Week2:

Types of data: Nominal (=, !=): Distinctness. **Categorical** labels without order. E.g. ID numbers, eye color, zip codes.

Ordinal Scale (<, >): Order, **no equal intervals** (E.g. taste of potato chips on a scale from 1-10), grades, height {tall, medium, short}. (Central tendency can be measured by mode or median. The mean and stdev cannot be defined from an ordinal set.)

Interval Scale(+, -): meaningful difference. Order, equal intervals, **no true zero**. Addition & subtraction. E.g. calendar dates, temperatures in Celsius or Fahrenheit.

Ratio Scale (*, /): **meaningful ratios**. Order, equal intervals, true zero. E.g. temperature in Kelvin, length, counts, elapsed time multiplication & division (e.g., time to run a race).

Central tendency can be measured by **mode, median, or mean**.

The **median** is robust to outliers and better represents the central tendency for skewed data. The **mode** only reflects the most frequently occurring value and does not consider the overall distribution. The **mean** is sensitive to outliers in the tail of the distribution. Left-Skewed Distribution: The tail extends to the left (negative skew). **Relationship**: Mean < Median < Mode

Dispersion can be estimated by the Inter-Quartile Range **(IQR), stdev, variance**

Calculating descriptive statistics: **Median** is the middle value (or average of two middle values). **Minimum** is the first value. **Maximum** is the last value. **10th** percentile is item at index 0.1*N. **90th** percentile is item at index 0.9*N. **Range** is Maximum minus Minimum. **IQR** is the difference between the first and third quartile.

How to calculate the IQR:

the 'cut-off' points are called **quartiles**

[1,1,2,2,2,2,2,2,2,2,3,3,3, 3] [3,3,3,3,3,3,3,3,3,3,3,3,3, 3]
[3,3,3,3,3,4,4,4,4,4,4,4,4,4, 4] [4,4,4,4,4,4,4,4,5,5,5,5,5, 5]

– i.e. Q3 – Q1 = 4 – 3 = 1.

Mean is the sum of values divided by the number of values: $\frac{\sum X_i}{N}$

Variance: $\frac{\sum(X_i - mean)^2}{N-1}$

Standard deviation: $\sqrt{variance}$

text data: Requires interpretation, **coding or conversion**.

Levels of Measurement

	Nominal	Ordinal	Interval	Ratio
Countable	✓	✓	✓	✓
Order defined		✓	✓	✓
Difference defined (addition, subtraction)			✓	✓
Zero defined (multiplication, division)				✓

Measures of Central Tendency

	Nominal	Ordinal	Interval	Ratio
Mode	✓	✓	✓	✓
Median		✓	✓	✓
Mean			✓	✓

Measures of Dispersion

	Nominal	Ordinal	Interval	Ratio
Counts / Distribution	✓	✓	✓	✓
Minimum, Maximum		✓	✓	✓
Range		✓	✓	✓
Percentiles		✓	✓	✓
Standard deviation, Variance			✓	✓

Data acquisition and cleaning:

Cleaning and Transforming Data

Why: Real data is often 'dirty'. **So**: Important to do some data cleaning and transforming first. **Typical steps**: 1. Type and name conversion. 2. Filtering of missing or inconsistent data. 3. Unifying semantic data representations. 4. Matching entries from different sources. 5. (Later also) Rescaling and optional dimensionality reduction.

Week3:

Using boxplots

Mean and stdev are **not informative when data is skewed**.

Box plots summarise data based on **5 numbers**:
– Lower inner fence – Q1–1.5*IQR.
– First quartile (Q1) – equivalent to 25th percentile.
– Median (Q2) – equivalent to 50th percentile.
– Third quartile (Q3) – equivalent to 75th percentile.
– Upper inner fence – Q3+1.5*IQR.
– IQR = Q1-Q3

Values outside fences are outliers.

Sometimes include outer fences at Q1-3*IQR and Q3+3*IQR.

Week4:

Relational databases: In the **relational model**, a database is a collection of one or more **relations**. Each relation is a table with rows and columns. Each relation has a schema, which describes the columns, or fields. -**Relational Model** defines the structure and relationships between tables using primary and foreign keys.

Definition of relation: A relation is a named, two-dimensional table of data

A relation schema specifies name of relation, and name and data type (domain) of each attribute.

$R = (A_1, A_2, \ldots, A_n)$ is a ***relation schema***

- $A_1, A_2, \ldots, A_n$ are ***attributes, each having a data type***

e.g. `Student(sid: string, name: string, login: string, addr: string, gender: char)`

A relation instance is a **set** of tuples (a table) for a schema

Some remarks

Not all tables qualify as a relation:
–Every relation must have **a unique name**.
– **Attributes (columns) in tables must have unique names**.
•The order of the columns is irrelevant.
– **All tuples in a relation have the same structure**
• Constructed from the same set of attributes.
–Every attribute value is atomic (not multivalued, not composite).
–**Every row is unique**.
• Can't have two rows with exactly the same values for all their fields.
–**The order of the rows is immaterial**.

