

Fundamentals of Machine Learning for Predictive Data Analytics

Machine Learning for Predictive Data Analytics

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

- 1 What is Predictive Data Analytics?
- 2 What is Machine Learning?
- 3 How Does Machine Learning Work?
- 4 What Can Go Wrong With ML?
- 5 The Predictive Data Analytics Project Lifecycle:
Crisp-DM
- 6 Summary

What is Predictive Data Analytics?

- Predictive Data Analytics encompasses the business and data processes and computational models that enable a business to make **data-driven decisions**.

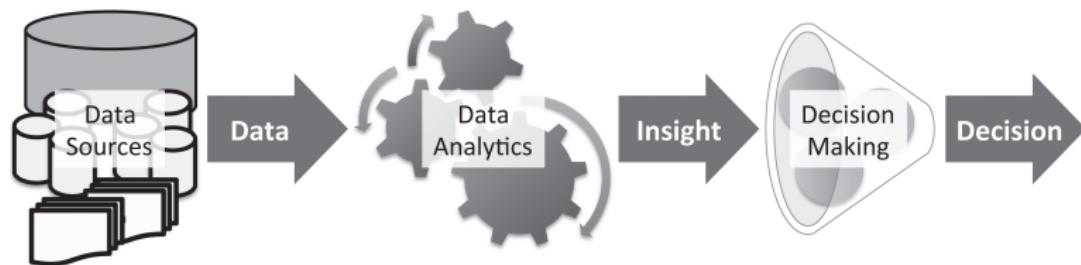


Figure: Predictive data analytics moving from **data** to **insights** to **decisions**.

Example Applications:

- Price Prediction
- Fraud Detection
- Dosage Prediction
- Risk Assessment
- Propensity modelling
- Diagnosis
- Document Classification
- ...

What is Machine Learning?

- (Supervised) Machine Learning techniques automatically learn a model of the relationship between a set of **descriptive features** and a **target feature** from a set of historical examples.

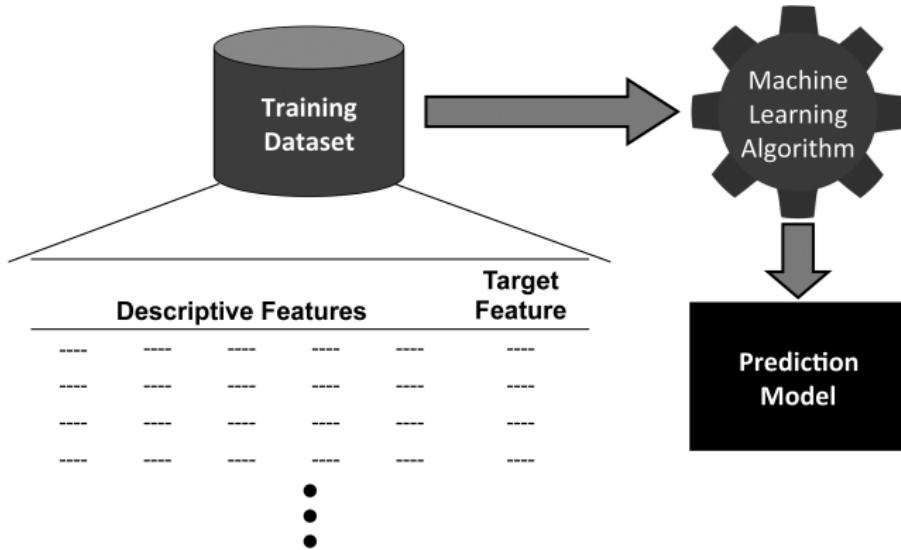


Figure: Using machine learning to induce a prediction model from a training dataset.



Figure: Using the model to make predictions for new query instances.

ID	OCCUPATION	AGE	LOAN-SALARY RATIO	OUTCOME
1	industrial	34	2.96	repaid
2	professional	41	4.64	default
3	professional	36	3.22	default
4	professional	41	3.11	default
5	industrial	48	3.80	default
6	industrial	61	2.52	repaid
7	professional	37	1.50	repaid
8	professional	40	1.93	repaid
9	industrial	33	5.25	default
10	industrial	32	4.15	default

- What is the relationship between the **descriptive features** (OCCUPATION, AGE, LOAN-SALARY RATIO) and the **target feature** (OUTCOME)?

```
if LOAN-SALARY RATIO > 3 then
    OUTCOME='default'
else
    OUTCOME='repay'
end if
```

```
if LOAN-SALARY RATIO > 3 then
    OUTCOME='default'
else
    OUTCOME='repay'
end if
```

- This is an example of a **prediction model**

```
if LOAN-SALARY RATIO > 3 then
    OUTCOME='default'
else
    OUTCOME='repay'
end if
```

- This is an example of a **prediction model**
- This is also an example of a **consistent** prediction model

```
if LOAN-SALARY RATIO > 3 then
    OUTCOME='default'
else
    OUTCOME='repay'
end if
```

- This is an example of a **prediction model**
- This is also an example of a **consistent** prediction model
- Notice that this model does not use all the features and the feature that it uses is a derived feature (in this case a ratio): **feature design** and **feature selection** are two important topics that we will return to again and again.

