Topic 1: Overview

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

Terminologies of similar meaning: ***Statistical learning***, ***machine learning***, `https://en.wikipedia.org/wiki/Predictive_analytics`.
Relevant Topics (To cover: ✓; Not cover: ✗):

- Supervised Learning Models:
  - Classifiers: kNN ✓, logistic regression ✓, neural network ✗, Naive Bayes ✓, LDA ✓, linear support vector machine (SVM), ✗, kernel SVM ✗, classification trees ✓, ensemble methods ✗, ...
  - Regressors: kNN ✓, linear regression ✓, regression tree ✓, support vector regressor (SVR ✗), ...
- Unsupervised Learning Models:
  - Feature agglomeration ✗
  - Random projections ✗
  - Dimensional Reduction: PCA ✓, t-SNE (`https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding`) ✗, ...
  - 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 ✗
- Time-series data mining: Time series data is time-stamped and collected over time at a particular interval (sales in a month, calls per day, web visits per hour, etc.). Time series data mining combines traditional data mining (such as sampling, clustering and decision trees) and forecasting techniques. ✗

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 + ... → Anaconda Python
- Java + NoSQL + Weka + SparkML + ...
- Excel, Power BI, Microsoft Access, SQL in business environment

Popular data:

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

Textbooks:

- Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, An Introduction to Statistical Learning: with Applications in R, 2nd ed., Springer 2021 `https://statlearning.com/` (Second edition is available for download)
- Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer-Verlag, 2008 `https://hastie.su.domains/ElemStatLearn/`

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 **2025.01 ... Channel** rather than the General channel)

Syllabus:

- Lecture (22 hours) : Friday classes are not affected by public holiday.
- Tutorial (11 hours) : All tutorial classes (Wednesday & Friday) are not affected by public holiday.
- Practical (12 hours) :
  - P2 (Tuesday 4-5pm; two students only. If possible, please change to Wednesday 3-4pm)
  - P1 and P2 classes (Tuesday) are affected by public holidays — Week 1 (Thaipusam), Week 6 (Nuzul Al-Quran), Week 8 (Hari Raya Puasa)

Class Arrangements:

Week 1:

- Start of tutorial and practical (we try to complete class by Week 12/13)
- Week 1 Tuesday is a Thaipusam holiday, a pre-recorded video will be prepared for P1 and P2. You may join the Wednesday 3-4pm class at KB904 if you prefer physical class. Please remember to scan the QR code.
- Start to form assignment group, 3–5 members in a group. Individuals without any group by week 3 may be assigned to groups with less than 5 members by lecturer

Week 2:

- Hopefully the assignment is ready for review.

Week 3:

- 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:

- Tuesday is Nuzul Al-Quran holiday, a pre-recorded video will be prepared for P1 and P2. You may join the Wednesday 3-4pm class at KB904 if you prefer physical class.
- **Quiz** covering practical 1, practical 2, topic 1 to topic 3 (logistic regression).

Week 8:

- Tuesday is Hari Raya Puasa holiday. The university has announced Week 8 as online learning. So I will prepare the pre-recorded videos for lecture, tutorial and practical. Please scan the QR code before 2 pm Friday. If you prefer online class for interaction, let me know.

Week 11:

- Wednesday (23 Apr) 6pm: Deadline for the submission of assignment report and computer program in correct format to lecturer by group leader.
- All tutorials should be completed this week.

Week 12:

- Physical Oral presentation (is it possible to start in Week 11?)

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 marks (in MS Teams's 2025.02 Channel) and inform the lecturer if there are any mistakes.