RDBMS table extends mathematical relation
– RDBMS allows **duplicate rows**, supports an **order of tuples or attributes**, and allows **null 'values'** for unknown information

SQL – The Structured Query Language

Supported commands from roughly two categories:

– **DDL** (Data Definition Language).
- Create, drop, or alter the relation schema.
- Example: `CREATE TABLE name (list_of_columns)`

– **DML** (Data Manipulation Language).
- For retrieval of information also called query language.
- `INSERT, DELETE, UPDATE`
- `SELECT … FROM … WHERE`

Creating Tables

Creation of tables/relations:

```
CREATE TABLE name ( list-of-columns );
```

– Example: Create the Student table.

```
CREATE TABLE Student (sid INTEGER,
                      name VARCHAR(20),
                      login VARCHAR(20),
                      gender CHAR,
                      address VARCHAR(50) );
```

Insertion of new data

Syntax: `INSERT INTO table (list-of-columns) VALUES (list-of-expression)`

Example:
```
INSERT INTO Students (sid, name)
VALUES (53688, "Smith")
```

Updating of tuples

Syntax:
```
UPDATE table
SET column = expression
WHERE search_condition
```

Example:
```
UPDATE students
SET gpa = gpa - 0.1
WHERE gpa >= 3.3
```

Deleting of tuples

Syntax: `DELETE FROM table WHERE search_condition`

Example: `DELETE FROM students WHERE name = "Smith"`

Integrity constraints

A variety of rules to maintain the integrity of data when it is manipulated
The rule must be satisfied for **any** instance of the database; e.g., **domain constraints**.
Integrity constraints are declared **in the schema**
A legal instance of a relation is one that satisfies all specified integrity constraints.
-Referential integrity ensures that relationships between tables remain valid and consistent. It enforces rules for foreign keys to maintain data accuracy.

Domain constraints

Domain constraints **restrict attributes to valid domains**.
– **NULL/NOT NULL**: whether an attribute is allowed to become NULL (unknown).
– **DEFAULT**: to specify a default value.
– **CHECK(condition)**: a Boolean condition that must hold for every tuple in the DB instance.

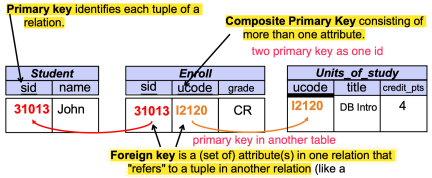
Example:

```
CREATE TABLE Student
(
  sid          INTEGER          PRIMARY KEY,
  name         VARCHAR(20)      NOT NULL,
  gender       CHAR             CHECK (gender IN ('M','F','T')),
  birthday     DATE             NULL,
  country      VARCHAR(20),
  level        INTEGER          DEFAULT 1 CHECK (level BETWEEN 1 and 5)
);
```

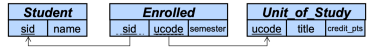
Relational keys

Key: unique, minimal identifier of a relation.

Foreign keys: identifiers that enable a dependent relation to refer to its parent relation



Example: Primary & Foreign Keys




```
CREATE TABLE Student ( sid INTEGER, ... ,
    CONSTRAINT Student_PK PRIMARY KEY (sid)
);                          Custom name
CREATE TABLE UoS  ( ucode CHAR(8), ... ,
    CONSTRAINT UoS_PK PRIMARY KEY (ucode)
);
CREATE TABLE Enrolled ( sid INTEGER, ucode CHAR(8), semester VARCHAR,
    CONSTRAINT Enrolled_FK1 FOREIGN KEY (sid) REFERENCES Student,
    CONSTRAINT Enrolled_FK2 FOREIGN KEY (ucode) REFERENCES UoS,
    CONSTRAINT Enrolled_PK  PRIMARY KEY (sid,ucode)
);
```

Key constraint: No two distinct tuples can have the same values in all key attributes

Example:

```
CREATE TABLE Enrolled  (
  sid   INTEGER,
  cid   CHAR(8),
  grade CHAR(2),
  PRIMARY KEY (sid,cid) );
```

"For a given student and course, there is a single grade."

```
CREATE TABLE Enrolled  (
  sid   INTEGER,
  cid   CHAR(8),
  grade CHAR(2),
  PRIMARY KEY (sid)
  UNIQUE (cid, grade) );
```

"Students can take only one course and receive a single grade for that course; further, no two students in a course receive the same grade."

Foreign Key Constraint

-For **each tuple in the referring relation** whose foreign key value is "a", there **must be a tuple in the referred relation** with a candidate key that **also has value "a"**

Keys and NULLs

Primary KEY: Up to one per table, and must be **unique**. Automatically **disallow NULL values**

Unique (candidate key): Possibly many candidate keys (specified using UNIQUE)

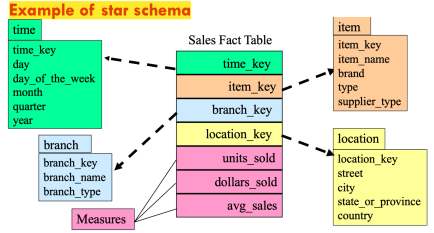
Foreign KEY: By default, **allows NULL values**. If there must be a parent tuple, then must combine with NOT NULL constraint

Conceptual modeling of data warehouses

dimensions & measures instead of relationalmodel. Data warehouse contains a **large** central table **(fact table).** Contains the data **without redundancy.** A set of **dimension tables**.

Star schema

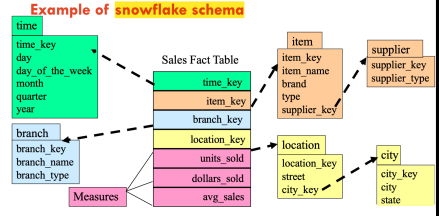
Example of star schema



Snowflake schema

A refinement of a star schema where **some dimensional hierarchy is normalized into a set of smaller dimension tables**, forming a shape similar to a snowflake.

Example of snowflake schema



Week5:

The SELECT – FROM – WHERE command

Example 1: Which station commence after 1900-1-1?

```
SELECT siteName, commence, orgcode  returned features
FROM station
WHERE commence > '1900-1-1';
```

Example 2: How many measurements have we done?

```
SELECT COUNT(*) FROM Measurement;
```

Example 3: List top five measurements ordered by date in descending order.

```
SELECT * FROM Measurement
ORDER BY date DESC limit 5;
```

SQL data types

Integers, Floats, Numeric, Date, Timestamp

Strings (CHAR, VARCHAR): CHAR: fixed length; VARCHAR: variable length strings up-to max length.

