 0 |
| B | 5 | 50 | 0 |
| C | 0 | 7 | 34 |

calculate the accuracy of the confusion matrix. (2 marks)

### 1.8.3 Performance Evaluation for Binary Classification Problems

For a classification problem with binary outcomes (only 2 classes), positive (+) and negative (-), the confusion matrix can be presented as follows.

| | | **Actual** examples | |
|---|---|---|---|
| | | Positive (+) | Negative (-) |
| **Predicted** | Positive (+) | True Positive Count (TP) | False Positive Count (FP) |
| | Negative (-) | False Negative Count (FN) | True Negative Count (TN) |

where the output of the classification problem is assumed to be two classes, positive (+) and negative (-), the predicted class fitted by model and the true class of the validation set are shown below:

| *Predicted Class* | *True Class* | *Counted as* |
|---|---|---|
| Positive (+) | Positive (+) | True Positive (TP) |
| Positive (+) | Negative (-) | False Positive (FP) |
| Negative (-) | Positive (+) | False Negative (FN) |
| Negative (-) | Negative (-) | True Negative (TN) |

Various accuracy measures associated with the confusion matrix are given by the following formulae:

Accuracy (ACR) $$\text{ACR} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

Sensitivity / (Positive) Recall / True Positive Rate (TPR), $$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Specificity / Negative Recall / True Negative Rate (TNR), $$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

Positive Predictive Value (PPV) / Positive Precision, $$\text{PPV} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Negative Predictive Value (NPV) / Negative Precision, $$\text{NPV} = \frac{\text{TN}}{\text{TN} + \text{FN}}$$

False Positive Rate (FPR) / Probability of False Alarm, $$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} = 1 - \text{TNR}$$

False Negative Rate (FNR), $$\text{FNR} = \frac{\text{FN}}{\text{TP} + \text{FN}} = 1 - \text{TPR}$$

Merging the accuracy measures to the confusion matrix leads to a larger table below.

| | | Actual examples | | |
|---|---|---|---|---|
| | | Positive (+) | Negative (-) | Precision |
| Predicted | Positive (+) | True Positive Count (TP) | False Positive Count (FP) | Positive Predictive Value (PPV) |
| | Negative (-) | False Negative Count (FN) | True Negative Count (TN) | Negative Predictive Value (NPV) |
| | Recall | True Positive Rate (TPR) (Sensivity) | True Negative Rate (TNR) (Specificity) | Accuracy (ACR) |

Precision is how certain we are of the true positives. Sensitivity is how certain we are that we are not missing any positives. The rationale behind the measures is as follows:

- Choose sensitivity if the occurrence of false negatives is unacceptable/intolerable. For example, in the case of diabetes that we would rather have some extra false positives (false alarms) over saving some false negatives.

- Choose (positive) precision if we want to be more confident of the true positives. For example, in case of spam emails, we would rather have some spam emails in our inbox rather than some regular emails in our spam box. We would like to be extra sure that email X is spam before we put it in the spam box.

- Choose specificity if we want to cover all true negatives, i.e. meaning we do not want any false alarms or false positives. For example, in case of a drug test in which all people who test positive will immediately go to jail, we would not want anyone drug-free going to jail.

**Example 1.8.5** (Scenario 1: The Fraudulent Loans)**.** A Malaysian bank operates a large personal loan business. This product helps their customers enjoy better cash liquidity, e.g. purchase big ticket items or take a family holidays. However, each lending business comes with a risk of revenue losses due to credit default (i.e. who failed to repay the loan).

The bank must assess the customer's payment behaviour when customer apply a loan product and make a final approve/reject decision. You have been asked to identify which customers have higher tendency to miss loan repayments so that the company could make a decision.

A predictive model has been built by using the training set. After predicting the outcome (fraud, not fraud) by implementing the model into validation set, the results are recorded as follow:

- Numbers of customer predicted to be fraud and the prediction is correct = 70
- Numbers of customer predicted to be fraud and the prediction is incorrect = 30
- Numbers of customer predicted not to be fraud and the prediction is correct = 80
- Numbers of customer predicted not to be fraud and the prediction is incorrect = 20

| | | True Class | |
|---|---|---|---|
| | | Fraud (+) | Not Fraud (-) |
| Predicted Class | Fraud (+) | 70 (TP) | 30 (FP) |
| | Not Fraud (-) | 20 (FN) | 80 (TN) |

Calculate the accuracy measures sensitivity, specificity, PPV, NPV, ACR, FPR, FNR.

