Topic 1: Overview

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

Terminologies with similar meaning:

Statistical learning refers to the application of mathematical statistics in the analysis of supervised and unsupervised models.

Machine learning refers to the use of machine-based methods which can learn from data. It cares more about heuristics rather than statistical theory compare to statistical learning.

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

- Supervised Learning Models:
  - Classifiers: kNN ✓, logistic regression ✓, neural network ✗, naive Bayes ✓, discriminant analysis (LDA & QDA) ✓, linear support vector machine (SVM) ✗, kernel SVM ✗, classification trees ✓, ...
  - Regressors: kNN ✓, linear regression ✓, regression tree ✓, support vector regressor (SVR ✗), ...
- Unsupervised Learning Models:
  - Feature agglomeration ✗
  - Random projections ✗
  - Dimensional Reduction: PCA ✓, t-SNE ✗, ...
  - Clustering: k-Means ✓, Hierarchical clustering ✓, Spectral clustering ✗, 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`).
  - Optimisation and Control ✗: Linear control, genetic algorithms, deep model predictive control, estimation of distribution algorithms, evolutionary strategies
  - Reinforcement learning ✗: Value-based, Policy-based, Model-based; Q-learning, Markov decision processes, deep reinforcement learning.

    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)
- Self-supervised learning ✗
- Structured data data mining ✗: pattern-based similarity search, sequence/sequential pattern mining (mines time series data (e.g. sales in a month, calls per day, web visits per hour, etc.), episode (subseqence of events) mining, graph mining, molecule mining, etc.

1.1 Software, Data and References

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. The learning of predictive models will require the use of data programming languages such as R, Python, etc.

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

- R + glmnet + tree + rpart + gbm + ... → `https://cran.r-project.org/` [Chambers, 2008], [Spector, 2008]
- Python + Pandas + Matplotlib + Google Tensorflow + PyTorch + ... → UV (`https://docs.astral.sh/uv/`) or Anaconda Python (too large and can be corrupted when downloading, if it is not working, you may need to uninstall, download and re-install again).
- Java + Weka + SparkML + ...
- Excel, Power BI, Microsoft Access, SQL, STATA, SAS, SPSS in business environment

Popular data:

- UC Irvine Machine Learning Repository (`https://archive.ics.uci.edu/`)
- Kaggle (`https://www.kaggle.com/datasets`, need registration)
- Prof. Julian McAuley has gathered datasets associated with online shopping and social network platforms at `https://cseweb.ucsd.edu/~jmcauley/datasets.html`.
- CRAN's `datasets`, `mlbench`, `AppliedPredictiveModeling`, [Kuhn and Johnson, 2013] and `ISLR` packages.

Textbooks:

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, An Introduction to Statistical Learning: with Applications in R, 2nd ed., Springer 2021
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, 2008

1.2 Learning Outcomes and Assessment

Learning Outcomes:

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.

WBLE is too annoying to use with the Captcha. Materials for the course:

- Lecture Notes, Tutorials, Practical Scripts, Announcements: `https://liaohaohui.github.io/UECM3993/`
- MS Teams Code: `ts5oqqt` (We will be using the **2026.02 ... Channel** rather than

the General channel)

Syllabus (To be updated):

- Lecture (22 hours) : affected by Thaipusam (Week 1), Chinese New Year (Week 3) and Hari Raya Puasa (Week 8).
- Tutorial (11 hours) : affected by Chinese New Year (Week 3).
- Practical (12 hours) : affected by Chinese New Year (Week 3).

Class Arrangements:

Week 1:
- Start of tutorial and practical (we try to complete class by Week 12/13)
- Week 1 Monday (2 Feb 2026) is a Thaipusam replacement holiday, a pre-recorded video will be prepared to replace the lecture. Remember to scan the QR code on Tuesday (3 Feb 2026) before 5 pm for the "replacement lecture".
- Start to form assignment group, 4–7 members in a group. Individuals without any group by week 3 may be assigned to groups with less than 7 members by lecturer.

Week 2:
- Hopefully the assignment is ready for review.

Week 3:
- Chinese New Year holidays. Lecture, tutorial and practical classes will be pre-recorded. Remember to scan the QR code on Monday (16 Feb 2026) before 5 pm for all replacement classes before enjoying your Chinese New Year.
- Everyone in class should be a member of an assignment group. Lecturer will assign anyone who has not formed a group to groups with least members.

Week 4:
- **Assignment** starts. Choose a group leader wisely because the group leader will be writing down your contribution to the assignment report.

Week 6:
- **Quiz** covering practical 1 to 3, topics 1 to 3 during lecture.

Week 8:
- Monday (23 Mar 2026) is a replacement for Hari Raya Puasa holiday. Remember to scan the QR code for the replacement lecture before 2 pm Friday.

  A pre-recorded video is prepared for the lecture class. If Tuesday is not a public holiday, practical classes will resume as usual.

Week 11:
- Wednesday (15 Apr) 6pm: Deadline for the submission of an assignment report in pdf or word format and computer program(s) in plain text format to lecturer by group leader.
- All tutorials should be completed by this week.

Week 12:
- Physical Oral presentation

Week 13:
- Physical Oral presentation will be continued if there are more than 6 groups.

Week 14:
- No more classes (self-study for final exam).
- Assignment markings should be out by Friday (if not, it would be in Week 15). Check the coursework marks (in MS Teams's 2026.02 Channel) and inform the lecturer if there are any mistakes.

Assessments:

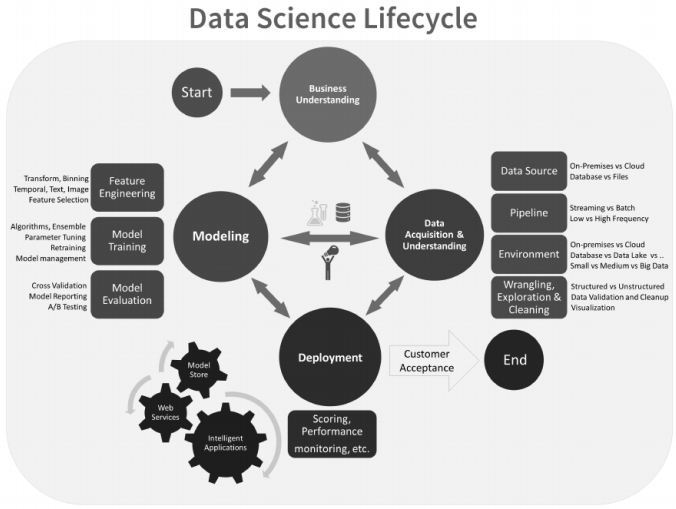
- **Practical Quiz** (12%, CO4):
  - A sample quiz from past semester is given.
- **Assignment** (38%):
  - Report 18% (CO1 + CO2 + CO3)
  - Programming Code 10% (CO4)
  - Oral Presentation 10% (CO5: need to submit presentation slides)
- **Final Exam** (100 marks $\times 0.5 \Rightarrow 50\%$): 3 + 1 Questions.
  - Q1: CO1, Supervised + Unsupervised, 25 marks
    - mainly Topic 1 and some **conceptual questions** from other topics (e.g. k-fold CV from Topic 2, neural network from Topic 3, discriminative vs generative models from Topic 4, bias-variance theory from Topic 6, ensemble methods from Topic 7 etc.).
  - Q2: CO2, Supervised Learning Models, 25 marks
    - kNN(?), LR (may include MLR and ANN in near future), LDA, QDA, NB, tree (and occasionally conceptual questions on tree ensembles)
  - Q3: CO3, Unsupervised Learning Models, 25 marks
    - PCA, K-means, hierarchical clustering (single, complete and average linkages and dendrogram)
  - Q4/Q5: CO5, Supervised + Unsupervised, 25 marks
    - a mixed of Q2 and Q3.

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]:

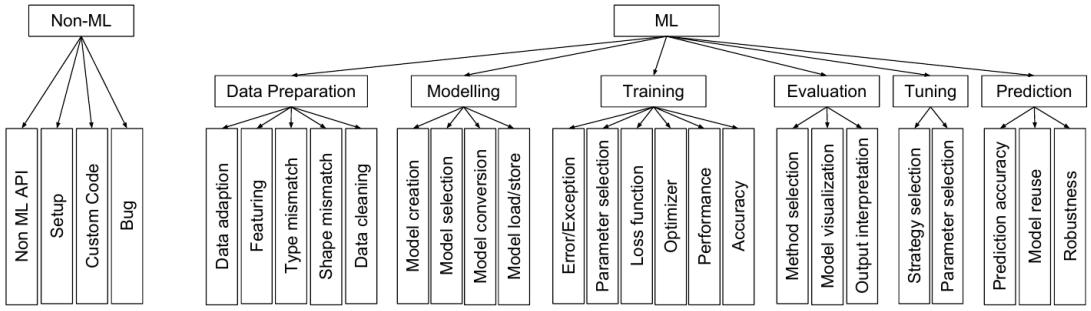
1. **Business Understanding**
   - Focuses on understanding the project objectives and requirements from a business perspective, and converts into a data mining problem definition and a preliminary plan.
2. **Data Understanding**
   - Starts with an initial data collection and proceeds with activities 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.