Comparison operations in SQL

Example 1:

```
SELECT *
FROM TelescopeConfig
WHERE( mindec BETWEEN -90 AND -50 )
      AND ( maxdec >= -45 )
      AND ( tele_array = 'H168' );
```

Example 2:

```
SELECT *                          模糊查询
FROM TelescopeConfig              _单个任意字符
WHERE tele_array LIKE 'H%';       %：多个任意字符
                                  表示查询tele_array中首字母为 H, 后面为任意多个字符
```

Date and time in SQL

CURRENT_DATE: db system's current date. CURRENT_TIME: db system's current timestamp.

Example:

```
SELECT *
FROM Epoch
WHERE startDate < CURRENT_DATE;
```

Main Operations

– **EXTRACT**(component **FROM** date).
- e.g., **EXTRACT**(year **FROM** startDate)

– **DATE** string (Oracle syntax: **TO_DATE**(string,template))
- e.g., **DATE** '2012-03-01'
- Some systems allow templates on how to interpret string.
- Oracle syntax: **TO_DATE**('01-03-2012', 'DD-Mon-YYYY')

+/- INTERVAL

- e.g. '2012-04-01' + **INTERVAL** '36 HOUR'

JOIN: Querying multiple tables

Join 适合两个表中有多个相同列,而 Neutral join 适合两个表只有一个相同列

Examples: 最终的结果集包含了所有可能的站台和组织的组合。

– Produces the **cross-product** Station x Organisation:

```
SELECT *
FROM Station, Organisation;
```

– Find the site name, commence date and organisation name of all stations:

```
SELECT sitename, commence, organisation
FROM Station, Organisation   这意味着只有当Station表中的orgcode与Organisation表中
WHERE orgcode = code;        的code相同的时候，相关的数据才会被选中。
```

查询的是select中的三个列

Examples:

– List all details of the first three measurements including Water data:

```
SELECT *  这部分表示将"Measurement"表和"Sensor"表进行联接（JOIN），联接的条件是这两个
FROM Measurement JOIN Sensor USING (sensor) LIMIT 3;  只返回前3行的结果
```

– Find the site name, commence date and organisation name of all stations:

```
SELECT sitename, commence, organisation
FROM Station JOIN Organisation ON orgcode = code;
```

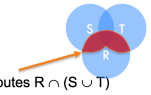
ON orgcode = code 是连接条件，表示 Station 表中的 orgcode 字段和 Organisation 表中的 code 字段相等时进行连接。

ersity of Sydney Page

An Unintuitive Query

```
SELECT DISTINCT R.A
  FROM R, S, T
 WHERE R.A = S.A OR R.A = T.A
```

这个条件表示要选择的 R.A 的值必须满足以下之一：
- R.A 的值等于 S.A 的值，
- 或者 R.A 的值等于 T.A 的值。



Computes R ∩ (S ∪ T)

But what if S = ϕ?

NULL values

The predicate IS **NULL or IS NOT NULL** can be used to check for nulls.

e.g., find measurements with an unknown intensity error value.

```
SELECT gid, band, epoch
FROM Measurement
WHERE intensity IS NULL
```

Any comparison with null returns unknown 不等于

– e.g., `5 < null or null <> null or null = null`

Result of **WHERE** clause predicate is treated as false if it evaluates to unknown.

unknown 不是 null

– e.g., `SELECT sid FROM enrolled WHERE grade = 'unknown'`

ersity of Sydney (ignores all students without a grade so far).

GROUP BY clause

Example: Find company and total amount of sales. 分组后的组可能**含有多行**，但每一个组中的所有样本都只有相同的 Group by 的条件(这里是 company)

(代码的效果)这个部分让 SQL 按照公司名称进行分组。这样的话，查询结果将会为每一家公司生成一行，显示该公司的总销售额。

```
SELECT Company, SUM(Amount)
  FROM Sales
 GROUP BY Company
```

Filtering groups with **HAVING clause**

HAVING 的作用是**对 GROUP BY 分组之后的每一组数据进行操作**

HAVING clause can further filter groups to fulfil a predicate:

```
SELECT uos_code AS unit_of_study, AVG(mark)
FROM Assessment                   取出uos_code并重命
GROUP BY uos_code                 名为unit_of_study
HAVING AVG(mark) > 10             这个条件是对上述分组结果的过滤。它只
                                  保留那些平均成绩大于10的课程单元。
```

Find the average marks of 6cp unit of studies with more than 2 results.

```
SELECT uos_code AS unit_of_study, AVG(mark)
FROM Assessment NATURAL JOIN UnitOfStudy
WHERE credit_points = 6
GROUP BY uos_code
HAVING COUNT(*) > 2
```

代码执行顺序:(不按照代码书写顺序来)

1.**FROM 和 JOIN** 先从指定的表中提取数据，并根据 JOIN 逻辑将表连接起来。2. **WHERE** 对提取的原始数据进行筛选，只保留符合条件的行。3.**GROUP BY** 对筛选后的数据按照指定的列进行分组。4.**聚合函数的计算(如 COUNT、SUM 等)**在每个分组内对数据进行聚合计算。5.**HAVING** 对分组结果进行过滤，只保留符合 HAVING 条件的分组。6.**SELECT** 从剩余的分组中提取需要的列或计算结果。7.**ORDER BY** 对最终的结果进行排序。(eg. ORDER BY title)
8.**LIMIT（如果有）**限制最终结果的输出行数。(eg. LIMIT 3)

Week6:

Observational Study

Simply observing what happens.

Records information about subjects without applying any treatments to subjects (passive participation of researcher).

Sample survey

Provide information about a population based on a sample at a specific point in time.– **Only establish correlation** not causality!