- What is the relationship between the **descriptive features** and the **target feature** (OUTCOME) in the following dataset?

ID	Amount	Salary	Loan-Salary	Age	Occupation	House	Type	Outcome
			Ratio					
1	245,100	66,400	3.69	44	industrial	farm	stb	repaid
2	90,600	75,300	1.2	41	industrial	farm	stb	repaid
3	195,600	52,100	3.75	37	industrial	farm	ftb	default
4	157,800	67,600	2.33	44	industrial	apartment	ftb	repaid
5	150,800	35,800	4.21	39	professional	apartment	stb	default
6	133,000	45,300	2.94	29	industrial	farm	ftb	default
7	193,100	73,200	2.64	38	professional	house	ftb	repaid
8	215,000	77,600	2.77	17	professional	farm	ftb	repaid
9	83,000	62,500	1.33	30	professional	house	ftb	repaid
10	186,100	49,200	3.78	30	industrial	house	ftb	default
11	161,500	53,300	3.03	28	professional	apartment	stb	repaid
12	157,400	63,900	2.46	30	professional	farm	stb	repaid
13	210,000	54,200	3.87	43	professional	apartment	ftb	repaid
14	209,700	53,000	3.96	39	industrial	farm	ftb	default
15	143,200	65,300	2.19	32	industrial	apartment	ftb	default
16	203,000	64,400	3.15	44	industrial	farm	ftb	repaid
17	247,800	63,800	3.88	46	industrial	house	stb	repaid
18	162,700	77,400	2.1	37	professional	house	ftb	repaid
19	213,300	61,100	3.49	21	industrial	apartment	ftb	default
20	284,100	32,300	8.8	51	industrial	farm	ftb	default
21	154,000	48,900	3.15	49	professional	house	stb	repaid
22	112,800	79,700	1.42	41	professional	house	ftb	repaid
23	252,000	59,700	4.22	27	professional	house	stb	default
24	175,200	39,900	4.39	37	professional	apartment	stb	default
25	149,700	58,600	2.55	35	industrial	farm	stb	default

```
if LOAN-SALARY RATIO < 1.5 then
    OUTCOME='repay'
else if LOAN-SALARY RATIO > 4 then
    OUTCOME='default'
else if AGE < 40 and OCCUPATION = 'industrial' then
    OUTCOME='default'
else
    OUTCOME='repay'
end if
```

```
if LOAN-SALARY RATIO < 1.5 then
    OUTCOME='repay'
else if LOAN-SALARY RATIO > 4 then
    OUTCOME='default'
else if AGE < 40 and OCCUPATION = 'industrial' then
    OUTCOME='default'
else
    OUTCOME='repay'
end if
```

- The real value of machine learning becomes apparent in situations like this when we want to build prediction models from large datasets with multiple features.

How Does Machine Learning Work?

- Machine learning algorithms work by searching through a set of possible prediction models for the model that best captures the relationship between the descriptive features and the target feature.

- Machine learning algorithms work by searching through a set of possible prediction models for the model that best captures the relationship between the descriptive features and the target feature.
- An obvious search criteria to drive this search is to look for models that are **consistent** with the data.

- Machine learning algorithms work by searching through a set of possible prediction models for the model that best captures the relationship between the descriptive features and the target feature.
- An obvious search criteria to drive this search is to look for models that are **consistent** with the data.
- However, because a training dataset is only a sample ML is an **ill-posed** problem.

Table: A simple retail dataset

ID	B BY	ALC	ORG	GRP
1	no	no	no	couple
2	yes	no	yes	family
3	yes	yes	no	family
4	no	no	yes	couple
5	no	yes	yes	single

Table: A full set of potential prediction models before any training data becomes available.

BBY	ALC	ORG	GRP	M ₁	M ₂	M ₃	M ₄	M ₅	...	M _{6 561}
no	no	no	?	couple	couple	single	couple	couple		couple
no	no	yes	?	single	couple	single	couple	couple		single
no	yes	no	?	family	family	single	single	single		family
no	yes	yes	?	single	single	single	single	single	...	couple
yes	no	no	?	couple	couple	family	family	family	...	family
yes	no	yes	?	couple	family	family	family	family		couple
yes	yes	no	?	single	family	family	family	family		single
yes	yes	yes	?	single	single	family	family	couple		family

Table: A sample of the models that are consistent with the training data

BBY	ALC	ORG	GRP	M ₁	M ₂	M ₃	M ₄	M ₅	...	M _{6 561}
no	no	no	couple	couple	couple	single	couple	couple		couple
no	no	yes	couple	single	couple	single	couple	couple		single
no	yes	no	?	family	family	single	single	single		family
no	yes	yes	single	single	single	single	single	single		couple
yes	no	no	?	couple	couple	family	family	family	...	family
yes	no	yes	family	couple	family	family	family	family		couple
yes	yes	no	family	single	family	family	family	family		single
yes	yes	yes	?	single	single	family	family	couple		family