Remark 1.3.1. Most redictive modelling frameworks are similar to CRISP-DM with some variations, for example, the Microsoft Data Science Lifecycle:

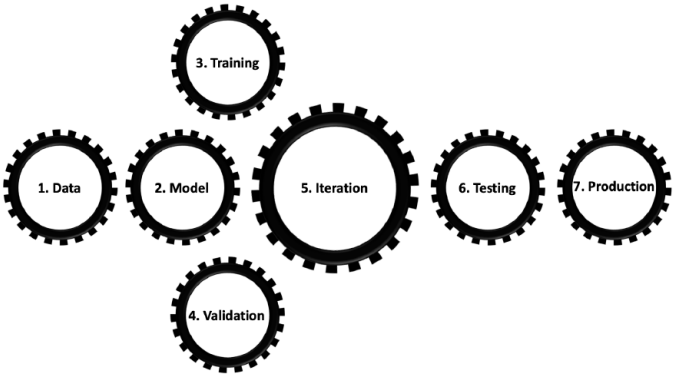


https://docs.microsoft.com/en-us/azure/machine-learning/team-data-science-process/lifecycle-business-understanding

Islam et al. [2019] uses the following tree diagram for machine learning:



Raissi [2023] uses the following big picture:



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: database design based on tabular relations. Microsoft SQL PostgresQL, MariaDB, OracleDB, MySQL (owned by Oracle), etc. are popular SQL servers.
  - NoSQL: database design that focuses on providing a mechanism for storage and retrieval of data that is modelled in means other than the tabular relations used in SQL (`https://en.wikipedia.org/wiki/NoSQL`), DataLog, `https://en.wikipedia.org/wiki/SPARQL`, etc.
- Unstructured Data (EDA cannot be used):
  - Texts: Reports in Word/PDF; Twitters; etc.
  - Images (of different sizes and resolutions)
  - Songs / Lyrics
  - Video and multimedia files
  - Time series: Stock price; Online game control sequence; Industrial robot control sequence; mobile activities; etc.
  - Sensor data from Internet of Things (IoT) devices

1.5.1 Unstructured data

- Types of documents:
  - Office document: doc, docx (zip+xml), odt, ...
  - PDF: binary format with text annotations
  - Scanned documents
- Types of 'images' of various dimensions:
  - Photo images (can be of different sizes depending on the camera models). The colour images are usually compressed in various image formats such as jpeg, png, etc.
  - Biometric data: fingerprint, face, iris recognition
  - Night Vision using thermal imaging in EVs

  - Medical imaging: ECG, X-ray, ultrasound imaging, CT (computer tomography), MRI (magnetic resonance imaging), PET (positron emission tomography), etc.
  - Sonar imaging and Radar imaging

There is no simple answer to convert unstructured data to structured data. The various conversions of unstructured data lead to a field of study called feature engineering. Photo images and simple text can be transformed to 1D array of large feature space as shown below:

- Convert texts to 2D matrix with individual words as columns, rows as word counts using the **bag of word** representation. More powerful representations include the TF-IDF (Term Frequency-Inverse Document Frequency) and Google Word2vec.
- Convert (rescale) an image to a 2D matrix or a 3D array of a fixed shape.

The conversions of time-dependent data (e.g. human speech signals, musics, videos, etc.) to structured data are extremely complex involving time-frequency sampling.

1.5.2 Structured data

Structured data is those data which has clear organisation such as table, graph, etc. We will limit our scope to **tabular data** since `https://en.wikipedia.org/wiki/Exploratory_data_analysis` (EDA) using (i) numeric summary statistics; and (ii) data visualisation can be carried out.

The data type of each column of a tabular data is either

- Categorical (or nominal) data (e.g. Gender): R's `factor`; Python's `astype("category")`
- Ordinal data (e.g. Student grade): R's `ordered`
- Continuous/Numerical data (e.g. Temperature): R's `numeric`; Python's `float64`

Univariate data analysis:

- Numeric summary:
  - For categorical data (or some integral data): frequency and mode, e.g. R's `table()`.
  - For ordinal data: percentiles(?)
  - For continuous data: mean, median, quantiles, range, variance, skewness, etc. E.g. R's `mean()`, `median()`, `var()`, `sd()`, etc.
  - In general, R's `summary()`, Python's `df.describe()`, `scipy.stats.describe` provide automatic numeric summaries for each column of a tabular data based on their datatype.
- Data Visualisation:
  - For categorical and ordinal data: pie chart, barplot / bar graph, e.g. R's `plot()`
  - For numerical data: histogram, stem-and-leaf plot, boxplot, density plot, Q-Q plot E.g. R's `hist()`, `stem()`, `qqplot()` Python's `plt.hist()`), etc.

Bivariate data analysis (`https://en.wikipedia.org/wiki/Bivariate_analysis`):

- Numeric summary:
  - categorical vs categorical: `table(dataA,dataB)`, etc.
  - categorical vs numerical: `aggregate(numData, by=list(col=catData), mean)`
  - numerical vs numerical: `cor(x, y, method="pearson")`, `cor.test(x, y)` (for

Hypothesis testing)

- Data visualisation:
  - categorical vs categorical: `barplot(table(catA,catB))`, `plot( catA, catB)`
  - categorical vs numerical: `boxplot(numB catA, d.f)`, `plot(catA,numB)`
  - numerical vs numerical: `pairs(d.f)`, `heatmap(cor(d.f))`, `pheatmap(cor(d.f),display_numbers=T,fontsize=12)`

1.6 Data Preparation / Preprocessing

Data preparation/preprocessing is a process to

- If the data is unstructured (Section 1.5.1), convert it a **structured** data;
- If the data is structured (Section 1.5.2), perform cleaning, imputation, feature scaling, dimensional reduction, etc.
  - 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

In classical predictive models, we need to transform list of texts to tabular data for processing.

Warning: Modern neural-network based models do not required the transformation of text to structured data.

- For scanned documents (which can be PDF format or tiff format), perform OCR (optical character recognition) to obtain the text (with figures ignored).
- For PDF documents with text annotations, extract the text.
- For Office documents, use appropriate programming libraries to extract the text.

Once we obtain the text, we can perform some preprocessing steps from natural language processing (NLP) toolkits which may involve:

- 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

In classical predictive models, image will normally be scaled and flattened into a high dimensional feature space.

Warning: Modern neural-network based models may require users to scale the images before uploading. However, modern models may scale an image to standard size before working on the image.

Image preprocessing steps:

- Deblurring, denoising, contrast enhancement, ...
- Cropping
- Rescaling (with Gaussian smoothing)
- Projective transforms: translation and rotation
- (Optional for classical predictive models) Reduce a coloured image (3D array) to a gray scale image (2D array)

Image preprocessing software libraries:

- R: `magick` uses the C++ ImageMagick library for image processing;
- Python:
  - imageio + 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 Categorical Data

Some predictive modelling software require the user to encode the categorical data and ordinal data to numeric data. We only list the most commonly used steps below.

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 (non-standard) encodings: `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 Numeric Data

Some predictive modelling software require the user to scale the numerical data to reduce rounding errors or for a more stable training process.

Popular preprocessing steps for numeric data:

- **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: R's `poly()`
- Non-linear and custom transformation
- Dimensional reduction
- Discretization: R's `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) Examples

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) - apply(X,2,min))/(apply(X,2,max)-apply(X,2,min)))
```

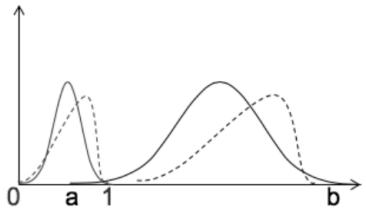
In Python, we can use numpy array with

```
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.



Example 1.6.1 (Final Exam Jan 2024 Sem, Q4(a)(i); Tutorial 6)**.** Given the training data with three numeric features "bill length" (unit: mm), "bill depth" (unit: mm), "flipper length" (unit: mm) and the label "species" in Table 4.1.

Table 4.1: Training data of the penguin data with three different labels of penguins — Adelie, Chinstrap and Gentoo.

Obs.	bill length	bill depth	flipper length	species
A	41.1	19.1	188	Adelie
B	35.9	19.2	189	Adelie
C	36.0	17.9	190	Adelie
D	43.4	14.4	218	Gentoo
E	50.0	15.2	218	Gentoo
F	44.5	14.7	214	Gentoo
G	50.6	19.4	193	Chinstrap
H	45.7	17.0	195	Chinstrap

Write down the min-max scaling for all the features in Table 4.1 which transform Table 4.1 to Table 4.2.

Table 4.2: Scaled training data from Table 4.1

Obs.	bill length	bill depth	flipper length	species
A	0.3537	0.94	0.0000	Adelie
B	0.0000	0.96	0.0333	Adelie
C	0.0068	0.70	0.0667	Adelie
D	0.5102	0.00	1.0000	Gentoo
E	0.9592	0.16	1.0000	Gentoo
F	0.5850	0.06	0.8667	Gentoo
G	1.0000	1.00	0.1667	Chinstrap
H	0.6667	0.52	0.2333	Chinstrap

(3 marks)

Solution:

$S_1(x) = \dfrac{x - 35.9}{14.7}$ .....[1 mark]

$S_2(x) = \dfrac{x - 14.4}{5}$ .....[1 mark]

$S_3(x) = \dfrac{x - 188}{30}$ .....[1 mark]

Average: 2.33 / 3 marks in Jan 2024; 14.81% below 1.5 marks.

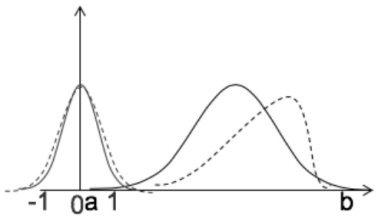
1.6.6 Standard Scaler/Standardisation Examples

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 1.6.2 (Final Exam May 2022 Sem, 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. (6 marks)

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.