Study 1: Tanning and Skin Cancer.
• The observational study involved 1,500 people.
• Selected a group of people who had skin cancer and another group of people who did not have skin cancer.
• Asked all participants whether they used tanning beds.
• Wanted to see if there was an association between tanning beds and skin cancer prevalence.

Experimental Study

Records information about subjects while **applying treatments** to subjects and **controlling study conditions** to some degree (active participation of researcher).

Strong hypotheses, sample size for desired power and controlled data collection per specified protocols. – **Establish causality**.

Example: randomized control trials.
– 100 subjects. – Factor: Average Computer Time. – Treatments:
• Control group (computer time: max. 30 minutes). • Treatment group (computer time: 2 hours).
– 50 subjects randomly assigned to each treatment. – Response: we measure the blood pressure for each group.

Experimental vs observational

Main difference: **Most experiments use random assignment, while observational studies do not**.

Observational studies typically only establish **correlation but not causality**

Experimental studies establish **causality**

Types of variables

Dependent variable: Measure of interest.

Independent variable: Manipulated to observe the effect on dependent variable

Controlled variables: Materials, measurements and methods that don't change.

Research question (Q) – Asks whether the independent variable has an effect.

Null hypothesis (H0): The assumption that there is no effect. 零假设是一种假设，指出在研究中没有观察到显著的效应或变化。换句话说，它认为任何变化仅仅是随机波动而不是受某种因素影响的结果。

Hypothesis testing

We use hypothesis testing to specify whether to accept or reject a claim about a population depending on the evidence provided by a sample of data.

A hypothesis test examines two opposing hypotheses about a population parameter (e.g. the mean):
– **The null hypothesis and the alternative hypothesis**.
– The null hypothesis represents our **initial assumption** about the parameter, and we **collect evidence** to possibly **reject the null hypothesis** in favour of the alternative hypothesis.

Testing reliability with p-values

Compare to significance level threshold α.
– **α is the probability of (wrongly) rejecting H0 given that it is true** (**Type I error rate**, i.e., false positive). – Commonly use **α of 5% or 1%.**

values

Test results

Actual Condition	Accept H_0	Reject H_0
H_0 (H_0 is True) - No difference	Right Decision	Type I error
H_1 (H_0 is False) - Difference exists	Type II error	Right Decision

P-value	Indicates	Reject H_0?
<α	Strong evidence against the null hypothesis	Yes
>α	Weak evidence against the null hypothesis	No
=α	Marginal	NA

Not every test result is correct

α=0.05 will **erroneously reject H0 5% of the time**

–Perform enough tests and you **will get a false result** (**p-hacking**): 这是一种在数据分析中不当调整测试的方法，研究者可能通过反复试验、选择性报告结果等手段来达到显著性，从而导致得到虚假的结果。

Good science:– Determine hypotheses before looking at data – Perform hypothesis-agnostic data cleaning – Remember that p-values do not replace common sense

One-side test

单侧检验是指在假设检验中，仅考虑一个方向的偏差，通常用于验证某一参数是否大于或小于某个特定值。与其相对的则是双侧检验，后者同时考虑两个方向的偏差。

Suppose we want to test whether $\mu_x > \mu_y$

- **null hypothesis:** $\mu_x = \mu_y$
- **alternative hypothesis:** $\mu_x > \mu_y$ Interpretation of p-values
- P-value: probability of generating two sets of samples of 100 each such that the difference between their empirical means is at least $\bar{x} - \bar{y}$, under the assumption that the null hypothesis is true

t-test

Tests the null hypothesis that two population means are equal

Assumptions:– The samples are independent.– Populations are normally distributed. – Standard deviations are equal (by default).

U test

Nonparametric version of unpaired t-test. 用于比较两个独立样本的中位数差异。它主要适用于数据不满足正态分布的情况，或者当样本量较小时，通常被用作替代 t 检验。

–Test the null hypothesis that the distribution underlying sample x is the same as the distribution underlying sample y.

Assumptions: The samples are independent.

Analysis of variance (ANOVA)

Tests the null hypothesis **two or more groups** have the same population **mean**. 方差分析（ANOVA）是一种统计方法，用于比较三个或更多组数据的平均值，以确定这些组之间是否存在显著差异。

Assumptions: The samples are independent. – Populations are normally distributed. – Standard deviations are equal.

Kruskall-Wallis H-test

Nonparametric version of ANOVA.

Test the null hypothesis that the population **median** of all of the groups are **equal**. Kruskal-Wallis H 检验是一种无参数统计检验方法，通常用于比较三个或更多独立样本的中位数差异。它是对方差分析（ANOVA）的一种非参数替代方法. 通常用于样本不符合正态分布的情况

Assumptions: Samples are independent.

Accuracy, precision, recall, f1

Model prediction

Actual results	Spam (s=1)	Ham (s=0)
Spam (g=1)	*TP* (true positives)	*FN* (false negatives)
Ham (g=0)	*FP* (false positives)	*TN* (true negatives)

– **Accuracy:** percentage of correct over all instances.
  – (TP+TN) / N
– **Precision:** percentage of correct system predictions.
  – TP / (TP+FP)
– **Recall:** percentage of correct gold labels.
  – TP / (TP+FN)
– **F1:** Harmonic mean of Precision and Recall.
  – 2PR / (P+R)