Table: A sample of the models that are consistent with the training data

B _{BY}	A _{LC}	O _{RG}	G _{RP}	M ₁	M ₂	M ₃	M ₄	M ₅	...	M ₆ 561
no	no	no	couple	couple	couple	single	couple	couple		couple
no	no	yes	couple	single	couple	single	couple	couple		single
no	yes	no	?	family	family	single	single	single		family
no	yes	yes	single	single	single	single	single	single		couple
yes	no	no	?	couple	couple	family	family	family	...	family
yes	no	yes	family	couple	family	family	family	family		couple
yes	yes	no	family	single	family	family	family	family		single
yes	yes	yes	?	single	single	family	family	couple		family

- Notice that there is more than one candidate model left! It is because a single consistent model cannot be found based on a **sample** training dataset that ML is **ill-posed**.

- Consistency \approx **memorizing** the dataset.
- Consistency with **noise** in the data isn't desirable.
- Goal: a model that **generalises** beyond the dataset and that isn't influenced by the noise in the dataset.
- So what criteria should we use for choosing between models?

- **Inductive bias** the set of assumptions that define the model selection criteria of an ML algorithm.
- There are two types of bias that we can use:
 - 1 restriction bias
 - 2 preference bias
- Inductive bias is necessary for learning (beyond the dataset).

How ML works (Summary)

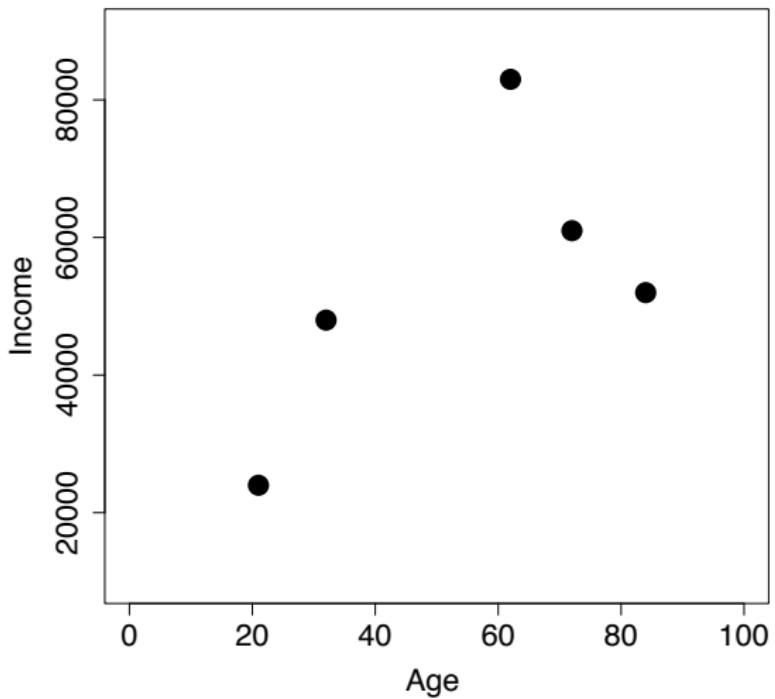
- ML algorithms work by searching through sets of potential models.
- There are two sources of information that guide this search:
 - ➊ the training data,
 - ➋ the inductive bias of the algorithm.

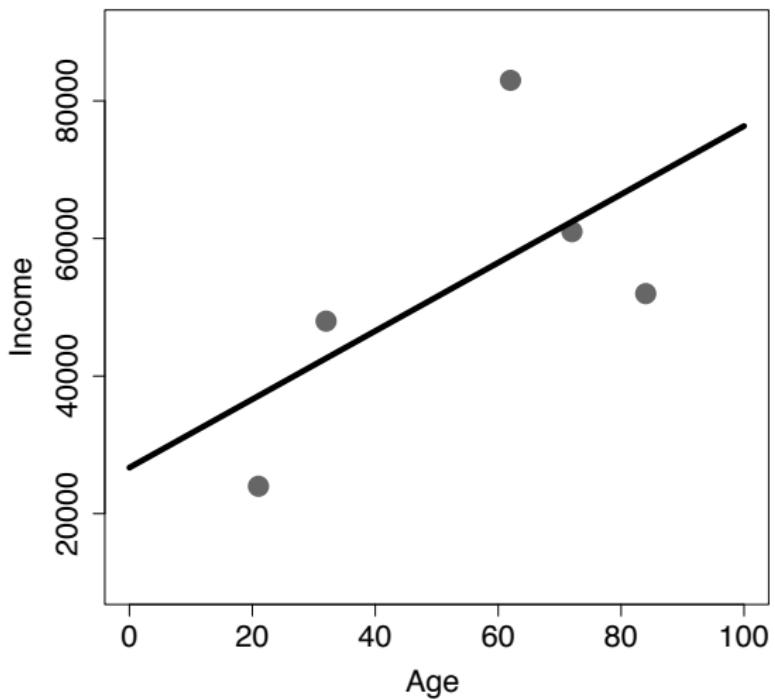
What Can Go Wrong With ML?