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

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

[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 Data Cleaning/Cleansing & Data Imputation

A tabular data may have "missing" or "corrupted" or "wrong" values.

① We **DO NOT THROW AWAY ANY DATA** by default, especially when we have very few data (e.g. less than 500). **DON'T use univariate analysis method to claim any data as outlier!** This is because many data are imbalanced.

② If we have more than 10% missing values, we may need to **impute** the data.
③ If we have a lot of data and a few missing values, we may just clean the data by removing the rows with missing values.

Popular steps in handling missing values in structured data:

- 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. $> 10\%$), we need to find out where are they coming from:
  - If one or two columns are the cause, we remove those columns.
  - If the missing values spread everywhere, we use missing value heatmap to try to identify some patterns and decide on how to impute the data if necessary.
    * 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

In predictive modelling, "modelling" is usually accomplish by "learning/generalising from data":

$$\boxed{\text{Generalisation} = \text{Data} + \text{Knowledge (Class of Models)}}$$

The predictive models we investigate can only handle **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 has the form **with no labels**

$$\boldsymbol{x}_1,\ \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n$$

and the items in $\boldsymbol{x}_i$ are mostly numeric (or can be encoded into numeric values) 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 models of the following classes could be tried:

- Descriptive Statistics / EDA. E.g. Data understanding (Section 1.5)
- Visualisation $\rightarrow$ 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 predictive modelling, if the tabular data has the form

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

where **target variable (or response)** Y associated with y_i is **numerical** or ***quantitative***, the problem is called a ***regression problem*** and the predictive model is called a ***regressor***.

Regression models below could be tried:

- Linear regression models:

$$Y = \beta_0 + \sum_{j=1}^{p} \beta_j X_j \quad \Rightarrow \quad y_i = \beta_0 + \beta_1 x_{i,1} + \cdots + \beta_p x_{i,p} \tag{1.3}$$

- Nonlinear regression models (`https://www.sciencedirect.com/topics/computer-science/nonlinear-regression`):

$$Y = \phi_{w_1,...,w_m}(X_1, ..., X_p) \quad \Rightarrow \quad y_i = \phi_{w_1,...,w_m}(x_{i1}, \cdots, x_{ip}) \tag{1.4}$$

- Nonparametric regression models.

Note: $i = 1, ..., n$.

Exercise 1.7.1 (Simple Linear Regression)**.** If the linear regression (a kind of supervised learning 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: $\hat{y} = 150.93$]

Hint: Use (1.3) with $n = 10$, $p = 1$ and $\widehat{\beta}_1 = \frac{n\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i)(\sum_{i=1}^{n} y_i)}{n(\sum_{i=1}^{n} x_i^2) - (\sum_{i=1}^{n} x_i)^2}$, $\hat{\beta}_0 = \overline{y} - \widehat{\beta}_1 \overline{x}$.

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

In predictive modelling, if the tabular data has the form

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

where the target variable Y is **categorical** or ***qualitative***, the problem is called a ***classification problem*** and the predictive model is called a ***classifier***.

> **Note**: the $\boldsymbol{x}_i$ are usually called **inputs / attributes / features / columns / independent variables / explanatory variables / predictors / explanatory factors / exogenous variable**, etc. [Afifi and Clark, 1997] with p 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 / endogenous variable**, etc. and is usually a single component.

The classifier models be to fit the observed data $(\boldsymbol{x}_i, y_i)$ could be tried:

- kNN (Chapter 2)
- logistic regression (Chapter 3)
- naive Bayes (Section 4.2)
- discriminant analysis (Section 4.3)
- classification trees (Chapter 5)
- ...

1.7.4 Approaches to Classifying Models

An approach to classify/develop the predictive models (classifiers in particular) is the 'Bayesian statistics' approach:

- **Discriminative models**: They try to model the input and output as the probability of observing output Y given inputs $X_1, ..., X_p$.

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

  E.g. linear regression (1.3), kNN, logistic model, classification trees, etc.

- **Generative models**: They try to model the conditional probability of the inputs $X_1, ..., X_p$ based on the output Y. This requires the distribution of the output $P(Y)$ to be modelled as well — this is difficult for regression problem and thus there are rarely generative models for regression problems.

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

  E.g. Naive Bayes, LDA, etc.

  Note: Generative models in effect require the modelling of the the joint distribution over all variables $\mathbb{P}(X_1, \cdots, X_p, Y)$.

Another approach to develop/classify predictive models are based on the 'internal' parameters.

- **Parametric models**: This approach use mathematical formulations with a **fixed set of parameters**. The "training" process tries to find the most suitable parameter values to minimise "errors". E.g. logistic regression
- **Nonparametric models**: This approach usually split the data into multiple sub-data and tries to fit the sub-data with a model. Therefore, **the model has no fixed set of parameters**. The internal "representation" **grows as data increases**. The "internal models" will get large when there are a lot of sub-data to be fitted. E.g. kNN, tree

1.7.5 Applications of Modelling

Suppose we have built a model h for the labelled data $(\boldsymbol{x}_i, y_i)$ (for both regression and classification problems):

$$\underbrace{Y}_{\text{output}} = h(\underbrace{X_1, \cdots, X_p}_{input}) =: h(X). \tag{1.5}$$

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 are "more significant" in sales?

All models can perform prediction but only some can perform inference.

Example 1.7.4 (Churn Prevention — Classification Problem)**.** Customer retention is important for any companies (e.g. in telecommunication companies) since markets are rather saturated. Many companies are engaged in battles to attract each other's customers while retaining their own. Customers switching from one company to another is called **churn**.

The churn prevention modelling is more interested in inference, i.e. to identify the significant factors that cause churn and try to apply strategies to reduce significant factors.

Some concrete programming analysis on churn identification can be found from the links below:

- `https://www.kaggle.com/c/kkbox-churn-prediction-challenge/data`
- `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.

1.7.6 Model Training and Prediction

Given a tabular batch/offline training data

$$D = ((\boldsymbol{x}_1, y_1),\ (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n)). \tag{1.6}$$

Suppose that $\mathcal{X}$ (containing inputs $\boldsymbol{x}_i$) is the domain and $\mathcal{Y}$ (containing y_i) is the label set. A **(model) training algorithm** tries to find $h : \mathcal{X} \to \mathcal{Y}$ from a class of functions (1.5) from the "hypothesis" $\mathcal{H}$, to fit the data D by minimising **training error** or **loss**.

Since **we do not know the theoretical distribution $\mathscr{D}$ for the data** D, we have to approximate $\mathscr{D}$ using "hypothesis" function h from a class of predictive models $\mathcal{H}$ using the **theoretical/generalisation/Bayes error** or *(population) risk*

$$L_{\mathscr{D}}(h) := \mathbb{E}_{(\boldsymbol{x},y)\in\mathscr{D}}[\ell(h(\boldsymbol{x}), y)] \tag{1.7}$$

by the **empirical error** or **empirical risk**:

$$L_D(h) := \frac{1}{n}\sum_{i=1}^{n} \ell(h(\boldsymbol{x}_i), y_i) \tag{1.8}$$

where $\ell(\cdot,\cdot) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is a nonnegative mapping, called the *loss function* (which is defined differently for regression problems and classification problems).

The minimisation of (1.8) is called the ***empirical risk minimisation*** (ERM). In practice, ERM may lead to **overfitting problem** (Chapter 6), i.e. **it fits the training data but the actual model.**

The loss function used in classification problems is normally **0-1 loss**:

$$\ell_{0-1}(h(\boldsymbol{x}), y) := I(h(\boldsymbol{x}) \neq y) := \begin{cases} 0, & \text{if } h(\boldsymbol{x}) = y \\ 1, & \text{if } h(\boldsymbol{x}) \neq y \end{cases} \tag{1.9}$$

The loss function used in regression problems is usually **square loss**:

$$\ell_{sq}(h(\boldsymbol{x}), y) := (h(\boldsymbol{x}) - y)^2.$$

Theory: Statisticians show mathematically that

- for a class of predictive models that satisfy uniform convergence, the empirical risk (1.8) approximates the theoretical error (1.7);
- for every learning algorithm, there is always a distribution $\mathscr{D}$ for which it fails but other learning algorithms succeed. (**No-free-lunch theorem**).

In summary, a learning/training algorithm gives a trained model as follows:

$$\textbf{learning algorithm}\Big(\textbf{model specification}, \textbf{batch data}\Big) \xrightarrow{\text{optimisation}} \textbf{trained model}.$$

If the learning algorithm **fails**, it may give us an error message or it may give us a wrong model.

If the learning algorithm **succeeds**, a trained model will be given. The following are an R example and a Python example.

- R's learning process for logistic regression:

```
d.f = read.table("flame.txt", header=FALSE)  # Loading Data
d.f$V3 = factor(d.f$V3)                      # Data Pre-Processing

model = glm(V3 ~ ., d.f, family=binomial)    # Model Training
print(summary(model))
```

- Python's learning for logistic regression:

```
import pandas as pd
XY = pd.read_table("flame.txt", header=None) # Loading Data
XY.columns = ["X1", "X2", "Y"]               # Data Pre-Processing
XY.Y = XY.Y - 1                              # Y must be in [0,1]

import statsmodels.formula.api as sm
model = sm.logit("Y ~ X1 + X2", data=XY)
fitted = model.fit()                         # Model Training
print(fitted.summary())
```

We have limited the scope of this course to batch/offline learning/training problems. In practise, we have other learning problems which are not explored:

- Event-driven learning 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, showing advertisements based on the online shopping clicks, etc.