Assessments:

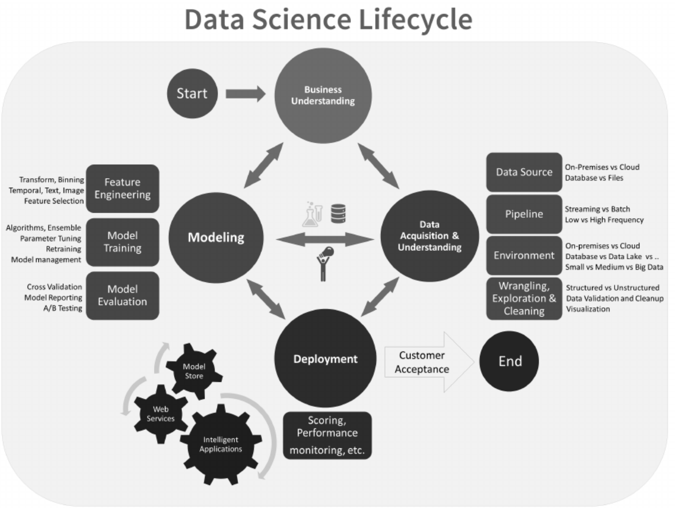
- **Practical Quiz** (12%, CO4):
  - A sample quiz from past semester is given.
- **Assignment** (38%):
  - Report 18% (CO1 + CO2 + CO3)
  - Programming Code 10% (CO4)
  - Oral Presentation 10% (CO5: need to submit presentation slides)
- **Final Exam** (100 marks ×0.5 ⇒ 50%): 3 + 1 Questions.
  - Q1: 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

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
  - Time series: Stock price; Online game control sequence; Industrial robot control sequence; etc.

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

To learn more about the **structured data**, we perform `https://en.wikipedia.org/wiki/Exploratory_data_analysis` (EDA) using (i) Summary statistics (in numeric or tabular forms); (ii) Data visualisation.

Standard data types for a column in the tabular data:

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

Univariate data analysis:

- Numeric summary:
  - For categorical data (or some integral data): R's `table()`.
  - For integral data and continuous data: R's `mean()`, `median()`, `var()`, `sd()`, etc.
  - In general, R's `summary()`, Python's `df.describe()`, `scipy.stats.describe` provide automatic summary of data based on their datatype.
- Data Visualisation:
  - Categorical and ordinal data: pie chart, barplot / bar graph
  - Numerical data: histogram (R's `hist()`, Python's `plt.hist()`), stem-and-leaf plot (R's `stem()`), boxplot, density plot, `qqplot()`

Bivariate data 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 tries to convert an unstructured data to a **structured** data or from a structured data to a clean (and may be scaled) structured data.

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

1.6.1 Preprocessing Text Data

- 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

Image is extremely difficult to process because the colours, brightness, resolution will make the same image having rather different 3D arrays.

It is quite common for colour images to be reduced to 2D gray scale images for image preprocessing involving the following steps:

- Deblurring, denoising, contrast enhancement, ...
- Cropping
- Rescaling (with Gaussian smoothing)
- Projective transforms: translation and rotation

R has a package `magick` which uses the C++ ImageMagick library for image processing while Python has much more image processing libraries:

- 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

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

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

- **Standardisation** (column scaling): `scale(d.f)`. It is a common requirement for many machine learning estimators; they might behave badly if the individual features do not more or less look like standard normally distributed data: Gaussian with zero mean and unit variance.
$$M(X_{ij}) = \frac{X_{ij} - \overline{X}_j}{s_{X,j}} \tag{1.1}$$
where
$$s_{X,j} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_{ij} - \frac{1}{N}\sum_{i=1}^{N} X_{ij})^2}.$$
- Min-max scaling
$$M(X_{ij}) = \frac{X_{ij} - X_{\text{min,j}}}{X_{\text{max,j}} - X_{\text{min,j}}}. \tag{1.2}$$
- Generating polynomial features: `poly()`
- Non-linear and custom transformation
- Dimensional reduction
- Discretization: `arules::discretize`
- **Normalisation** (row scaling): It scales individual samples to have unit norm. This process can be useful if we plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