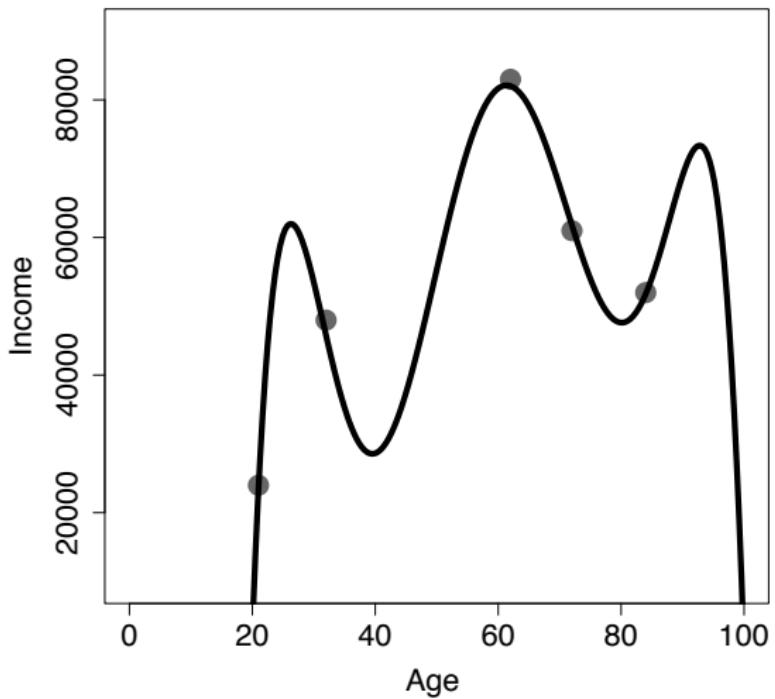
- No free lunch!
- What happens if we choose the wrong inductive bias:
 - ① underfitting
 - ② overfitting

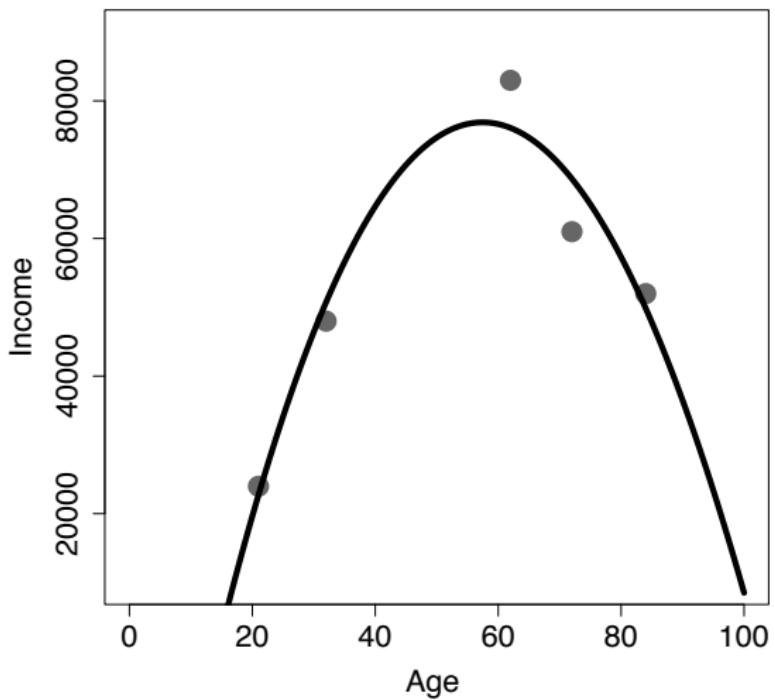
Table: The age-income dataset.

ID	AGE	INCOME
1	21	24,000
2	32	48,000
3	62	83,000
4	72	61,000
5	84	52,000









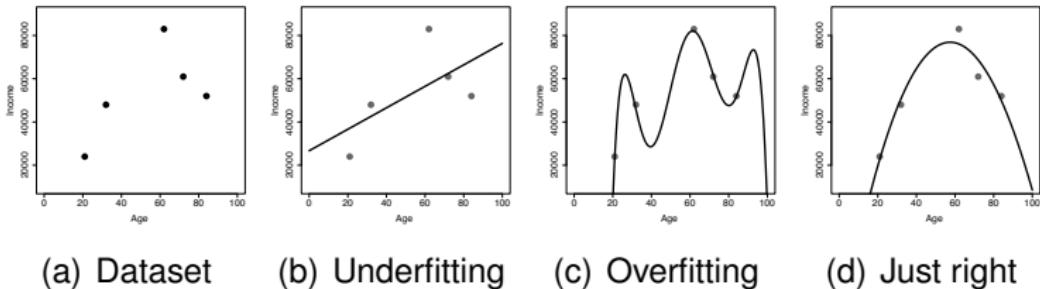


Figure: Striking a balance between overfitting and underfitting when trying to predict age from income.