- Real-time learning 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.

1.7.7 Flexibility vs Interpretability

There are different classes of predictive models (Section 1.7.3). Some of them are more flexible (which fit the training data with small error) while some of them are more interpretable (the mathematical formulation can explain the relations between inputs and output). These two features are usually conflicting, i.e. flexible models are not interpretable while interpretable models are usually inflexible.

To illustrate this point, we use polynomial regression as an example.

The linear regression is a class of polynomial regressions of degree 1 which is inflexible because it cannot fit data which are not on a line:

$$y = ax + b.$$

However, it is very interpretable because from the formula, we can have an interpretation: When x increase by 1, y will increase by a.

The quadratic regression is a class of polynomial regressions of degree 2 which is more flexible, but it could only fit data which are convex in some range:

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

The model is slightly more difficult to interpret: The input x will influence y at a "quadratic manner" when $|x| > 1$ but when $|x| \ll 1$, then y has little changes.

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. Figure 1.1 illustrates of some predictive models based on flexibility and interpretability [James et al., 2013].

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

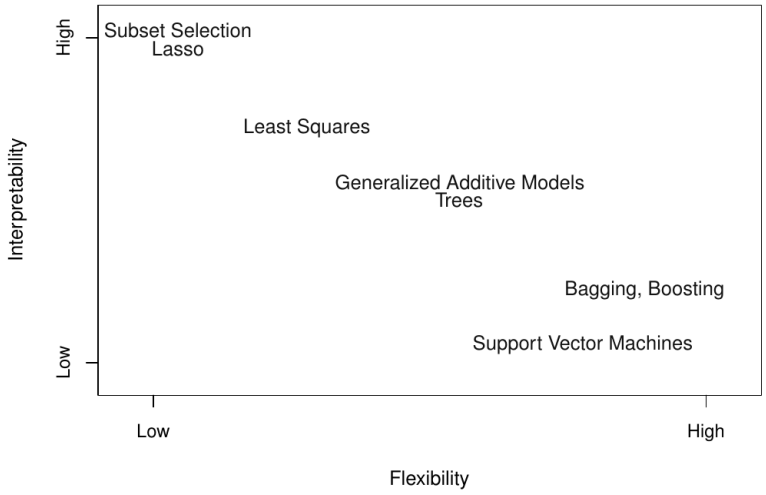


Figure 1.1: Flexibility vs Interpretability

Solution: A more flexible (complicated) model can usually lead to a higher accuracy. However, a more complicated model is more difficult to interpret. Hence, the accuracy and interpretability are always opposing each other — a model with higher flexibility (accuracy) will have lower interpretability, and vice versa.

An analyst should choose a model that best fit their goal — prediction accuracy or model interpretability.

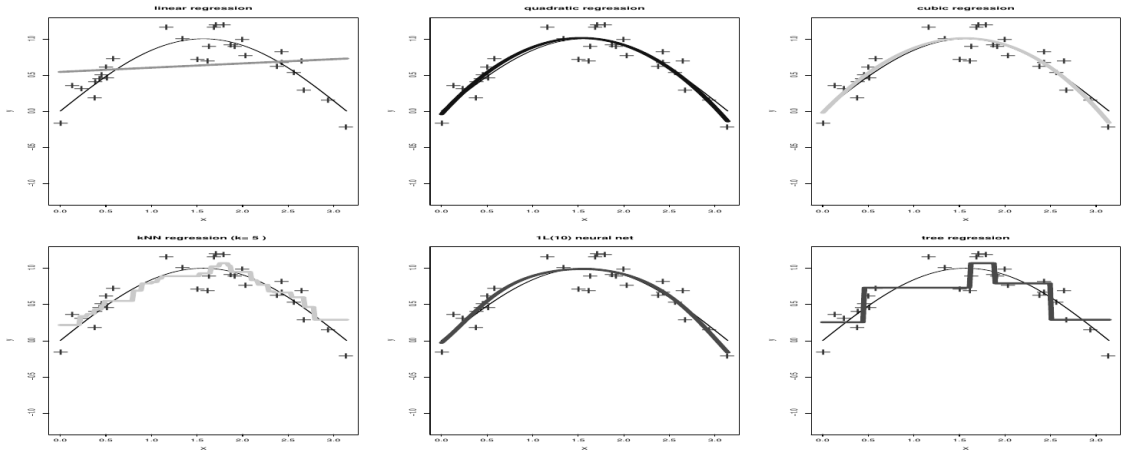
Example 1.7.6. Suppose the output y is governed by the input x following the equation:

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

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

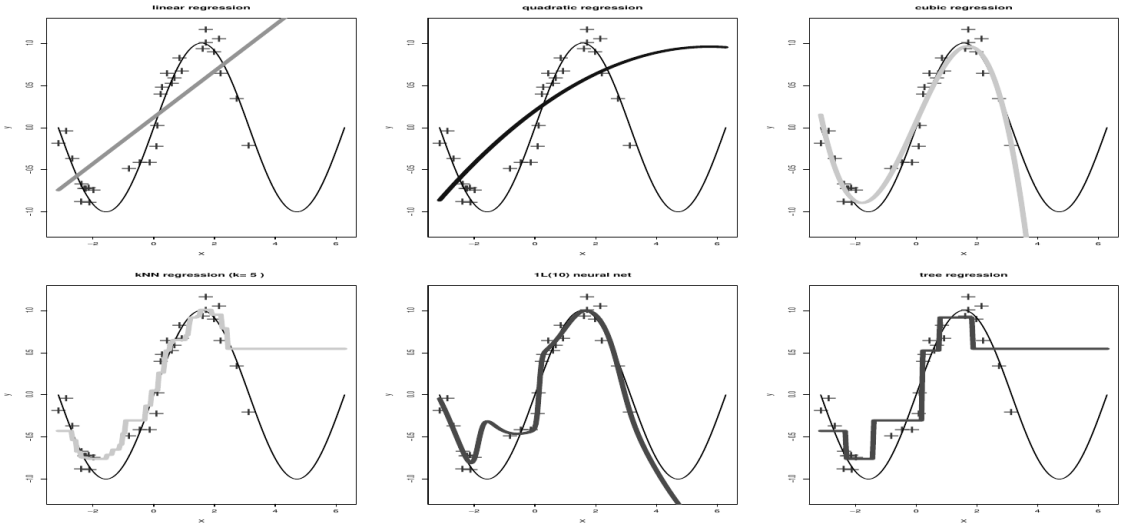
and determine which model is the best based on the interpretability and flexibility.



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

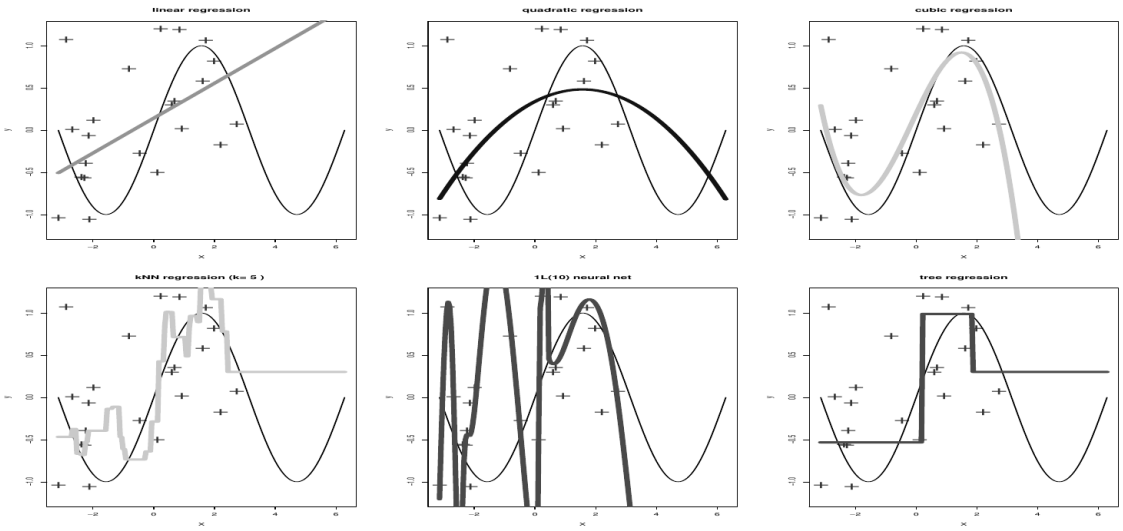
- Inflexible ⇒ Simpler math formula ⇒ Poorer Predictibility, better inference(?)
- Flexible ⇒ Complicated math formula ⇒ Good Predictibility, poorer inference(?) □

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. σ 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: The **validation approach** to estimating the theoretical/generalisation error (1.7) from the observed data (1.6) is to use some part of it as validation data [Lindholm et al., 2022].

The theoretical result for having a new set of data following the theoretical distribution $\mathscr{D}$ is given by the following theorem.