**Solution**: The accuracy measures are calculated as follows.

- Sensitivity / True Positive Rate, TPR $= \dfrac{70}{70+20} = 0.7778 = 77.78\%$
- Specificity / True Negative Rate, TNR $= \dfrac{80}{30+80} = 0.7273 = 72.73\%$
- Positive Predictive Value, PPV $= \dfrac{70}{70+30} = 0.7 = 70\%$
- Negative Predictive Value, NPV $= \dfrac{80}{20+80} = 0.8 = 80\%$
- Accuracy $= \dfrac{70+80}{70+30+20+80} = 0.75 = 75\%$
- False Positive Rate, FPR $= 1 - 72.73\% = 27.27\%$
- False Negative Rate, FNR $= 1 - 77.78\% = 22.22\%$

### 1.8.4 More Performance Evaluations for Binary Classification Problems

- Kappa: a statistics accuracy measure that takes the base distribution of classes into account, see `https://stats.stackexchange.com/questions/82162/cohens-kappa-in-plain-english`
- **Receiver operating characteristic (ROC curve)**: It is a graphical plot that illustrates the diagnostic ability of a **binary classifier** system as its discrimination threshold is varied. The ROC curve is created by plotting the **sensitivity** against the **false positive rate (FPR)** ("1 − specificity") at various threshold settings. ROC analysis provides tools to select possibly optimal models — the best model has the **largest area under the curve (AUC)**.
- Log(arithmic) loss (related to cross-entropy): It measures the performance of a classification model where the prediction input is a probability value between 0 and 1.

**Example 1.8.6.** Apply the ROC analysis on the ISLR's Smarket data using the `pROC` package.

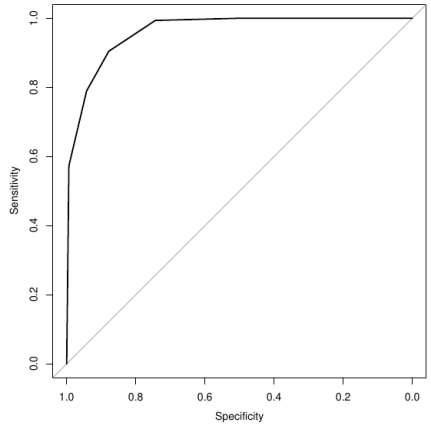
**Solution:**

```
1 library(ISLR)
2 Smarket = Smarket[,-1]  # Remove Year
3 N = nrow(Smarket)
4
```

```
set.seed(59) #set.seed(9)
train_idx = sample(seq(N), size=0.75*N)
Smarket_train = Smarket[ train_idx,]
Smarket_test  = Smarket[-train_idx,]

library(class)  # for knn
M = ncol(Smarket)
Smarket_predict = knn(train=Smarket_train[,-M], test=Smarket_test[,-M],
                  cl=Smarket_train[,M], k=5, prob=TRUE)
suppressWarnings(library(pROC))
prob = attr(Smarket_predict,"prob")
prob = ifelse(Smarket_predict=="Up", prob, 1-prob)
proc.obj = roc(Smarket_test[,M], prob, plot=TRUE)
```

The output is shown below.




**Example 1.8.7** (May 2022 Semester Final Exam, Q1(c))**.** Given the confusion matrix of a 70 testing data for a predictive model of the inflammation of urinary bladder diagnostic with a response variable of values "no" (positive) and "yes" (negative) in Table 1.3.

| | Actual | |
|---|---|---|
| Prediction | No | Yes |
| No | 27 | 12 |
| Yes | 8 | 23 |

Table 1.3: Confusion matrix.

Calculate the following statistical measures for evaluating the performance of the predictive model.