1.6.5 Min-Max Normalisation (Rescaling)

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

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

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

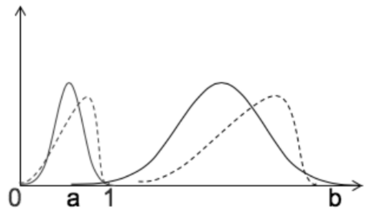
In Python, we can use numpy array with

```
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.



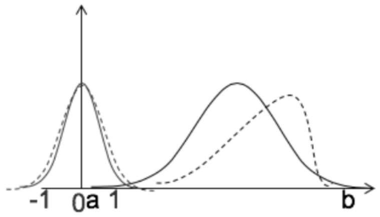
1.6.6 Standard Scaler/Standardisation

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

Software Implementations

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

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



Example 1.6.1 (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.

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

Table 1.1: Two-dimensional data.

(6 marks)

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

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

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

and the table after standardisation is .... [3 marks]

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

1.6.7 Data Cleaning/Cleansing: Imputation of Missing Values

Whether the real-world data comes from the 'databases' or from Excel files or Internet, there may be **noise** (due to measurement), **missing values**, **wrong values** (e.g. height is 16cm instead of 160cm), etc. To clean up the data:

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

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

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

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

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. > 50%), 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 are spreaded everywhere, we use missing value heatmap to try to identify some patterns and decide on how to impute the data if necessary. E.g.
    - ∗ for many predictive models such as logistic regression, naive Bayes, etc. we need to impute the data;
    - ∗ for CART decision tree, we may not need to impute the data

1.7 Modelling

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

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

1.7.1 Modelling Unlabelled Data

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

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

Unsupervised Learning methods below could be tried:

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

1.7.2 Modelling 'Continuous' Labelled Data — Regression Problems

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

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

- Linear regression models:

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

- Nonlinear regression models:

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

Exercise 1.7.1 (Simple Linear Regression)**.** If the linear regression (a kind of supervised 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: $\hat{\beta}_1 = \frac{n\sum_{i=1}^n x_iy_i - (\sum_{i=1}^n x_i)(\sum_{i=1}^n y_i)}{n(\sum_{i=1}^n x_i^2) - (\sum_{i=1}^n x_i)^2}$, $\hat{\beta}_0 = \overline{y} - \hat{\beta}_1\overline{x}$.

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

A good classifier is a mathematical model $Y = h(X)$ that fits the 'discrete' labelled data (coming from survey data or experimental observations in a tabular form which have been cleaned and pre-processed):

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

such that the total loss $\sum_i L(y_i, h(\boldsymbol{x}_i))$ is acceptably small. Here L is the loss function.

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

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

The model h may be used for

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

1.7.4 Approaches to Classifying/Developing 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 $X_1, ..., X_p$.

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

  E.g. linear regression, logistic model, kNN, 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. Note that generative models in effect requires the joint distribution over all variables $\mathbb{P}(X_1, \cdots, X_p, Y)$.

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

  E.g. Naive Bayes, LDA, etc.

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

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**.

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.

1.7.5 Model Training and Prediction

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

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

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

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

- R's training and prediction process:

```
lrmodel = glm(y ~ ., data=X, family=binomial)
# glm is the learning algorithm for logistic regression
# X  is the data
# y ~ . is the model specification of output y against the rest .
Yhat = predict(lrmodel, newdata=data.frame(x1=...,x2=...))
# We can put in new data and ask the trained model to predict
```

- Python's training and prediction process:

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

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

- 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.
- 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.

1.7.6 Flexibility vs Interpretability

The reason we need to develop so many predictive models is because some of them are more flexible while some of them are more interpretable. 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 polynomial regression 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 polynomial regression of degree 2 which is more flexible, but it could only fit data which are convex in some range:

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

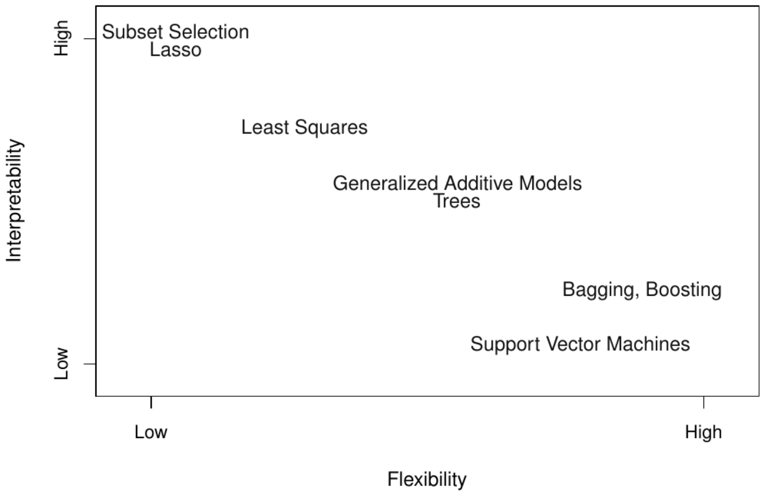
The model is slightly more difficult to interpret: The input x will influence y at a "quadratic manner" when $|x| > 1$ but when $|x| \ll 1$, then y has little changes.

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. An illustration of some predictive models based on flexibility and interpretability are given below James et al. [2013]:



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

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.

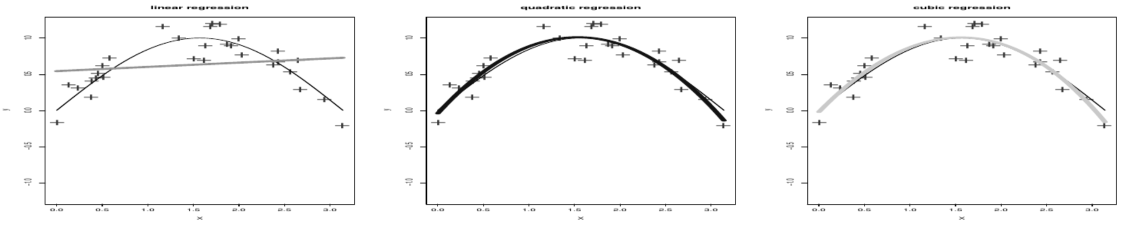
Example 1.7.6. Suppose the output y is governed by the input x following the equation:

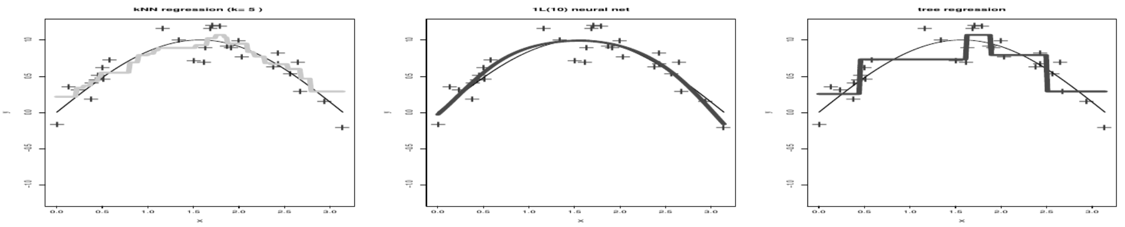
$$y = \sin(x) + R, \quad R \sim Normal(0, 0.2^2)$$

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.

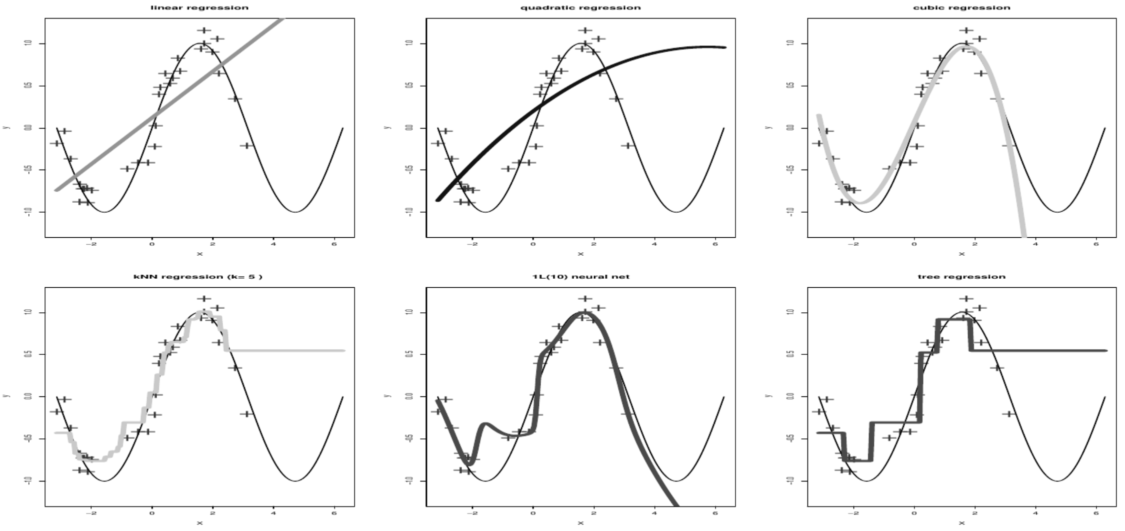




Things to consider: Flexibility (more parameters, model more complex) vs Interpretability (less parameters, model simpler)

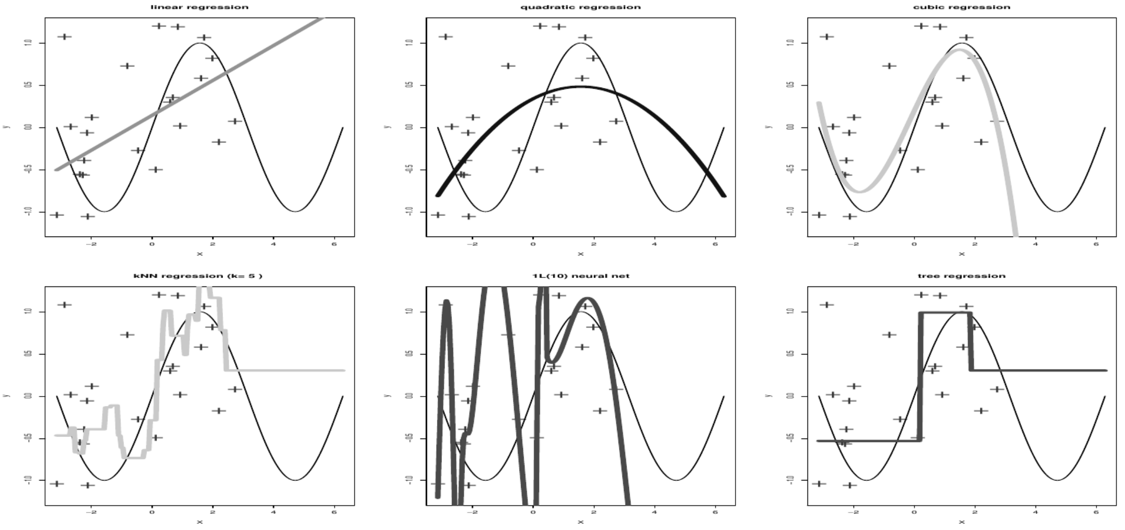
- Inflexible ⇒ Simpler math formula ⇒ Poorer Predictibility, better inference(?)
- Flexible ⇒ Complicated math formula ⇒ Good Predictibility, poorer inference(?) □

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):

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



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

1.8 Evaluation

Unsupervised Learning:

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

Supervised Learning:

- We have output Y associated with an input X, so we can put in the input of an observation $\boldsymbol{x}_i$ to the predictive model $h(\boldsymbol{x}_i)$ and **compare** it to the actual observed label y_i;