Theorem 1.8.1. *Let h be a predictive model and assume that the loss function is in the range $[0, 1]$. Then $\forall \delta \in (0, 1)$, with a probability of at least $1 - \delta$, we sample a validation set V s.t. $|V| = n_v$ from the theoretical distribution $\mathscr{D}$ i.i.d. and independent of the training data D, we have*

$$|L_V(h) - L_{\mathscr{D}}(h)| \leq \sqrt{\frac{\ln(2/\delta)}{2n_v}}.$$

Sampling a training set D and then sampling an independent validation set V is equivalent to randomly partitioning our random set of examples into two parts. This is why **validation set** is often referred to as a **hold out set**.

The empirical error associated with the *training data* D is called the *training error* while the empirical error associated with the *validation data* is called the *testing error*. It is an estimate of the generalisation error. The difference between the generalisation error and the testing error is called the *generalisation gap* [Lindholm et al., 2022].

The hold out method assumes the data is plentiful and we have the ability to sample a fresh validation set. If the observed data is scarce, we need the k-fold cross validation approach discussed in Section 2.9.

1.8.1 The Holdout / Validation Set Approach

Why is overfitting (Section 1.7.6) a problem?

- An overfitting model works extremely well on **training data** but may **perform terribly with unseen/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.

In practise, the ***holdout method***, ***split validation*** or ***validation set approach*** [Coelho and Richert, 2015, Chapter 2] or train/test split method (the term used in Python's scikit-learn library) is used to shuffle and divide a time-independent tabular data into "training set" for model fitting and "validation set" for evaluating the fitted model [James et al., 2013, Section 5.1]:

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 1.8.2 (Linear Sampling Illustrating with Excel)**.** 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)				

The following figure illustrates 70%-30% holdout method.

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. □

Example 1.8.3 (Linear Sampling Illustrating with Computer Programs)**.**

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 the data is **imbalanced** in a classification problem, **stratified sampling** should be used to prevent bias in the sampling. The **linear sampling** is useful only when the size of the data is large, mostly balanced (i.e. the classes are **uniformly distributed**).

When the data is **time-dependent**, e.g. the ISLR's Smarket data in the Practical 3 script `p03_knn1.R`, the holdout method will be based on a random cut-off time t_c, i.e. we split the data by an earlier period $(\boldsymbol{x}_i, y_i)|t_i < t_c$ and a later period $(\boldsymbol{x}_i, y_i)|t_i \geq t_c$.

When we have a rather large data and a set of predictive models, we may 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 the set of predictive models to have a final comparison. This is usually used in predictive models with hyperparameters.

Example 1.8.4 (Final Exam Jan 2019 Sem, 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.

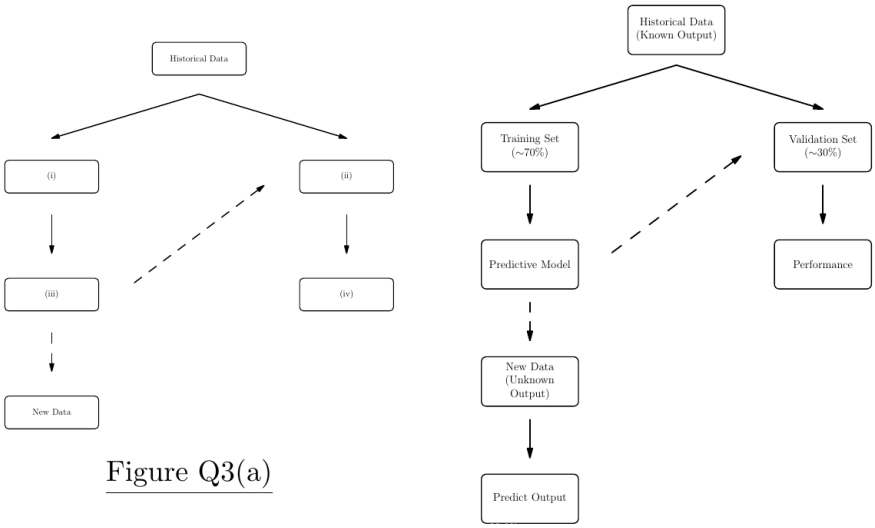


Figure Q3(a)

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)

Solution:

(i) Training Data [1 mark]
(ii) Validation Data [1 mark]
(iii) Predictive Model [1 mark]
(iv) Performance [1 mark]

Differences between regression and classification: ..... [8 marks]

Step	Regression	Classification
Historical data	Numerical response	Categorical response
Splitting data	Linear sampling	Stratified sampling
Performance	Sum of squared error, R^2	Confusion matrix
Scoring data	Predicted value $\pm$ s.d.	Probability of classes

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

Solution: Three examples are:

• Banking industry uses credit scores to predict individual's delinquency behaviour and evaluates the credit worthiness of each applicant.

• Insurance industry predicts 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.

1.8.2 Performance Evaluation Examples for Regression Problems

In a regression problem, the popular **empirical evaluation metrics** with different loss functions are listed below:

- (Empirical) Sum of Squared Error (SSE) / (Empirical) Residue Sum of Squares (RSS) and (Empirical) MSE:
$$SSE = \sum_{i=1}^{n} \left(y_i - \widehat{y}_i\right)^2, \quad MSE = \frac{SSE}{n}, \quad \ell(y_1, y_2) = (y_1 - y_2)^2.$$
- (Empirical) Mean Absolute Error (MAE):
$$MAE = \frac{1}{n}\sum_{i=1}^{n} \left|y_i - \widehat{y}_i\right|, \quad \ell(y_1, y_2) = |y_1 - y_2|.$$

The **coefficient of determination**, R^2, despite not being an empirical error, is a popular statistical measure that represents the proportion of the variance for a dependent variable (the output) that is explained by an independent variable in (1.5) and is used in performance evaluation:
$$R^2 = 1 - \frac{\sum(y_i - \widehat{y}_i)^2}{\sum(y_i - \frac{1}{n}\sum_i y_i)^2} = 1 - \frac{SSE}{SST}.$$

Remark 1.8.5. The list above is not comprehensive. More loss functions can be found at `https://en.wikipedia.org/wiki/Loss_function`. In reality, to make the training process more robust (i.e. a few data should not cause the trained model to change drastically), hybrid loss such as Huber loss, etc. are introduced in Lindholm et al. [2022, Chapter 5].

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].

Example 1.8.6 (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.

Solution: The **default** empirical regression error is usually the mean square error that we are familiar with:
$$MSE = \frac{1}{10}((11.64 - 10.96)^2 + ... + (10.52 - 10.15)^2) = 0.30231$$
$$MAE = \frac{1}{10}(|11.64 - 10.96| + ... + |10.52 - 10.15|) = 0.453$$

The empirical R^2 calculation is

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

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

Example 1.8.7 (Final Exam May 2022 Sem, 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)

Solution: Two popular empirical accuracy measures associated with the generalisation error are:

1. Mean Squared Error (MSE) $= \frac{1}{n}\sum_{i=1}^{n}(y_i - h(\boldsymbol{x}_i))^2$ ..... [1.5 marks]
2. $R^2 = 1 - \dfrac{\sum_{i=1}^{n}(y_i - h(\boldsymbol{x}_i))^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i)^2}$ ..... [1.5 marks]

According to the mathematical formula, MSE should be close to the variance of the noise when the regressor fits the theoretical model well and can be very large when the regressor fits the theoretical model poorly. MSE ranges from 0 to ∞. ..... [1 mark]

According to the mathematical formula, R^2 should be close to 1 when the regressor fits the theoretical model well and close to 0 or negative values when the regressor fits the theoretical model badly. R^2 ranges from $-\infty$ to 1. ..... [1 mark]

Example 1.8.8 (Final Exam Jan 2024 Sem, Q1(c))**.**

When the linear regression model is applied to study the relation between average hourly earnings (y_i) against the years of education (`educ`), the predictions $\widehat{y}_i$ in Table 1.2 are obtained.

Table 1.2: Comparison between results from a regression model to actual data

`educ`	y_i	$\widehat{y}_i$
14	8.8	7.2
10	5.1	4.7
16	8.3	8.4
18	10.0	9.6
6	2.9	2.2
12	2.9	5.9

1. Calculate the sum of squared errors (SSE). (3 marks)

   Ans: 12.38 (Average: 2.65 / 3 marks in Jan 2024; 10.91% below 1.5 marks.)

2. the coefficient of determination, R^2. (3 marks)

Ans: 0.7447 (Average: 2.31 / 3 marks in Jan 2024; 20% below 1.5 marks.)

1.8.3 Performance Evaluation Examples for General Classification Problems — Confusion Matrix

In a classification problem, ***confusion matrix*** (or ***contingency table***) is a matrix or a table that summarises the number of correct and incorrect classifications and it provides more information than the misclassification error stated below.