1. Specificity (2 marks)

2. Negative Predictive Value (NPV) (2 marks)

3. F1 score, $F1 = \dfrac{TP}{TP + \frac{1}{2}(FP + FN)}$ (2 marks)

4. Accuracy and Kappa Statistic

$$\text{Kappa} = \frac{\text{Accuracy} - \text{RandomAccuracy}}{1 - \text{RandomAccuracy}}$$

where

$$\text{RandomAccuracy} = \frac{(\text{TN+FP}) \times (\text{TN+FN}) + (\text{FN+TP}) \times (\text{FP+TP})}{(\text{Total Number of Test Data})^2}.$$

(2 marks)

There are a few libraries in R, such as `caret` (providing `confusionMatrix`) and `gmodels` (providing `CrossTable`), for generating confusion matrix. In Python, under `sklearn.metrics`, `accuracy_score`, `precision_score`, `recall_score` are available for computing various scores while `confusion_matrix` can be used to generate confusion matrix.

The R's `caret` package contains functions `R2(predicted, observed)` and `R2(predicted, observed)` for calculating $R^2$ and the root mean squared error (RMSE) values respectively. The `pROC` package can create the ROC curve and derive associated statistics [Robin et al., 2011]. In Python, it is supported by `sklearn.metrics`'s `roc_curve(trueY, scoreY)`, `roc_auc_score`, `precision_recall_curve`.

### 1.8.5 The Holdout / Validation Set Approach

If we have a data $D$, then we use $D$ to train a predictive model $h_D$. This predictive model $h_D$ may be 100% true on $D$. But how to we know that it does not include the noise and overfit?

Why is overfitting a problem?

- An overfitting model works extremely well on **historical data** but may perform terribly with new data when dealing with new data. For example, an email spam filter that works well with known spam/ham but when it sees new spam emails, it treats them as ham, then users may receive a lot of spam mixing with ham which is annoying.
- An overfitting model is said to be **not generalising**, i.e. the model will still **provide a reasonable prediction on an unseen data** rather than providing a wrong prediction based on the overfitted noise.

The ***holdout method***, ***split validation*** or ***validation set approach*** [Coelho and Richert, 2015, Chapter 2], is the most basic strategy to estimate the accuracy with fitting a particular statistical learning method on a set of observations [James et al., 2013, Section 5.1]. It involves randomly dividing the available set of observations into two parts:

- Training set — to build/fit the model
- Validation/Test set — to test/evaluate the fitted model

A schematic display of the validation set approach is shown in the following figure.


A set of $n$ observations are randomly split into a training set (the box containing observations 7, 22 and 13, among others) and a validation set (the box containing observation 91, among others).

The model which is built from training set is then used to predict the outcomes of the observations in validation set, and the performance is evaluated.

**Example 1.8.8** (Linear Sampling)**.** Consider a 4-dimensional data $D$:

All Historical Data, D

| Index | X1 | X2 | X3 | X4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |

h_D(X1,...,X4)

If we train a predictive model with the all the data $D$, some predictive models give us **empirical error** of 0.

This implies that **noise** $\epsilon_i$ is also accepted into the model $h_D$ and when we apply the trained model to new data, the zero-empirical-error model will produce bad predictions nearly 100% of the time!

To prevent overfitting problem, the **holdout method** (or **validation set approach**, or train/test split method) is used: some data (10% to 50%, typically 30%) are hold out for testing.

Holdout for Training, D1

| Index | X1 | X2 | X3 | X4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |

h_D1(X1,...,X4)

Holdout for Testing, D2

| Index | X1 | X2 | X3 | X4 |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |

The predictive model $h_{D_1}(X)$ trained on data $D_1$ can be tested against the 'unseened data' $D_2$ to $h_{D_1}$. So if the error is small, we can be a bit more confident that the model $h$ didn't fit the noise too much. □

Validation set (Linear Sampling) approach in R

```
library(datasets)
set.seed(0)
test.index = sample(1:nrow(iris), size=0.4*nrow(iris))
X_y.test  = iris[ test.index, ]
X_y.train = iris[-test.index, ]
library(e1071)
```

```
clf = svm(Species ~ ., data = X_y.train, kernel='linear')
predicted = predict(clf, newdata=X_y.test)
conftbl   = table(predicted, X_y.test$Species)
# Accuracy of prediction
sum(diag(conftbl))/sum(conftbl)
```

Validation set (Linear Sampling) approach in Python

```
from sklearn.model_selection import train_test_split
from sklearn import datasets, svm
X, y = datasets.load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(
         X, y, test_size=0.4, random_state=0)
clf_obj = svm.SVC(kernel='linear', C=1)
clf = clf_obj.fit(X_train, y_train)
# Accuracy of prediction (see Confusion matrix)
clf.score(X_test, y_test)
```

When we have multiple "predictive" models, we may need to split the data into three sets, one training set, one validation set (which is used to validate one particular model), and one common test set for all different models to compare.