- The 'theoretical' measurement of the **difference** between a true model Y and a predictive model $h(X)$ is called a **generalisation error** or an *expected new data error* [Lindholm et al., 2022].
- Let $y = f(x_1, \cdots, x_1)$ be the theoretical actual model, ϵ is a noise with zero mean and $\widehat{y} = h(x_1, \cdots, x_p)$ be the predictive model. The **theoretical evaluation metrics** and the **empirical evaluation metrics** for supervised regression and supervised classification are listed below:

  Regression:

  - Theoretical Mean Squared Error (MSE):
  $$\mathbb{E}_X \mathbb{E}_\epsilon[((f(X) + \epsilon) - h(X))^2].$$
  - (Empirical) Sum of Squared Error (SSE) / (Empirical) Residue Sum of Squares (RSS) and (Empirical) MSE:
  $$SSE = \sum_{i=1}^{n} \Big(y_i - \widehat{y}_i\Big)^2, \quad MSE = \frac{SSE}{n}, \quad L(y_1, y_2) = (y_1 - y_2)^2.$$
  - (Empirical) Mean Absolute Error (MAE):
  $$MAE = \frac{1}{n} \sum_{i=1}^{n} \Big| y_i - \widehat{y}_i \Big|, L(y_1, y_2) = |y_1 - y_2|.$$
  - The **coefficient of determination**, R^2, is a statistical measure that represents the proportion of the variance for a dependent variable (the output) that is explained by an independent variable in (1.3), i.e.
  $$R^2 = 1 - \frac{\sum(y_i - \widehat{y}_i)^2}{\sum(y_i - \frac{1}{n}\sum_i y_i)^2} = 1 - \frac{SSE}{SST}.$$

  Classification:

  - Error rate: The loss function is the ***misclassification error*** $I(y \neq \hat{y})$ and the "error rate" **cost function** (or **objective function**) is the "proportion of mistakes" that are made if we apply our model $\hat{f}$ to the data:
  $$\frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i), \quad L(y_1, y_2) = I(y_1 \neq y_2) \tag{1.4}$$
  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 the complement of error rate with a formula $\frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{y}_i)$.

Note that the `https://en.wikipedia.org/wiki/Loss_functions` $L(y_1, y_2)$ mentioned above are not comprehensive. 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. They are explained in Lindholm et al. [2022, Chapter 5].

In the supervised learning process, when the *training data* is employed, the error is called a *training error*; when the *testing/new data* is employed, the error is called a *testing error.* Testing error is usually used to estimate the generalisation error. The difference between the generalisation error and the testing error is called the *generalisation gap* [Lindholm et al., 2022].

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