- There are many different types of machine learning algorithms.
- In this course we will cover four families of machine learning algorithms:
 - 1 **Information based learning**
 - 2 **Similarity based learning**
 - 3 **Probability based learning**
 - 4 **Error based learning**

The Predictive Data Analytics Project Lifecycle: Crisp-DM

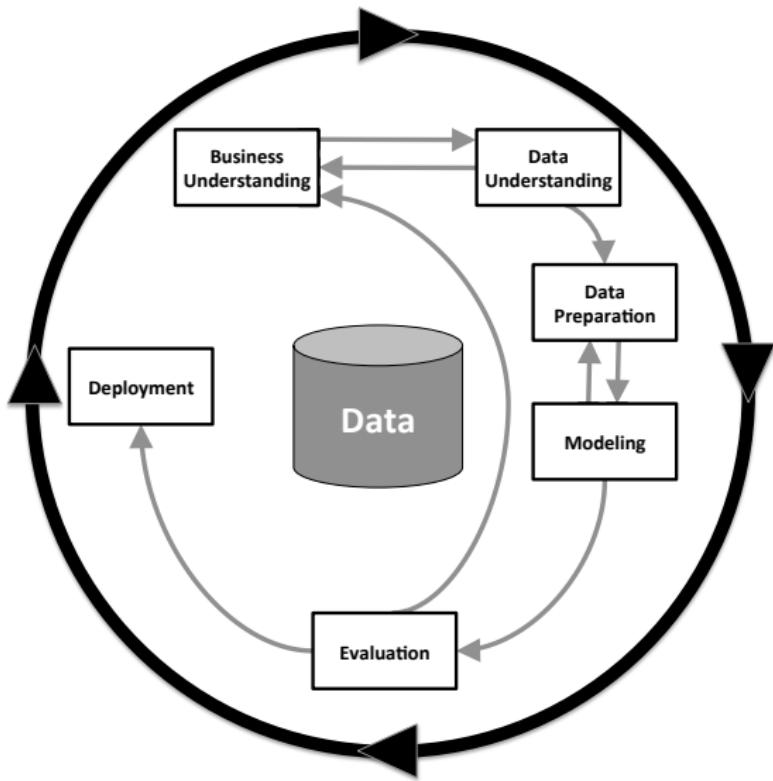


Figure: A diagram of the CRISP-DM process which shows the six key phases and indicates the important relationships between them. This figure is based on Figure 2 of [1].

Summary

- Machine Learning techniques automatically learn the relationship between a set of **descriptive features** and a **target feature** from a set of historical examples.
- Machine Learning is an **ill-posed** problem:
 - 1 **generalize**,
 - 2 **inductive bias**,
 - 3 **underfitting**,
 - 4 **overfitting**.
- Striking the right balance between model complexity and simplicity (between underfitting and overfitting) is the hardest part of machine learning.

[1] R. Wirth and J. Hipp.

Crisp-dm: Towards a standard process model for data mining.

In *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*, pages 29–39. Citeseer, 2000.

- 1 What is Predictive Data Analytics?
- 2 What is Machine Learning?
- 3 How Does Machine Learning Work?
- 4 What Can Go Wrong With ML?
- 5 The Predictive Data Analytics Project Lifecycle:
Crisp-DM
- 6 Summary

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 2: Data to Insights to Decisions

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

- 1 **Converting Business Problems into Analytics Solutions**
 - Case Study: Motor Insurance Fraud
- 2 **Assessing Feasibility**
 - Case Study: Motor Insurance Fraud
- 3 **Designing the Analytics Base Table**
 - Case Study: Motor Insurance Fraud
- 4 **Designing & Implementing Features**
 - Different Types of Data
 - Different Types of Features
 - Handling Time
 - Legal Issues
 - Implementing Features
 - Case Study: Motor Insurance Fraud
- 5 **Summary**

Converting Business Problems into Analytics Solutions



- Converting a business problem into an analytics solution involves answering the following key questions:
 - ➊ What is the business problem?
 - ➋ What are the goals that the business wants to achieve?
 - ➌ How does the business currently work?
 - ➍ In what ways could a predictive analytics model help to address the business problem?

Case Study: Motor Insurance Fraud

In spite of having a fraud investigation team that investigates up to 30% of all claims made, a motor insurance company is still losing too much money due to fraudulent claims.

- What predictive analytics solutions could be proposed to help address this business problem?

Case Study: Motor Insurance Fraud



Assessing Feasibility

- Evaluating the feasibility of a proposed analytics solution involves considering the following questions:
 - ① Is the data required by the solution available, or could it be made available?
 - ② What is the capacity of the business to utilize the insights that the analytics solution will provide?

- What are the data and capacity requirements for the proposed Claim Prediction analytics solution for the motor insurance fraud scenario?

- What are the data and capacity requirements for the proposed Claim Prediction analytics solution for the motor insurance fraud scenario?