The **empirical evaluation metric** normally used in classification problems is the ***misclassification error*** (similar to (1.9)):

$$\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i), \quad \ell(y_1, y_2) = I(y_1 \neq y_2) \tag{1.10}$$

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 is used in the statistical summary of confusion matrix and it is the complement of the misclassification error:

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

Example 1.8.9. Suppose that 75% of the iris data is used as training data and the remainder 25% iris data is used as testing data (using set.seed(301) in WSL2 @ Windows 11 with R-4.1.2). For the particular testing sample, the following confusion matrix is obtained using the kNN($k = 5$) predictive model:

	actual		
predicted	setosa	versicolor	virginica
setosa	12		
versicolor		9	1
virginica		3	11

By using the gmodels's CrossTable, we obtain

```
             | testing$Species
  knnPredict |     setosa | versicolor |  virginica |  Row Total |
-------------|------------|------------|------------|------------|
      setosa |         12 |          0 |          0 |         12 |
             |     16.000 |      4.000 |      4.000 |            |
             |      1.000 |      0.000 |      0.000 |      0.333 |
             |      1.000 |      0.000 |      0.000 |            |
             |      0.333 |      0.000 |      0.000 |            |
-------------|------------|------------|------------|------------|
  versicolor |          0 |          9 |          1 |         10 |
             |      3.333 |      9.633 |      1.633 |            |
             |      0.000 |      0.900 |      0.100 |      0.278 |
             |      0.000 |      0.750 |      0.083 |            |
             |      0.000 |      0.250 |      0.028 |            |
-------------|------------|------------|------------|------------|
   virginica |          0 |          3 |         11 |         14 |
             |      4.667 |      0.595 |      8.595 |            |
             |      0.000 |      0.214 |      0.786 |      0.389 |
             |      0.000 |      0.250 |      0.917 |            |
             |      0.000 |      0.083 |      0.306 |            |
-------------|------------|------------|------------|------------|
Column Total |         12 |         12 |         12 |         36 |
             |      0.333 |      0.333 |      0.333 |            |
-------------|------------|------------|------------|------------|
```

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.

By using the caret's confusionMatrix under similar conditions, we obtain the confusion matrix:

```
Confusion Matrix and Statistics

            Reference
Prediction   setosa versicolor virginica
  setosa         12          0         0
  versicolor      0          9         1
  virginica       0          3        11

Overall Statistics

               Accuracy : 0.8889
                 95% CI : (0.7394, 0.9689)
    No Information Rate : 0.3333
    P-Value [Acc > NIR] : 6.677e-12

                  Kappa : 0.8333

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: setosa Class: versicolor Class: virginica
Sensitivity                 1.0000            0.7500           0.9167
Specificity                 1.0000            0.9583           0.8750
Pos Pred Value              1.0000            0.9000           0.7857
Neg Pred Value              1.0000            0.8846           0.9545
Prevalence                  0.3333            0.3333           0.3333
Detection Rate              0.3333            0.2500           0.3056
Detection Prevalence        0.3333            0.2778           0.3889
Balanced Accuracy           1.0000            0.8542           0.8958
```

The `caret::confusionMatrix` generalised the binary classification metrics TPR, TNR, PPV, NPV, etc. to more than two classes by comparing each factor level to the remaining levels (i.e. a "one versus all" approach).

Example 1.8.10 (Final Exam May 2022 Sem, 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)

   Solution: Decision tree model (kNN need preprocessing or Gower distance to work, so minor marks deducted for it) ..... [2 marks]

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

Solution: Naive Bayes model (linear discriminant analysis (LDA) needs preprocessing or Gower distance to work, so minor marks deducted for it) ..... [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)

Ans: 0.8951

1.8.4 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, $\text{NPV} = \dfrac{\text{TN}}{\text{TN} + \text{FN}}$

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

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

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 1.8.11 (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 $= 1 - 77.78\% = 22.22\%$

1.8.5 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 1.8.12. Apply the ROC analysis on the ISLR's Smarket data using the `pROC` package.

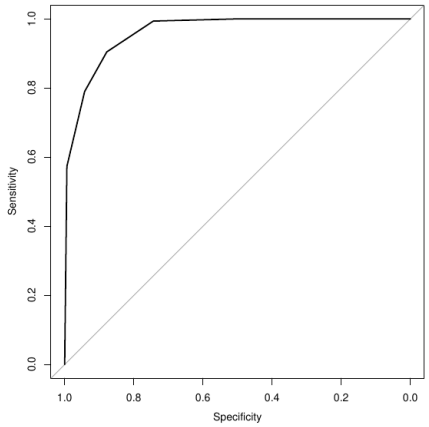
Solution:

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

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 1.8.13 (Final Exam May 2022 Sem, 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)

Ans: 0.6571429

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

Ans: 0.7419355

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

Ans: 0.7297297

4. Accuracy and Kappa Statistic

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

where

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

Ans: 0.4285714 (2 marks)

Example 1.8.14 (Final Exam Jan 2024 Sem, Q1(b))**.** Given the confusion matrix of a trained predictive model in Table 1.1 with 0 as positive.

Table 1.1: Confusion matrix on the training data. 0 is positive.

	Actual	
Prediction	0	1
0	579	127
1	71	143

1. Find the balanced accuracy, i.e. the average of the recalls. (2 marks)

Ans: 0.7102 (Average: 1.02 / 2 marks in Jan 2024; 47.27% below 1 mark.)

2. Find the accuracy and kappa statistic

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

where $\text{RandomAccuracy} = \frac{(\text{TP+FN}) \times (\text{TP+FP}) + (\text{TN+FP}) \times (\text{TN+FN})}{(\text{Total Number of Data})^2}$. (5 marks)

Ans: 0.7848, 0.4476 (Average: 4.57 / 5 marks in Jan 2024; 7.27% below 2.5 marks.)

3. Use proper examples to discuss whether the accuracy is a good performance metric for an imbalanced data. (3 marks)

(Average: 0.42 / 3 marks in Jan 2024; 87.27% below 1.5 marks.)

Solution: The accuracy is a **not** a good performance metric for imbalanced data because it is unable to identify the bad predictive model which identifies the majority correctly which the minority very poorly as illustrated below.

..... [1 mark]

Model A	Actual	
Prediction	0	1
0	900	100
1	0	0

Model B	Actual	
Prediction	0	1
0	800	0
1	100	100

Model A and Model B both give an accuracy of 0.9. However, Model A cannot predict 1 at all while Model B can predict 1 very well. ..... [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`.

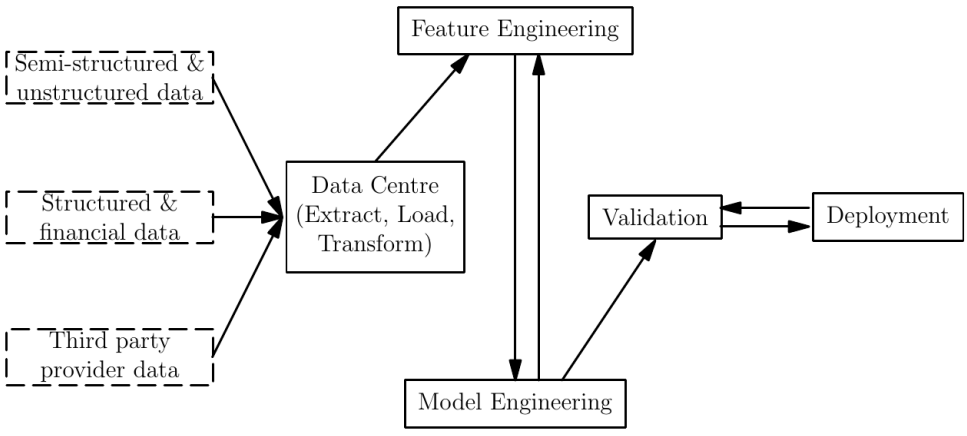
1.9 Deployments

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

- **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.

- **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.
- **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.
- **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.

The report Bank of England [2019] has summarised a survey showing 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.



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

Solution:

Classification (Ref: `https://blog.usejournal.com/machine-learning-algorithms-use-cases-7264`

(i) classify email to spam and non-spam;

(ii) classifying patients based on the tumour they have: benign or malignant.

(iii) credit ratings — By considering factors such as customer's earning, age, savings and

financial history we can do it. This information is taken from the past data of the loan. Hence, Seeker uses to create a relationship between customer attributes and related risks.

(iv) face detection — The categories might be face versus no face present. There might be a separate category for each person in a database of several individuals.

(v) character recognition — We can segment a piece of writing into smaller images, each containing a single character. The categories might consist of the 26 letters of the English alphabet, the 10 digits, and some special characters.

(vi) 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`)

Regression:

(i) A factory manager, for example, can create a statistical model to understand the impact of oven temperature on the shelf life of the cookies baked in those ovens.

(ii) In a call centre, we can analyse the relationship between wait times of callers and number of complaints.

(iii) Sales forecasting: Data-driven decision making eliminates guesswork, hypothesis and corporate politics from decision making. This improves the business performance by highlighting the areas that have the maximum impact on the operational efficiency and revenues.

Unsupervised learning (Ref: `https://pythonistaplanet.com/applications-of-unsupervised-learn`

(i) **Cluster analysis** is the process of grouping the given data into different clusters or groups. E-commerce websites like Amazon use clustering algorithms to implement the user-specific recommendation system. (Quora answer)

(a) Market segmentation divides the consumers of the market into some groups. In a group, consumers will be similar to each other based on some predefined set of characteristics. If two customers are not similar based on these characteristics, they are in different groups. Companies use this clustered data and the features of the customers to decide their market strategies, like which group of customer they should target or which group of customers needs more advertising etc. etc.

(b) There are millions of people in social networking websites and analysing their behaviours sounds really fun. Clustering plays the role here. This idea of social networks analysis can be extended to real life social scenarios.

(c) Search engines and many other websites use clustering to group similar web pages, videos, songs etc. and improve results for their users.

(ii) Visualisation is the process of creating diagrams, images, graphs, charts, etc., to communicate some information.

(iii) **Dimensionality reduction**: Many machine learning problems contain thousands of features for each training instance. This will make the training slow as well as it will be difficult to obtain a good solution to the problem. Dimensionality reduction simplifies the data without losing too much information.