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

1.8.1 Performance Evaluation Examples for Regression Problems

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

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

Calculate the MSE, MAE and R^2.

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 1.8.2 (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 1.8.3 (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.)

1.8.2 Performance Evaluation Examples for General Classification Problems — Confusion Matrix

The performance of a predictive model or a "classifier" can be checked based on their prediction accuracy. Suppose that we seek to estimate f on the basis of training observations $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_n, y_n)\}$, where $y_1, \cdots, y_n$ are **qualitative**. The most common approach for quantifying the accuracy of the "estimated" predictive model $\hat{f}$ is the ***test error rate***, i.e. the misclassification error (1.4) for test data. An “acceptable” classifier is one for which the test error (1.4) is smallest.

The ***confusion matrix*** (or ***contingency table***) is matrix or table that summarises the number of correct and incorrect classifications and it provides more information than the misclassification error (1.4).

Example 1.8.4. 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 1.8.5 (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

1.8.3 Performance Evaluation for Binary Classification Problems

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Example 1.8.6 (Scenario 1: The Fraudulent Loans)**.** A Malaysian bank operates a large personal loan business. This product helps their customers enjoy better cash liquidity, e.g. purchase big ticket items or take a family holidays. However, each lending business comes with a risk of revenue losses due to credit default (i.e. who failed to repay the loan).

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

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

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

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

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

Solution: The accuracy measures are calculated as follows.

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

1.8.4 More Performance Evaluations for Binary Classification Problems

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

Example 1.8.7. Apply the ROC analysis on the ISLR's Smarket data using the `pROC` package.

Solution:

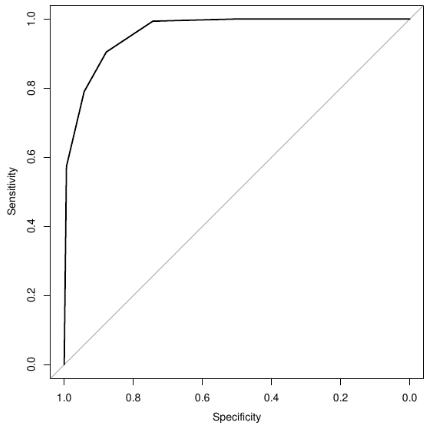
```
library(ISLR)
Smarket = Smarket[,-1]  # Remove Year
N = nrow(Smarket)

```