**Example 1.8.9** (Final Exam Jan 2019, Q3)**.** (a) A predictive model can be built when historical data with known response are presented. The predictive model is then used to predict the response of a new data set with predictors given. Figure Q3(a) shows the process to form a predictive model.

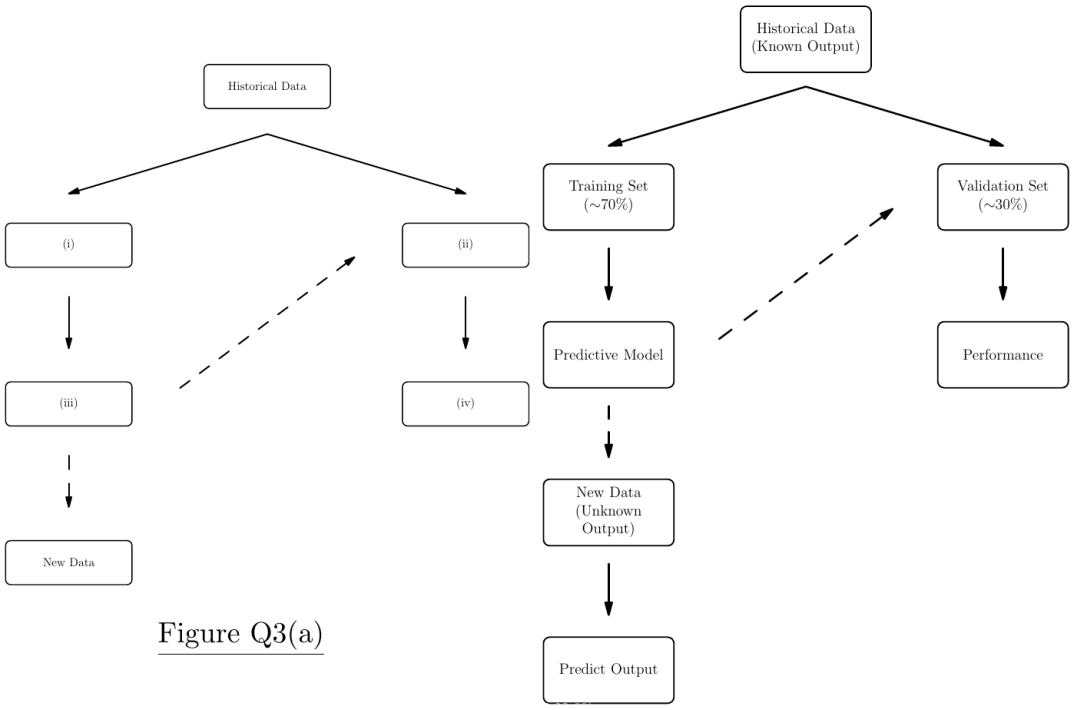



Figure Q3(a)

Fill in the blanks (i) to (iv) in Figure Q3(a). State the differences between regression and classification for each step in the process of forming a predictive model. (12 marks)

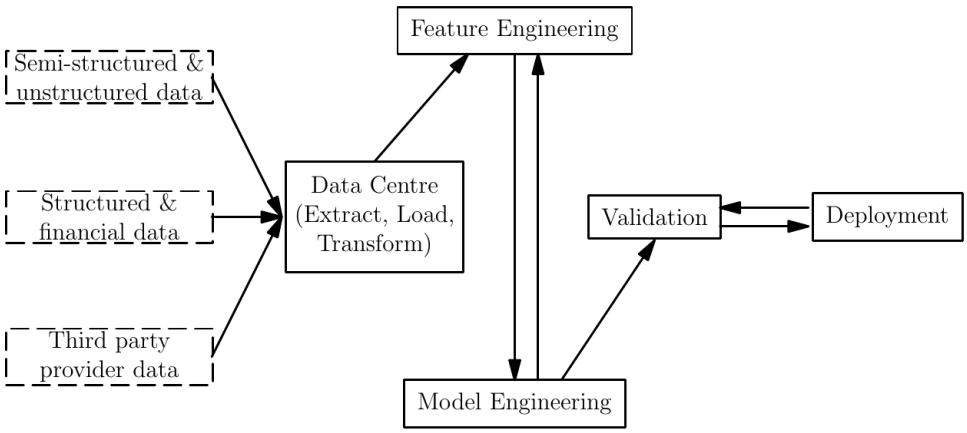
(b) Give three examples on how statistical learning can help in risk/fraud analytics. (3 marks)