Case Study: Motor Insurance Fraud

[Claim prediction]

Data Requirements: A large collection of historical claims marked as '*fraudulent*' and '*non-fraudulent*'. Also, the details of each claim, the related policy, and the related claimant would need to be available.

Capacity Requirements: The main requirement is that a mechanism could be put in place to inform claims investigators that some claims were prioritized above others. This would also require that information about claims become available in a suitably timely manner so that the claims investigation process would not be delayed by the model.



Designing the Analytics Base Table

- The basic structure in which we capture historical datasets is the **analytics base table (ABT)**

Descriptive Features					Target Feature
---	---	---	---	---	---
---	---	---	---	---	---
---	---	---	---	---	---
---	---	---	---	---	---
---	---	---	---	---	---

Figure: The general structure of an **analytics base table**—descriptive features and a target feature.

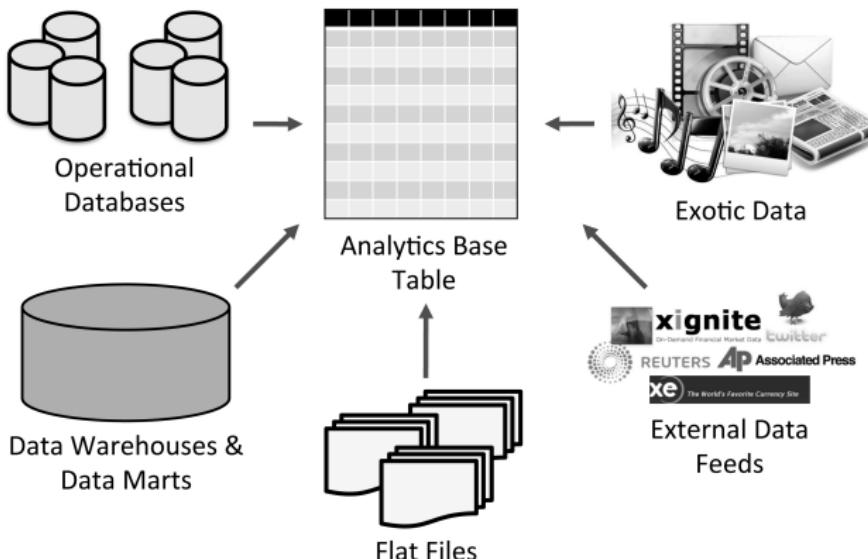


Figure: The different data sources typically combined to create an analytics base table.

- The **prediction subject** defines the basic level at which predictions are made, and each row in the ABT will represent one instance of the prediction subject—the phrase **one-row-per-subject** is often used to describe this structure.
- Each row in an ABT is composed of a set of descriptive features and a target feature.
- Defining features can be difficult!

- A good way to define features is to identify the key **domain concepts** and then to base the features on these concepts.

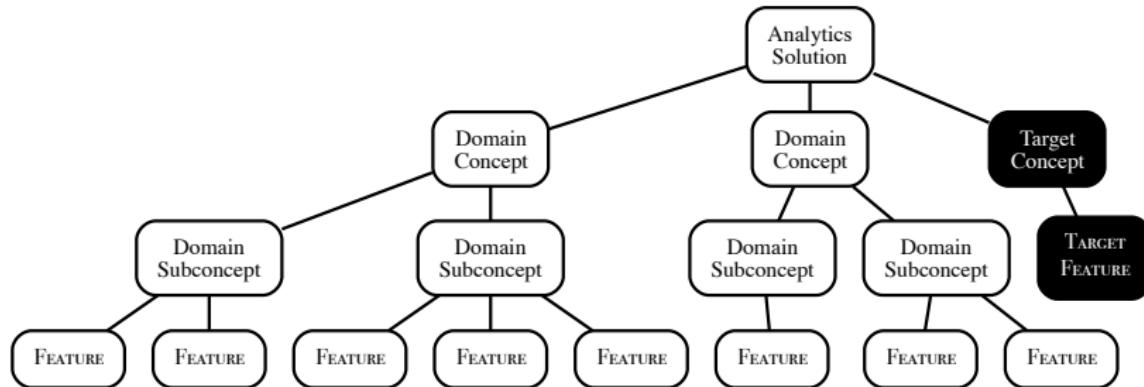


Figure: The hierarchical relationship between an analytics solution, domain concepts, and descriptive features.

- There are a number of general domain concepts that are often useful:
 - Prediction Subject Details
 - Demographics
 - Usage
 - Changes in Usage
 - Special Usage
 - Lifecycle Phase
 - Network Links