(iv) Bioinformatics: try to determine the living species based on the (complete or fragment of) genetic information

(v) **Finding Association Rules**: This is the process of finding associations between different parameters in the available data. It discovers the probability of the co-occurrence of items in a collection, such as people that buy X also tend to buy Y. It is used in 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`

(vi) 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)

(vii) **Anomaly detection**: The identification of rare items, events or observations which brings suspicions by differing significantly from the normal data.

Example 1.9.2. Consider the sort of issues one will face when deploying the predictive models in the following case studies:

- local server
- cloud

For a **local server**, one may need a larger IT team to choose and maintain the hardware (the number of memory, CPU, GPU, etc.), the operating system (various GNU/Linux distros (e.g. Ubuntu Server, Redhat), FreeBSD (https://freebsdfoundation.org/freebsd/, `https://papers.freebsd.org/2019/fosdem/looney-netflix_and_freebsd/`), OpenBSD or Windows) and a local server stack (a web server such as Nginx, Apache HTTPd, etc. and a database server such as PostgreSQL, MariaDB, etc.). It is possible to test the server stack with a virtual environment such as the Docker or Kubernetes, however, for the best performance, the server stack should be local rather than virtual.

For a local server, the maintainance can be more complex depending company policy. If one deploys the distro from big software companies such as Redhat or Microsoft, the software will be constantly up to date and service support is available. However, the annual service cost can be high. If free servers are deployed, then one may need to face

- Operating system security: 0-day exploit, encryption,
- Software security: memory vulnerability
- Data management: How often should the database server be updated to enhance the security? Is encryption being used? Who controls the encryption key? Can the encrypted data be recovered with reasonable effort?
- Model update: How often should the predictive model be updated with incoming new data?

Despite being rare but hardware failure is also important:

- Can the data be recovered from hardware failure?
- Are the data backup regularly?

For a **Cloud**: one may need a smaller IT team to manage a virtual environment with an appropriate operating system (similar to the local server case) and then deploy either a docker of server stacks or the more complex kubernetes servers. Despite the convience, a disruption of the Internet may cause the data inaccessible when the predictive model is required for local use.

In both case studies, vendor lockdown may be an issue but the cloud is more prone to monopoly by cloud service provider such as Amazon AWS, Microsoft Azure, Google, etc. because the data are on the Web rather than local data centre. E.g. Google Cloud DOWNSCALED Its Customers (`https://www.youtube.com/watch?v=qBmNSQGifKQ`)

Another issue common to both case studies is the proper use cryptographic protocols and service protocols. Network service may be prone to DDoS, cyber attack. Setting up intrusion detection and preventing phishing and social engineering are equally important.

Example 1.9.3 (Final Exam May/June 2023 Sem, Q1(d))**.** In a case study where a company is considering whether to deploy the predictive models in (i) a local server or (ii) a cloud server. Compare the pros and cons of both cases from the perspectives of administrative cost, disaster recovery, data safety and vendor lock-in problem. (5 marks)

Solution: The comparison is shown in the table below. ..... [$1.25 \times 4 = 5$ marks]

	(i) a local server	(ii) a cloud server
administrative cost	usually higher because a larger IT team is required to maintain the hardware and software such as webserver, data analytical software, etc.	slightly lower because a smaller IT team is required to maintain the software and there is usually no hardware administration burden
disaster recovery	need to face higher risk in hardware failure disaster and and disaster due to network attacks. The recovery needs to be conducted by an highly experience IT team performing regular data backup.	need to face lower risk in hardware failure disaster (due to cloud servers having regular hardware maintainance) and lower risks in network attacks since cloud servers have very strong team of experts in dealing with network attacks.
data safety	data is kept safe locally if regular backups are conducted and the data are kept in a secure centre with proper data protection.	data is at the hand of cloud servers which may lock out the access when there is any late payment or change of terms of service by cloud server provider.
vendor lock-in problem	less vendor lock-in problem because there are a lot of free distributed servers such as FreeBSD, Ubuntu server, etc.	serious vendor lock-in problem when the cloud storage is large and when the cloud server limits the upload and download bandwidth, things become worst.

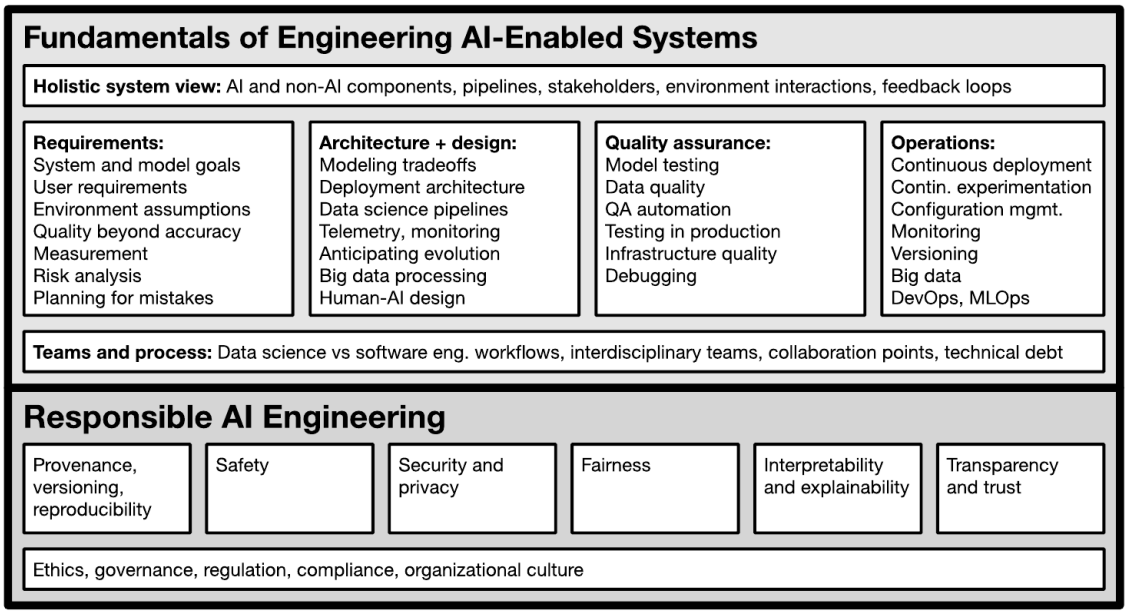


Figure: Deployment framework from `https://mlip-cmu.github.io/s2025/`

1.10 Appendix: Data Format Handling

Usual CSV Data Suppose a data is stored in `file.csv`, then we can usually read it using the command

```
d.f = read.csv("file.csv")
```

However, if the CSV file does not have a header, we need to read it using

```
d.f = read.csv("file.csv", header=F)
```

If we want to convert strings into categorical data and handle missing values (e.g. as empty string), then we need to use the command

```
d.f = read.csv("file.csv", stringsAsFactors=T, na.strings=c(""))
```

More cases will be studied in the practical.

Usual Tabular Text Data Suppose we have tabular data `file.dat` in text format with a header, we can try the command

```
d.f = read.table("file.csv", header=T)
```

Zip Archive Suppose a zip file `data.zip` contains `a.csv`, `b.csv`, etc. The command to read the data from `a.csv` is

```
d.f.a = read.csv(unz("data.zip", "a.csv"))
```

Excel Data If we want to open an 2003+ Excel data file `data.xlsx` in R, we can use the `read_excel()` function from the `readxl` library.

```
#install.packages("readxl")
library(readxl)
print(excel_sheets("data.xlsx"))
# Read the first sheet in an Excel workbook
sh1 = read_excel(data_file)
# Read the second sheet in an Excel workbook
sh2 = read_excel(data_file, sheet=2)
# Read the third sheet in an Excel workbook
sh3 = read_excel(data_file, sheet=3)
```

1.11 Appendix: Probability Distributions

In this section, we list some probability distributions in R which may be used in the Practical 1. Detail references can be found at `https://en.wikipedia.org/wiki/List_of_probability_distributions`, `https://www.stat.umn.edu/geyer/old/5101/rlook.html`, Field Guide to Continous Probability Distributions (https://threeplusone.com/pubs/FieldGuide.pdf).

1.11.1 Discrete Distributions

Binomial

The binomial distribution is a discrete probability distribution of n independent experiments, each asking yes-no question and each with p probability of success and $1-p$ probability of failure. The probability mass function is given by

$$f_{n,p}(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad 0 \le p \le 1, \ x = 0, 1, ..., n \tag{1.11}$$

is available in R through `dbinom(`x`, size=`n`, prob=`p`, log=FALSE)`.

It is used in GLM for the `binomial(link="logit")` family.

The random generation, distribution function and quantile function are given by `rbinom`, `pbinom`, `qbinom` respectively.

Negative Binomial

The negative binomial distrbution is a discrete probability distrbution that models the number of failures in a sequence of independent and identically distributed Bernoulli trials before n successes occur.

$$f_{n,p}(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!}p^n(1-p)^x, \quad 0 < p \leq 1,\ x = 0, 1, ..., n \tag{1.12}$$

is available in R through `dnbinom(`x`, size=` n`, prob=`p`, mu, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rnbinom`, `pnbinom`, `qnbinom` respectively.

Poisson

The Poisson distribution is a discrete probability distribution that expresses the probability of a given number of events occuring in a fixed interval of time if these events occur with a known constant mean rate and independently of the time since the last event.

$$f_\lambda(x) = \frac{\lambda^x e^{-\lambda}}{x!}, \quad \lambda > 0,\ x = 0, 1, 2, ... \tag{1.13}$$

is available in R through `dpois(`x`, lambda=`λ`, log = FALSE)`.