Holdout & cross-validation methods
Holdout method– Splits the data randomly into two independent sets.
Cross-validation (k-fold, where k = 10 is most popular)– Randomly partition the data into k mutually exclusive subsets, each approximately equal size.
Leave-One-Out is a particular form of cross-validation: – k folds where k = N
Paired Student's t-test
Tests the null hypothesis that two population **means** are equal. 成对样本 t 检验（Paired Student's t-test）是一种用于比较两组相关样本平均值的统计方法。它常用于在同一组对象上进行两次测量，目的是判断这两次测量的结果是否存在显著差异。
Assumptions:– The samples are **paired** (e.g. before and after a treatment). – Populations are normally distributed.– Standard deviations are equal.
Paired tests for non-parametric data
Nonparametric version of paired t-test. 配对非参数检验（Paired tests for non-parametric data）是用于比较两个相关样本之间的差异的一种统计方法。与参数检验（如配对 t 检验）不同，非参数检验不要求数据服从正态分布，因此适用于数据分布较为特殊或样本量较小的情况。
Assumptions: – The samples are paired.
Experimental design step
1. Formulating a research question and null hypothesis. 2. Designing and running experiments. 3. Analysing results using appropriate statistics.
Week7:
An itemset is a collection of one or more items{Milk,Bread,Diaper}
– A k-itemset is an itemset containing k items
Frequent Itemset

Support count (σ) is the itemset frequency
$\sigma(\{Milk,Diaper,Beer\}) = 2$
T: number of total transaction

Support (s) is the normalised itemset frequency
$s = \frac{\sigma(\{Milk,Diaper,Beer\})}{|T|} = \frac{2}{5}$

A **frequent itemset** has
$s \geq min\_support$

Association Rule
An association rule is an implication of the form X->Y where X and Y are itemsets {Milk,Diaper}->{Beer}

An **association rule** has
$c \geq min\_conf$

Confidence (c) measures how often Y occurs in transactions with X
$c = \frac{\sigma(\{Milk,Diaper,Beer\})}{\sigma(\{Milk,Diaper\})} = \frac{2}{3}$

Association Rule Mining Task
Given a set of transactions T, the **goal of association rule mining** is to find all rules having: support ≥ min_sup threshold, confidence≥min_conf threshold
Two-step approach
1. **Frequent itemset generation**
–Generate all itemsets with s ≥ min_support
2. **Rule generation**
–Generate high-confidence rules from each frequent itemset –Each rule is a binary partitioning of a frequent itemset
But brute force enumeration is computationally prohibitive
Frequent itemset generation
Reducing the number of candidates
Apriori Principle: If an itemset is frequent, then all of its **subsets** must also be frequent
Conversely: If an itemset is infrequent, then its **supersets** are also infrequent

The support of an itemset never exceeds the support of its subsets
- s({Bread}) ≥ s({Bread, Beer})
- s({Milk}) ≥ s({Bread, Milk})

This is known as the anti-monotone property of support

The Apriori Algorithm—An Example

Min_sup = 2

Database TDB

Tid	Items
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

1st scan → C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

2nd scan → C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{A, B, C}
{A, B, E}
{A, C, E}
{B, C, E}

L_3

Itemset	sup
{B, C, E}	2

Step: 1.Count the frequency of each item in the dataset. Retain items whose support is greater than or equal to a predefined minimum support threshold(1-itemsets). 2.Generate Candidate Itemsets: Combine frequent 1-itemsets to form 2-itemsets. Use the Apriori property to prune candidates. 3. Repeat for Higher-Order Itemsets. Stop when no more candidate itemsets can be generated.
Rule Generation

Given a frequent itemset **L**, find all non-empty subsets **x ⊂ L** such that **x→ L – x** satisfies the minimum confidence requirement
– If {A,B,C,D} is a frequent itemset, candidate rules are:
A →BCD, B →ACD, C →ABD, D →ABC,
AB →CD, AC → BD, AD → BC, BC →AD, BD →AC, CD →AB,
ABC →D, ABD →C, ACD →B, BCD →A,

If **|L| = k**, then there are **$2^k – 2$** candidate association rules
Similar to support, the confidence of rules generated from the same itemset has an anti-monotone property
Example: X = {A,B,C,D}:
– c(ABC → D) ≥ c(AB → CD) ≥ c(A → BCD)
Confidence is anti-monotone w.r.t. number of items on the RHS of the rule

Step: 1.For each frequent itemset X, consider all possible non-empty subsets A⊂X. Generate a rule in the form: A→(X–A). 2. Calculate Confidence for Each Rule. Only retain rules where confidence is greater than or equal to a predefined minimum confidence threshold.
Performance Bottlenecks of Apriori
Generate huge candidate sets, Multiple scans of database
Benefit of FP-Growth
FP-tree 算法主要改进的是 **Apriori 算法的 Frequent Itemset Generation**。
Frequent-Pattern tree (FP-tree) structure - Avoid costly repeated database scans frequent pattern mining method (FP-growth) - Avoid candidate generation
Construct FP-tree
Step 1: Scan the transaction DB for the first time, find frequent items (single item patterns) and order them into a list L in frequency descending order.

Min_sup = 2 Scan DB for the first time to generate L

TID	Items bought
1	A, C, D
2	B, C, E
3	A, B, C, E
4	B, E

Item	Frequency
A	2
B	3
C	3
D	1
E	3

Step 2: For each transaction, order its frequent items according to the order in L; Scan DB the second time, construct FP-tree by putting each frequency ordered transaction onto it.