## 1.9 Deployments

According to SAS (see `https://www.sas.com/en_ae/insights/analytics/predictive-analytics.html`), predictive analytics are deployed by companies to tackle the following issues:

- **Detecting fraud**. Combining multiple analytics methods can improve pattern detection and prevent criminal behaviour. As cybersecurity becomes a growing concern, high-performance behavioural analytics examines all actions on a network in real time to spot abnormalities that may indicate fraud, zero-day vulnerabilities and advanced persistent threats.
  - Banking industry: credit scores to predict individual's delinquency behaviour and evaluate the credit worthiness of each applicant.
  - Insurance industry: changes of an event (accident/disease) to calculate premium.
  - Financial institutions like Online Payment Gateway companies to analyse if a transaction was genuine or fraud.
- **Optimising marketing campaigns**.
  - Predictive analytics are used to determine customer responses or purchases, as well as promote cross-sell opportunities.
  - Predictive models help businesses attract, retain and grow their most profitable customers.
  - Demographic studies, customer segmentation and other techniques allow marketers to use large amounts of consumer purchase, survey and panel data to understand and communicate marketing strategy.
- **Improving operations**. Many companies use predictive models to forecast inventory and manage resources. Airlines use predictive analytics to set ticket prices. Hotels try to predict the number of guests for any given night to maximise occupancy and increase revenue. Predictive analytics enables organisations to function more efficiently.
- **Reducing risk**. Credit scores are used to assess a buyer's likelihood of default for purchases and are a well-known example of predictive analytics. A credit score is a number generated by a predictive model that incorporates all data relevant to a person's creditworthiness. Other risk-related uses include insurance claims and collections.

According to a report from Bank of England [2019], the survey shows that approximately 2/3 of the UK financial services are using "predictive modelling" in some form. The deployment is most advanced in the banking and insurance sectors. Predictive modelling is most commonly used in anti-money laundering (AML) and fraud detection as well as in customer-facing applications (e.g. customer services and marketing). Some firms also use ML in areas such as credit risk management, trade pricing and execution, asset management, as well as general insurance pricing and underwriting. Firms mostly design and develop applications in-house. However, they sometimes rely on third-party providers for the underlying platforms and infrastructure, such as cloud computing. The report shows an outline of the "modelling process" as follows.




Human-beings have tried to automate "data analysis" using computer since 1642 where Pascal tries to build a calculator (see `https://en.wikipedia.org/wiki/Pascal%27s_calculator`). It was until lately we have achieved better image classifications, playing chess and go. However, many human activities such forecasting job performance, predicting criminal recidivism, etc. are still beyond the reach of machine [Narayanan, 2019]. "Predictive Modelling" or "machine learning" is not a key to solving everything by automation as pointed out by Narayanan [2019].

Apart from the classical domains mentioned above, data mining or predictive modelling is also deployed in:

- Information extraction: search documents using keywords
  - Compression of data
  - Querying using regular expressions and predicates (e.g. find documents related to the applications of calculus)
- Bioinformatics: try to determine the living species based on the (complete or fragment of) genetic information
- Recommender system: Recommend movies or musics to users with similar preferences
  - MovieLens Datasets (`https://grouplens.org/datasets/movielens/`)
  - Yahoo Movie / Music Ratings Datasets (`https://webscope.sandbox.yahoo.com/catalog.php?datatype=r`)
- Supermarket item placement (association rules) and logistics
  - `https://cran.r-project.org/web/packages/arules/index.html`

– `https://cran.r-project.org/web/packages/arulesCBA/index.html`

**Example 1.9.1** (Final Exam Jan 2018, Q5(a))**.** Describe three real-life applications for each of the following statistical learning setting: Classification; Regression; Unsupervised learning. (9 marks)