It is used in GLM for the `poisson(link="log")` family.

The random generation, distribution function and quantile function are given by `rpois`, `ppois`, `qpois` respectively.

Geometric

The geometric distribution of the number $Y = X - 1$ of failures before the first success:

$$f_p(x) = (1-p)^x p, \quad 0 \leq p \leq 1,\ x = 0, 1, 2, 3, ... \tag{1.14}$$

is available in R through `dgeom(`x`, prob=`p`, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rgeom`, `pgeom`, `qgeom` respectively.

Hypergeometric

The hypergeometric distribution is a discrete probability distribution that describes the probability of x successes (random draws for which the object drawn has a specified feature) in k draws, without replacement, from a finite population of size $m + n$ that contains exactly m objects with that feature, wherein each draw is either a success or a failure. In contrast, the binomial distribution describes the probability of x successes in n draws with replacement.

$$f_{m,n,k}(x) = \binom{m}{x}\binom{n}{k-x}/\binom{m+n}{k}, \quad x = 0, ..., k \tag{1.15}$$

is available in R through `dhyper(`x`, `m`, `n`, `k`, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rhyper`, `phyper`, `qhyper` respectively.

1.11.2 Continuous Distributions

Uniform

The continuous uniform distributions or rectangular distributions are a family of symmetric probability distributions with a probability density function:

$$f_{a,b} = \begin{cases} \frac{1}{b-a}, & a \leq x \leq b \\ 0, & \text{otherwise} \end{cases} \tag{1.16}$$

is available in R through `dunif(`x`, min=`a`, max=`b`, log=FALSE)`.

The random generation, distribution function and quantile function are given by `runif`, `punif`, `qunif` respectively.

Beta

The beta distribution is a family of continuous probability distributions defined on the interval $[0, 1]$ in terms of two positive parameters α and β, that appear as exponents of the variable and its complement to 1, respectively, and control the shape of the distribution:

$$f_{\alpha,\beta}(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{\alpha-1}(1-x)^{\beta-1}, \quad \alpha > 0,\ \beta > 0,\ 0 \leq x \leq 1 \tag{1.17}$$

is available in R through `dbeta(x, shape1=`α`, shape2=`β`, ncp=0, log=FALSE)`

The random generation, distribution function and quantile function are given by `rbeta`, `pbeta`, `qbeta` respectively.

Logistic

The logistic (or sech-square) distribution is a continuous probability distribution which is a special case of the Tukey lambda distribution.

$$f_{\mu,s}(x) = \frac{e^{-\frac{x-\mu}{s}}}{s(1+e^{-\frac{x-\mu}{s}})^2} = \frac{1}{4s}\text{sech}^2\left(\frac{x-\mu}{2s}\right) \tag{1.18}$$

is available in R through `dlogis(`x`, location=`μ`, scale=`s`, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rlogis`, `plogis`, `qlogis` respectively.

Normal

A normal distribution or Gaussian distribution has a probability density function:

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{1.19}$$

is available in R through `dnorm(`x`, mean=`μ`, sd=`σ`, log=FALSE)`

It is used in GLM for the `gaussian(link="identity")` family.

The random generation, distribution function and quantile function are given by `rnorm`, `pnorm`, `qnorm` respectively.

Log Normal

A log-normal distribution is a continuous probability distribution of a random variable whose logarithm is normally distributed:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} \exp(-\frac{(\ln x - \mu)^2}{2\sigma^2}) \tag{1.20}$$

is available in R through `dlnorm(`x`, meanlog=`μ`, sdlog=`σ`, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rlnorm`, `plnorm`, `qlnorm` respectively.

Exponential

The exponential distribution is the probability distribution of the distance between events in a Poisson point process, i.e., a process in which events occur continuously and independently at a constant average rate. It is a particular case of the gamma distribution. It is the continuous analogue of the geometric distribution, and it has the key property of being memoryless.

$$f_\lambda(x) = \lambda e^{-\lambda x}, \quad x \geq 0 \tag{1.21}$$

is available in R through `dexp(`x`, rate=`λ`, log=FALSE)`

The random generation, distribution function and quantile function are given by `rexp`, `pexp`, `qexp` respectively.

Gamma

The Gamma or Erlang distribution of order a measures the distribution of the continuous random variable X, which is the time until r randomly generated events with rate λ have occurred. Its probability density function is

$$f_{\lambda,\alpha}(x) = \frac{\lambda(\lambda x)^{\alpha-1} e^{-\lambda x}}{\Gamma(\alpha)}, \quad x \geq 0,\ \lambda > 0,\ \alpha > 0 \tag{1.22}$$

is available in R through `dgamma(x, shape=`α`, rate=`λ`, scale=`$\frac{1}{\lambda}$`, log=FALSE)`

It is used in GLM for the `Gamma(link="inverse")` family.

The random generation, distribution function and quantile function are given by `rgamma`, `pgamma`, `qgamma` respectively.

Cauchy

The Cauchy distribution is often used in statistics as the canonical example of a "pathological" distribution since both its expected value and its variance are undefined, i.e. the Cauchy distribution does not have moment generating function because it does not have finite moments of order greater than or equal to one:

$$f_{x_0,s}(x) = \frac{s^2}{\pi(s^2 + (x - x_0)^2)} \tag{1.23}$$

is available in R through `dcauchy(x, location=`x_0`, scale=`s`, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rcauchy`, `pcauchy`, `qcauchy` respectively.

Weibull

If Weibull distribution random variable X describes the failure rate is proportional to a power of time.

$$f(x) = (\frac{a}{b})(\frac{x}{b})^{a-1}e^{-(x/b)^a}, \quad a > 0,\ b > 0,\ x \leq 0 \tag{1.24}$$

is available in R through `dweibull(x, shape=`a`, scale=`b`, log=FALSE)`

The random generation, distribution function and quantile function are given by `rweibull`, `pweibull`, `qweibull` respectively.

1.11.3 Sampling Distributions

χ^2, t and F distributions are sampling distributions associated with the standard normal distribution (1.19).

Chi-Square

When the normal random variables X_i are i.i.d., the random variable $X_1^2 + X_2^2 + \cdots + X_\nu^2$ is called a χ^2 random variable with a degree of freedom ν and it has a probability distribution

$$f_\nu(x) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}x^{\nu/2-1}e^{-x/2}, \quad x > 0 \tag{1.25}$$

is available in R through `dchisq(`x`, df=`n`, ncp=0, log=FALSE)`

The random generation, distribution function and quantile function are given by `rchisq`, `pchisq`, `qchisq` respectively.

Student t

The Student's t distribution with a degree of freedom ν is usually used to estimate the mean of a normally distributed population by using the small portion of the data from its population and has a probability density function

$$f_\nu(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\pi\nu}\Gamma(\frac{\nu}{2})}(1 + \frac{x^2}{\nu})^{-(\nu+1)/2}, \quad \nu > 0 \tag{1.26}$$

is available in R through `dt(`x`, df=`ν`, ncp, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rt`, `pt`, `qt` respectively.

F

F distribution is the sampling distribution that describes the ratio of two χ^2 distributions with the degree of freedoms ν_1 and ν_2 respectively:

$$F(\nu_1, \nu_2) = \frac{\chi_1^2(\nu_1)}{\nu_1} / \frac{\chi_2^2(\nu_2)}{\nu_2}$$

has a probability density function:

$$f_{\nu_1,\nu_2}(x) = \frac{\Gamma(\frac{\nu_1+\nu_2}{2})}{\Gamma(\frac{\nu_1}{2})\Gamma(\frac{\nu_2}{2})}\left(\frac{\nu_1}{\nu_2}\right)^{\nu_1/2} x^{\nu_1/2-1}\left(1 + (\frac{\nu_1}{\nu_2})x\right)^{-(\nu_1+\nu_2)/2} \tag{1.27}$$

is available in R through `df(x, df1=`ν_1`, df2=`ν_2`, ncp, log=FALSE)`.

The random generation, distribution function and quantile function are given by `rf`, `pf`, `qf` respectively.

1.11.4 Higher Dimensional Distributions

- `mvtnorm`: Provides the functions to compute multivariate normal and t probabilities, quantiles, random deviates and densities. It is used in C5.0 decision tree. E.g. `dmvnorm(x, mean=rep(0,`p`), sigma=diag(`p`), log=FALSE, checkSymmetry=TRUE)`, `rmvt(n, sigma=diag(2), df=1, delta=rep(0, nrow(sigma)), type=c("shifted", "Kshirsagar"), ...)`

- `mnormt`: Provides the probability density function `dmnorm(x, mean=rep(0,`d`), varcov, log=FALSE)`, `dmt(x, mean=rep(0,`d`), S, df=Inf, log=FALSE)`, the distribution function `pmnorm(x, mean=rep(0,`d`), varcov, ...)`, `pmt(x, mean=rep(0,`d`), S, df=Inf, ...)` and random number generation `rmnorm(n=1, mean=rep(0,`d`), varcov, sqrt=NULL)`, `rmt(n=1, mean=rep(0,`d`), S, df=Inf, sqrt=NULL)` for a d-dimensional multivariate normal (Gaussian) random variable and a d-dimensional multivariate Student's t random variable..

- `LaplacesDemon`: Provides the density function `ddirichlet(x, alpha, log=FALSE)` and random generation `rdirichlet(n, alpha)` from the Dirichlet distribution.