Case Study: Motor Insurance Fraud

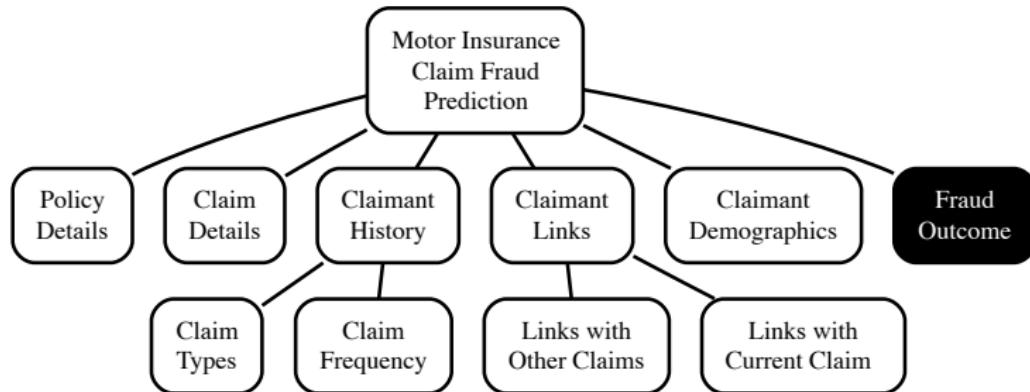


Figure: Example domain concepts for a motor insurance fraud claim prediction analytics solution.

Designing & Implementing Features

- Three key data considerations are particularly important when we are designing features.
 - **Data availability**
 - **Timing**
 - **Longevity**

Different Types of Data

				Ordinal			Categorical
ID	NAME	DATE OF BIRTH	GENDER	CREDIT RATING	COUNTRY	salary	
0034	Brian	22/05/78	male	aa	ireland	67,000	
0175	Mary	04/06/45	female	c	france	65,000	
0456	Sinead	29/02/82	female	b	ireland	112,000	
0687	Paul	11/11/67	male	a	usa	34,000	
0982	Donald	01/12/75	male	b	australia	88,000	
1103	Agnes	17/09/76	female	aa	sweden	154,000	

Textual Interval Binary Numeric

Figure: Sample descriptive feature data illustrating numeric, binary, ordinal, interval, categorical, and textual types.

Different Types of Features

- The features in an ABT can be of two types:
 - **raw features**
 - **derived features**
- There are a number of common derived feature types:
 - **Aggregates**
 - **Flags**
 - **Ratios**
 - **Mappings**

Handling Time

- Many of the predictive models that we build are **propensity models**, which inherently have a temporal element
- For **propensity modeling**, there are two key periods:
 - the **observation period**
 - the **outcome period**

- In some cases the observation and outcome period are measured over the same time for all predictive subjects.

(a) Observation period and outcome period

(b) Observation and outcome periods for multiple customers (each line represents a customer)

Figure: Modeling points in time.

Handling Time

- Often the observation period and outcome period will be measured over different dates for each prediction subject.

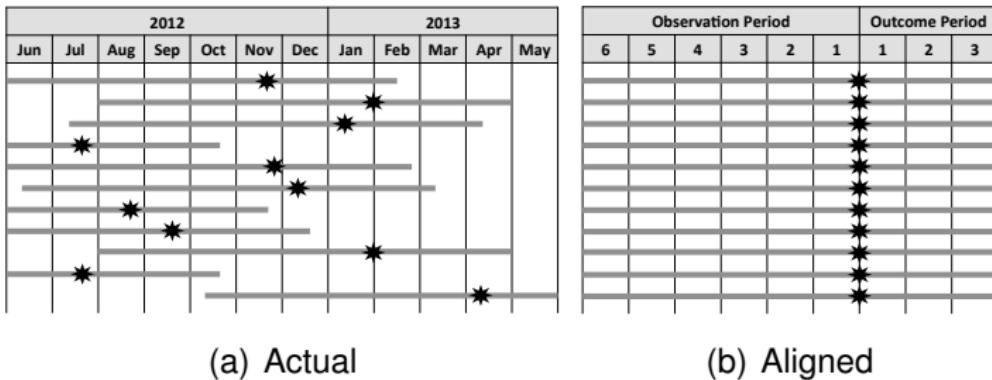
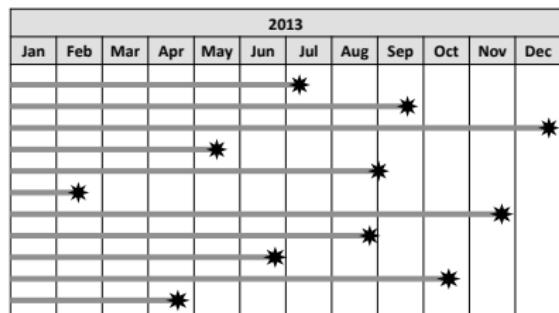


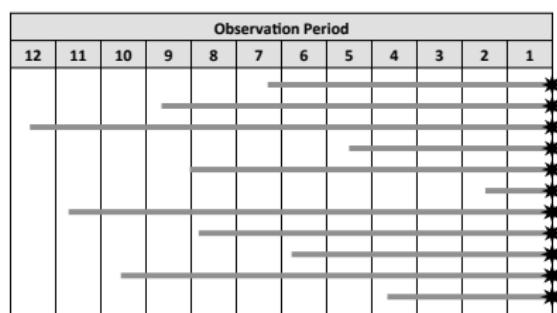
Figure: Observation and outcome periods defined by an event rather than by a fixed point in time (each line represents a prediction subject and stars signify events).