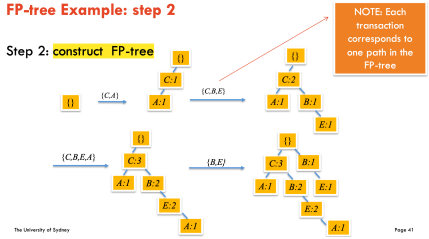
scan the DB for the second time, order frequent items in each transaction

TID	Items bought	Ordered frequent items
1	A, C, D	C, A
2	B, C, E	C, B, E
3	A, B, C, E	C, B, E, A
4	B, E	B, E

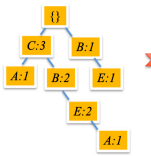
FP-tree Example: step 2

NOTE: Each transaction corresponds to one path in the FP-tree

Step 2: construct FP-tree



Mining Frequent Patterns Using FP-tree
Step 1: Construct Conditional Pattern Base
Accumulate all of **transformed prefix paths**(前缀路径) of that item to form a **conditional pattern base**



Conditional pattern bases

item	cond. pattern base
A	EBC:1, C:1
E	BC:2, B:1
B	C:2
C	{}

Step 2: Construct Conditional FP-tree

A-cond. pattern base: *A-conditional* FP-tree Frequent patterns ending with *A:*
EBC:1, C:1 → { (C:2) } | A CA:2

E-cond. pattern base: *E-conditional* FP-tree Frequent patterns ending with *E:*
BC:2, B:1 → { (B:3, C:2, BC:2) } | E *BE:3, CE:2, BCE:2*

B-cond. pattern base: *B-conditional* FP-tree Frequent patterns ending with *B:*
C:2 → { (C:2) } | B *CB:2*

All Frequent Itemsets: { A:2, E:3, B:3, C:3, BC:2, EC:2, AC:2, EB:3, BCE:2 }

Week8:
Clustering
Group data points into clusters such that: – Data points in one cluster are **more similar to one another**. – Data points **in separate clusters are less similar to one another**. – **Distance function specifies the "closeness" of two objects**.
Distances are normally used to measure the similarity or dissimilarity between two data objects

Some popular ones include: *Minkowski distance*:

$$d(i,j) = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \cdots + |x_{ip} - x_{jp}|^q}$$

where $x_i = (x_{i1}, x_{i2}, \cdots, x_{ip})$ and $x_j = (x_{j1}, x_{j2}, \cdots, x_{jp})$ are two *p*-dimensional data objects, and q is a positive integer
If q = 1, d is Manhattan distance, If q = 2, d is Euclidean distance
Dissimilarity matrix $d(i,j)$ is the dissimilarity between objects i and j
– expresses the pairwise dissimilarities (**distances**) between observations in the data set – **the desired data input to some clustering algorithm**
Types of Clusterings
Partitional clustering: A division of data objects into **non-overlapping subsets** (clusters) such that **each data object is in exactly one subset**
Hierarchical clustering: A method of cluster analysis which seeks to build a **hierarchy of clusters**. It produces **a set of nested clusters** organized as a hierarchical tree

Partitional Clustering: K-Means Clustering

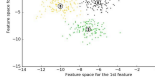
1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

Initial centroids are often chosen **randomly**. **Clusters produced vary** from one run to another. K-means will converge for common similarity measures. Most of the **convergence happens in the first few iterations**
–Complexity is *O(n × k × I × d)*
–n = number of points, k = number of clusters, i = number of iterations, d = number of attributes (or dimensions)
Hierarchical Clustering
Agglomerative (bottom-up)
The key operation is computing the **distance between two clusters** (the **proximity matrix**)
1. Compute the proximity matrix
2. Let each data point be a cluster
3. **Repeat**
4. Merge the two closest clusters
5. Update the proximity matrix
6. **Until** only a single cluster remains

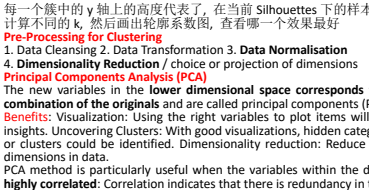
Divisive (top-down)
Minimum spanning tree: Each time, select the point with the smallest distance to the tree and add it to the tree (create a link)
Split: Each time, disconnect the edge with the longest distance (lowest similarity)
Distance Calculation between two clusters
single linkage: The minimum distance between elements of the two clusters
complete linkage: The maximum distance between elements of the two clusters
average linkage: i.e. mean distance calculation.
Measures of Cluster Validity
External Index: Measure the extent to which cluster labels match externally supplied class labels (e.g., accuracy, precision, recall, F1-score)
Homogeneity ranges from 0 to 1, preferring each cluster contains only members of a single class.
(analogous to precision, P = TP / (TP+FP))
Completeness ranges from 0 to 1, preferring all members of a given class are assigned to the same cluster
(analogous to recall, R = TP / (TP+FN))
V-measure is the harmonic mean of homogeneity and completeness
(analogous to F1 score = 2PR / (P+R))
Internal Index: Measure the goodness of a clustering structure without respect to external information (e.g., Sum of Squared Error)
For each point, the error is the distance to the nearest cluster
To get SSE, we square these errors and sum them.

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(),$$

Internal: Silhouette Coefficient
In a similar fashion **you need to calculate the silhouette coefficient for each data point in each cluster**. Then we **average them** to calculate the overall silhouette coefficient to evaluate the resultant clusters.
The closer to 1 the better
For an individual point *x*
– Calculate ***a*** = average distance of *x* to points in its cluster
– Calculate ***b*** = average distance of *x* to points in the next nearest cluster
– The silhouette coefficient for *a* point is then given by
s = 1 – a/b if a < b, (or s = b/a - 1 if a ≥ b, not the usual case)
– The closer to 1 the better



Relative Index: Compare two different clusterings or clusters (often an external or internal index is used)
Choosing k
Using Silhouettes to choose k
High average silhouette indicates points far away from neighbouring clusters