```
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 1.8.8 (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 1.8.9 (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 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`.

1.8.5 The Holdout / Validation Set Approach

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

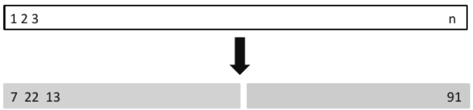
Why is overfitting a problem?

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

The ***holdout method***, ***split validation*** or ***validation set approach*** [Coelho and Richert, 2015, Chapter 2], is the most basic strategy to **estimate** the accuracy with fitting a particular statistical learning method on a set of observations which is **time-independent** (e.g. images but not time-dependent economics data). It involves randomly dividing the available set of observations into two parts [James et al., 2013, Section 5.1]:

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

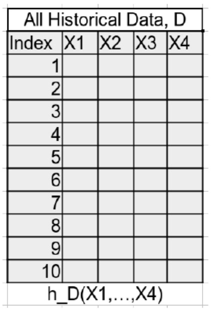
A schematic display of the validation set approach is shown in the following figure.



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

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

Example 1.8.10 (Linear Sampling)**.** Consider a 4-dimensional data D:



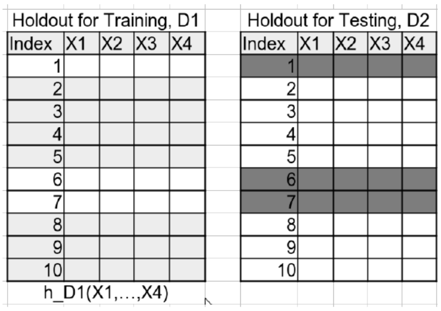
All Historical Data, D

Index	X1	X2	X3	X4
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
h_D(X1,...,X4)				

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

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

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



Holdout for Training, D1

Index	X1	X2	X3	X4
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
h_D1(X1,...,X4)				

Holdout for Testing, D2

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

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

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 1.8.11 (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.

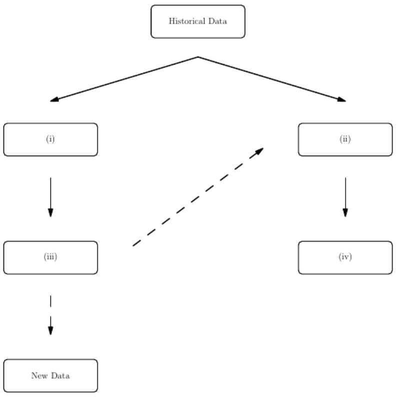
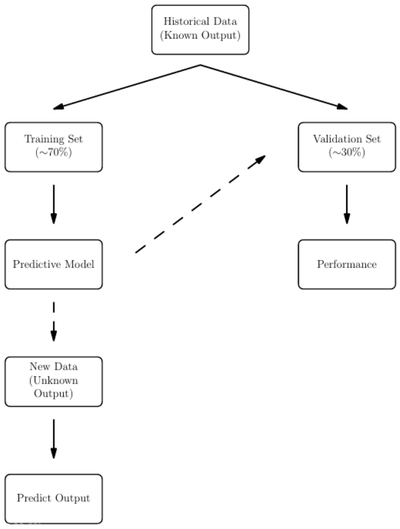


Figure Q3(a)



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 ± 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.

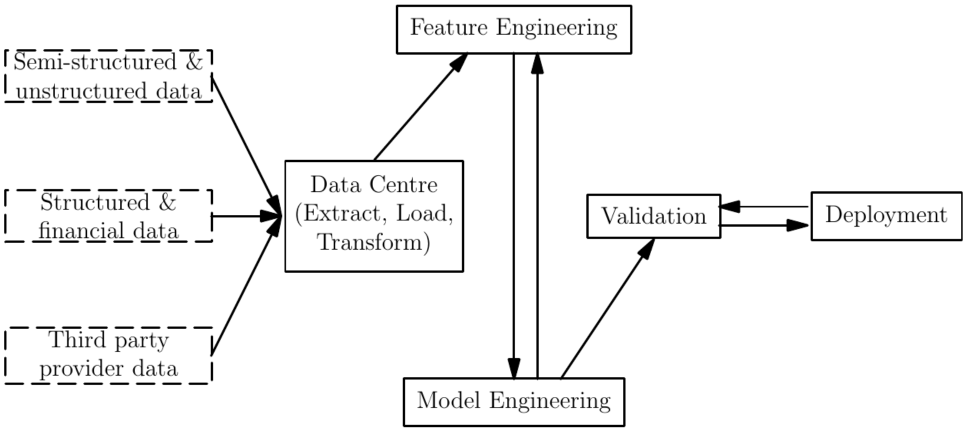
1.9 Deployments

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 × 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 web-server, data analytical software, etc.	slightly lower because a smaller IT team is required to maintain the softare 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.

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("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

$$f_{n,p}(x) = \binom{n}{x} p^x (1-p)^{n-x}, \quad 0 \le p \le 1,\ x = 0, 1, ..., n \tag{1.5}$$

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.

$$f_{n,p}(x) = \frac{\Gamma(x+n)}{\Gamma(n)x!} p^n (1-p)^x, \quad 0 < p \le 1,\ x = 0, 1, ..., n \tag{1.6}$$

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.

$$f_\lambda(x) = \frac{\lambda^x e^{-\lambda}}{x!}, \quad \lambda > 0,\ x = 0, 1, 2, ... \tag{1.7}$$

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:

$$f_p(x) = (1-p)^x p, \quad 0 \le p \le 1, \; x = 0, 1, 2, 3, ... \tag{1.8}$$

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.

$$f_{m,n,k}(x) = \binom{m}{x}\binom{n}{k-x} / \binom{m+n}{k}, \quad x = 0, ..., k \tag{1.9}$$

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:

$$f_{a,b} = \begin{cases} \frac{1}{b-a}, & a \le x \le b \\ 0, & \text{otherwise} \end{cases} \tag{1.10}$$

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:

$$f_{\alpha,\beta}(x) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{\alpha-1}(1-x)^{\beta-1}, \quad \alpha > 0, \; \beta > 0, \; 0 \le x \le 1 \tag{1.11}$$

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.

$$f_{\mu,s}(x) = \frac{e^{-\frac{x-\mu}{s}}}{s(1+e^{-\frac{x-\mu}{s}})^2} = \frac{1}{4s}\text{sech}^2\left(\frac{x-\mu}{2s}\right) \tag{1.12}$$

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:

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp(-\frac{(x-\mu)^2}{2\sigma^2}) \tag{1.13}$$

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:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x}\exp(-\frac{(\ln x-\mu)^2}{2\sigma^2}) \tag{1.14}$$

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.

$$f_\lambda(x) = \lambda e^{-\lambda x}, \quad x \geq 0 \tag{1.15}$$

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

$$f_{\lambda,\alpha}(x) = \frac{\lambda(\lambda x)^{\alpha-1}e^{-\lambda x}}{\Gamma(\alpha)}, \quad x \geq 0,\ \lambda > 0,\ \alpha > 0 \tag{1.16}$$

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:

$$f_{x0,s}(x) = \frac{s^2}{\pi(s^2 + (x - x_0)^2)} \tag{1.17}$$

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.

$$f(x) = (\frac{a}{b})(\frac{x}{b})^{a-1}e^{-(x/b)^a}, \quad a > 0,\ b > 0,\ x \leq 0 \tag{1.18}$$

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 (1.13).

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

$$f_\nu(x) = \frac{1}{2^{\nu/2}\Gamma(\nu/2)}x^{\nu/2-1}e^{-x/2}, \quad x > 0 \tag{1.19}$$

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

$$f_\nu(x) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\pi\nu}\Gamma(\frac{\nu}{2})}(1 + \frac{x^2}{\nu})^{-(\nu+1)/2}, \quad \nu > 0 \tag{1.20}$$

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:

$$f_{\nu_1,\nu_2}(x) = \frac{\Gamma(\frac{\nu_1+\nu_2}{2})}{\Gamma(\frac{\nu_1}{2})\Gamma(\frac{\nu_2}{2})} \left(\frac{\nu_1}{\nu_2}\right)^{\nu_1/2} x^{\nu_1/2-1} \left(1 + (\frac{\nu_1}{\nu_2})x\right)^{-(\nu_1+\nu_2)/2} \tag{1.21}$$

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.