每一个簇中的 y 轴上的高度代表了，在当前 Silhouettes 下的样本个数
计算不同的 k，然后画出轮廓系数图，查看哪一个效果最好
Pre-Processing for Clustering
1. Data Cleansing 2. Data Transformation 3. **Data Normalisation**
4. **Dimensionality Reduction** / choice or projection of dimensions
Principal Components Analysis (PCA)
The new variables in the **lower dimensional space corresponds to a linear combination of the originals** and are called principal components (PC)
Benefits: Visualization: Using the right variables to plot items will give more insights. Uncovering Clusters: With good visualizations, hidden categories or clusters could be identified. Dimensionality reduction: Reduce number of dimensions in data.
PCA method is particularly useful when the variables within the data set are **highly correlated**: Correlation indicates that there is redundancy in the data
Step: 1.Standardize each feature to have a mean of 0 and a standard deviation of 1. 2. For a dataset with n features, compute the covariance matrix. 3. Solve the eigenvalue equation. Cv=λv. Where C is the covariance matrix, v are the eigenvectors, and λ are the eigenvalues. 4. Sort eigenvalues in descending order. Arrange the corresponding eigenvectors accordingly. 5. Choose the top k eigenvectors based on the cumulative explained variance threshold (e.g., 95%). 6. To project the original data into the new space defined by the selected principal components. Z'=ZV_k. Multiply the standardized data matrix Z by the matrix of selected eigenvectors V_k
Covariance Matrix
In the covariance table above, **the off-diagonal values are different from zero. This indicates the presence of redundancy in the data.**
PCA creates **uncorrelated PC variables (called eigenvectors)** having zero covariations and **variances (called eigenvalues)** sorted in decreasing order.
The first PC captures the greatest variance, the second greatest variance is the second PC, and so on.
Week9:
$Y_i = \alpha + \beta X_i + \varepsilon_i$
Simple linear regression
Method for **finding the line** of best fit between the dependent variable Y and the independent variable X
– α: Intercept (where the line crosses the y axis)– β: Slope (direction and steepness of the line) – εi: Error **(error term describing variation of data)**– Yi is the dependent variable (response). – Xi is the independent variable (predictor).
Error/residual: Difference between the observed value and predicted value
– **Sum of squared errors:**

$$SST = \sum_i^n (y_i - \bar{y})^2 = (n-1)\,\mathrm{var}(y) \qquad \varepsilon = SSE = \sum (y_i - \hat{y}_i)^2$$
$$SSR = \sum_i^n (\hat{y}_i - \bar{y})^2$$

Sum Error Square

Ordinary Least Squares (OLS)
Our goal is to find the optimal value of $\hat{\alpha}$ and $\hat{\beta}$ such that
$\frac{1}{n}\sum_{i=1}^n \left((\hat{\alpha} + \hat{\beta}x_i) - y_i\right)^2$ is the minimum
Using calculus

$$\hat{\beta} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = \frac{\mathrm{cov}(x,y)}{\mathrm{Var}(x)} = \frac{r(x,y) * sd(y)}{sd(x)}$$
$$\hat{\alpha} = \bar{y} - \hat{\beta}\bar{x}$$

$\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ is the average of x_i values
$\bar{y} = \frac{1}{n}\sum_{i=1}^n y_i$ is the average of y_i values
sd is the standard deviation and r is the correlation

Coefficient of determination (R^2)
R^2: **ratio of explained variation in y to total variation in y**
Ranges from 0 to 1, with **higher values indicating better** fit for the linear model.
R^2 value **may be negative** that indicate overfit.

$$R^2 = \frac{\sum(\hat{y}_i - \bar{y})^2}{\sum(y_i - \bar{y})^2} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} = 1 - \frac{SSE}{SST}$$

Multiple linear regression
two or more independent variables and – one dependent variable
There is also a **closed-form solution**

$$Y = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j = \theta^T x$$

Assume $x_0 = 1$

Polynomial transformation
This allows the use of linear regression techniques to **fit non-linear datasets**.

$$Y = h_\theta(x) = \theta_0 + \theta_1 x^1 + \theta_2 x^2 + \ldots + \theta_d x^d = \sum_{j=0}^d \theta_j x^j$$

Overfitting
The learned model may fit the training set very well but fails to generalise to new examples
Regularization aims to penalize for large values of coefficients (θj)
Can also address overfitting by eliminating features
Linear regression cost function

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^n (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^d \theta_j^2$$

Model fit to data regularization

No regularization on *θ0*. There is also a closed-form solution for minimizing $J(\theta)$
Gradient Descent
For many other models/cost functions, there is **no closed-form solution of *θ***
Initialize θ
Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha * \frac{\partial}{\partial \theta_j} J(\theta)$$

α is a learning rate (taking a small value) e.g. $\alpha = 0.05$

$$\theta_j \leftarrow \theta_j - \alpha * \frac{1}{n}\sum_{i=1}^n (h_\theta(x^{(i)}) - y^{(i)})\, x_j^{(i)}$$

Gradient descent **always converges to global minimum (assuming *α* is small)**
Selecting learning rate
If *α* is small, gradient descent can be slow. If *α* is too large, gradient descent might overshoot the minimum. **One good strategy is gradually shrinking the learning rate**
Batch and stochastic gradient descents
Batch: Accurate but slow: has to scan through the **entire training set** before taking a single step. costly operation if n is large
Stochastic: Fast, start making progress right away. It may not converge to the minimum
A trade-off between these two extremes is **mini-batch**
Logistic regression
Predict probability of categorical label
Logistic or sigmoid function

$$\ln\frac{p}{1-p} = b_0 + b_1 x \qquad 0 \leq h_\theta(x) \leq 1 \qquad h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

Cost function and optimization
Linear regression cost function was convex. The **same cost function for logistic regression is nonconvex** because of **nonlinear sigmoid** function
Instead of Mean Squared Error, we use a cost function called **Cross-Entropy**, also known as Log Loss.

$$J(\theta) = -\frac{1}{2n}\sum_{i=1}^n \left[y^{(i)} \log\left(h_\theta(x^{(i)})\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta(x^{(i)})\right)\right]$$

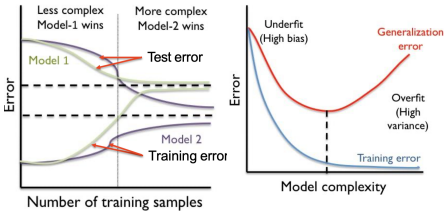
Multi-class classification
One-vs-rest strategy: We train one logistic regression classifier for each class *i* to predict the probability that $y = i$. For each x, pick the class having highest value of probability.
Or use the **softmax function**
Regularization parameters in scikit-learn
Penalty: **l1 (lasso)**: estimates sparse coefficients; equivalent to feature selection.
l2 (ridge): minimizes coefficients; pulls coefficients toward 0
C: Inverse of regularization strength. Small values specify stronger regularization
Grid search
Parameters like penalty and regularization strength are not learned from data by default. Can be set using exhaustive search through combinations of specified possible values. Perform n-fold cross validation for each combination
Week10:
Decision tree
Basic algorithm (a greedy ID3 algorithm).
– Tree is constructed in a **top-down recursive divide-and-conquer manner**.
– At start, all the training examples are at the root.
– **Examples are partitioned recursively** based on selected attributes.
– Test attributes are selected on the basis of a heuristic or statistical measure (e.g., **Information Gain** (IG)).
Conditions for stopping partitioning
– **All samples for a given node belong to the same class**.
– There are **no remaining attributes for further partitioning – majority voting** is employed for classifying the leaf.
Information Gain (IG)
IG calculates **effective change in entropy** after making a decision based on the value of an attribute.
Entropy $IG(Y|X) = H(Y) - H(Y|X)$
To measure the uncertainty associated with data:
Interpretation: Higher entropy => higher uncertainty. Lower entropy => lower uncertainty. H(Y | X): **the average conditional entropy of Y.**

$$H(Y) = -\sum_{i=1}^m p_i \log_2(p_i \qquad H(Y|X) = \sum_i p(X = v_i) * H(Y|X = v_i)$$

Gain = T1 –T2; **The best attribute is the one with highest information gain**
it will consider the entropies of S1, S2 and S3 and the **proportion of examples** following **each branch (5/14, 4/14, 5/15)**

$$T2 = H(S\,|\,outlook) = \frac{5}{14}\cdot H(S_1) + \frac{4}{14}\cdot H(S_2) + \frac{5}{14}\cdot H(S_3)$$

Model choice depends on amount of data available
当训练数据量小于某个阈值时，较简单的模型 1 获胜。
如果能够获得更多的数据，较复杂的模型 2 将最终获胜。
如果现有的数据量已经很多，增加数据量对于两种模型的性能提升都不大。
The dashed line on right shows point where we switch from **under-fitting to overfitting**. – Goal: **Find this dotted line.**



Data drift (non-stationary data)
What is it: Should be near-true for train and test samples. Only near-true in production for a little while. 随着时间的推移，生产环境中的数据可能会发生变化，不再符合最初训练模型时的数据分布。
What to do: Monitor offline metric on live data. May require monitoring/annotation. If there are large changes, then retrain on new data. Online/incremental learning.
How to train a model
Build a simple model first, evaluate, iterate
Start by building an end-to-end pipeline and evaluation. Replicate published benchmarks to sanity check pipeline.
Wash, rinse, repeat:– Review the data and problem.– Hypothesize next best approach in terms of elegance and impact. – Implement and evaluate approach.
Benefits of Feature engineering
Relates back to understanding the problem. Design informative and discriminative features. Understand and validate features to avoid overfitting.
Ensembles of predictors
Random forest: Lose explainability of trees! Vote across many classifiers. Generally boosts the performance of the final model.
Week11:
Unstructured data refers to information that does not have a pre-defined data model. Unstructured data is typically text-heavy which make it more difficult to understand. E.g.: Images, Video, Email, Social media
Modelling spam detection
Tokenisation: Split a string (document) into pieces called **tokens**.
Normalisation: Map similar words to the same token
Indicator features: **Binary indicator feature** for each word in a document
Term frequency weighting:
Term frequency. Give **more weight** to terms that are **common in document**
TF = |occurrences of term in doc|
Damping: Sometimes want to **reduce impact of high counts**
TF = log(|occurrences of term in doc|)
TFIDF Weighting: Inverse document frequency. Give **less weight** to terms that are **common across documents**
IDF = log(|docs|/|docs containing term|). TFIDF = TF * IDF
Characteristics of the text: 1. Emails are usually represented using TF-IDF vectorization, resulting in high-dimensional feature spaces. 2. Many words in an email are zero (absent), making the data matrix sparse. 3. The task is to classify emails into one of two classes. 4. Features are textual.
Naïve Bayes
Bayes theorem:

$$P(H\,|\,E) = \frac{P(E\,|\,H)P(H)}{P(E)}$$

Two assumptions: **independence** and **equally importance for all attributes**
How to use: 1. Calculate P(H|E) for each H (class), i.e. P(yes|E) and P(no|E) 2. Compare them and assign E to the class with the highest probability

$$P(yes\,|\,E) = \frac{P(E_1\,|\,yes)P(E_2\,|\,yes)P(E_3\,|\,yes)P(E_4\,|\,yes)P(yes)}{P(E)}$$

We don't need to calculate P(E), **just compare the numerator**
add 1 to the numerator and **m to the denominator** (m - number of attribute values = 3 for outlook) that is called **Laplace correction or smoothing**. It ensures that the probabilities will never be 0
During **training**, this algorithm **doesn't include the missing values** and just based on training samples without missing values for each attribute
When apply Naïve Bayers to **numeric attribute**, using probability density function for a **normal distribution** to calculate probability.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

For spam detection: Naive Bayes handles textual, high-dimensional, sparse data efficiently. It computes probabilities using word frequencies without the need for complex transformations. It has low computational requirements, making it suitable for large email datasets. It assigns probabilities to predictions.