Handling Time

- In some cases only the descriptive features have a time component to them, and the target feature is time independent.



(a) Actual



(b) Aligned

Figure: Modeling points in time for a scenario with no real outcome period (each line represents a customer, and stars signify events).

Handling Time

- Conversely, the target feature may have a time component and the descriptive features may not.

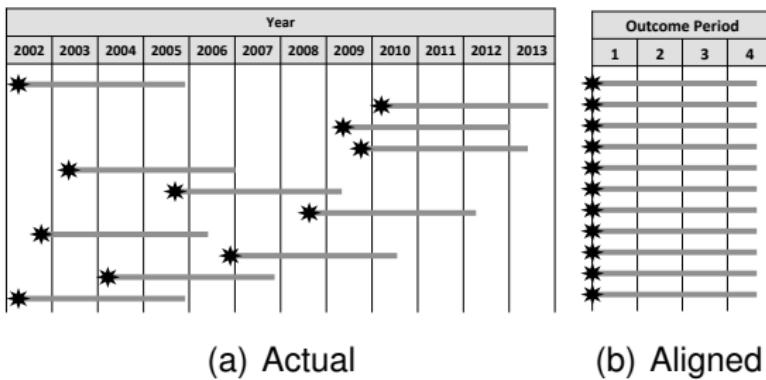


Figure: Modeling points in time for a scenario with no real observation period (each line represents a customer, and stars signify events).

Legal Issues

- Data analytics practitioners can often be frustrated by legislation that stops them from including features that appear to be particularly well suited to an analytics solution in an ABT.
- There are significant differences in legislation in different jurisdictions, but a couple of key relevant principles almost always apply.
 - 1 **Anti-discrimination legislation**
 - 2 **Data protection legislation**

Legal Issues

- Although, data protection legislation changes significantly across different jurisdictions, there are some common tenets on which there is broad agreement which affect the design of ABTs
 - The **collection limitation principle**
 - The **purpose specification principle**
 - The **use limitation principle**

Implementing Features

- Implementing a **derived feature**, however, requires data from multiple sources to be combined into a set of single feature values.
- A few key **data manipulation** operations are frequently used to calculate derived feature values:
 - joining data sources
 - filtering rows in a data source
 - filtering fields in a data source
 - deriving new features by combining or transforming existing features
 - aggregating data sources

Case Study: Motor Insurance Fraud

- What are the observation period and outcome period for the motor insurance claim prediction scenario?

Case Study: Motor Insurance Fraud

- What are the observation period and outcome period for the motor insurance claim prediction scenario?
- The observation period and outcome period are measured over different dates for each insurance claim, defined relative to the specific date of that claim.

Case Study: Motor Insurance Fraud

- What are the observation period and outcome period for the motor insurance claim prediction scenario?
- The observation period and outcome period are measured over different dates for each insurance claim, defined relative to the specific date of that claim.
- The observation period is the time prior to the claim event, over which the descriptive features capturing the claimant's behavior are calculated

Case Study: Motor Insurance Fraud

- What are the observation period and outcome period for the motor insurance claim prediction scenario?
- The observation period and outcome period are measured over different dates for each insurance claim, defined relative to the specific date of that claim.
- The observation period is the time prior to the claim event, over which the descriptive features capturing the claimant's behavior are calculated
- The outcome period is the time immediately after the claim event, during which it will emerge whether the claim is fraudulent or genuine.

Case Study: Motor Insurance Fraud

What features could you use to capture the Claim Frequency domain concept?

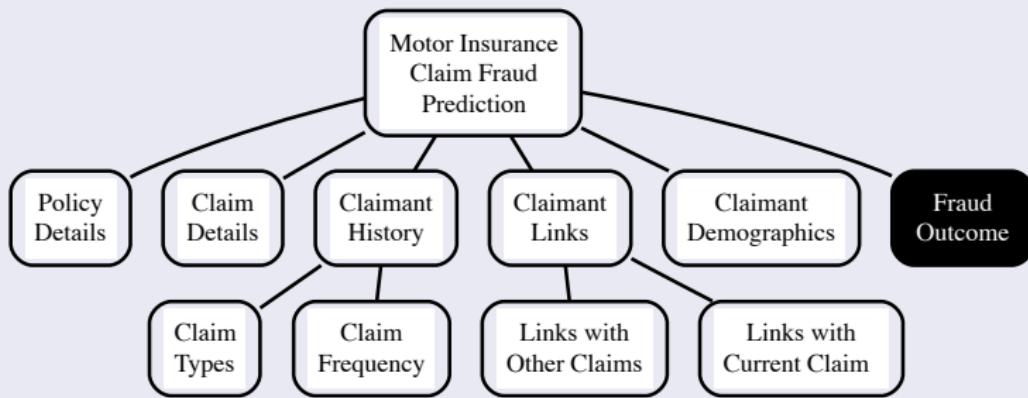


Figure: Example domain concepts for a motor insurance fraud prediction analytics solution.

Case Study: Motor Insurance Fraud

What features could you use to capture the Claim Frequency domain concept?

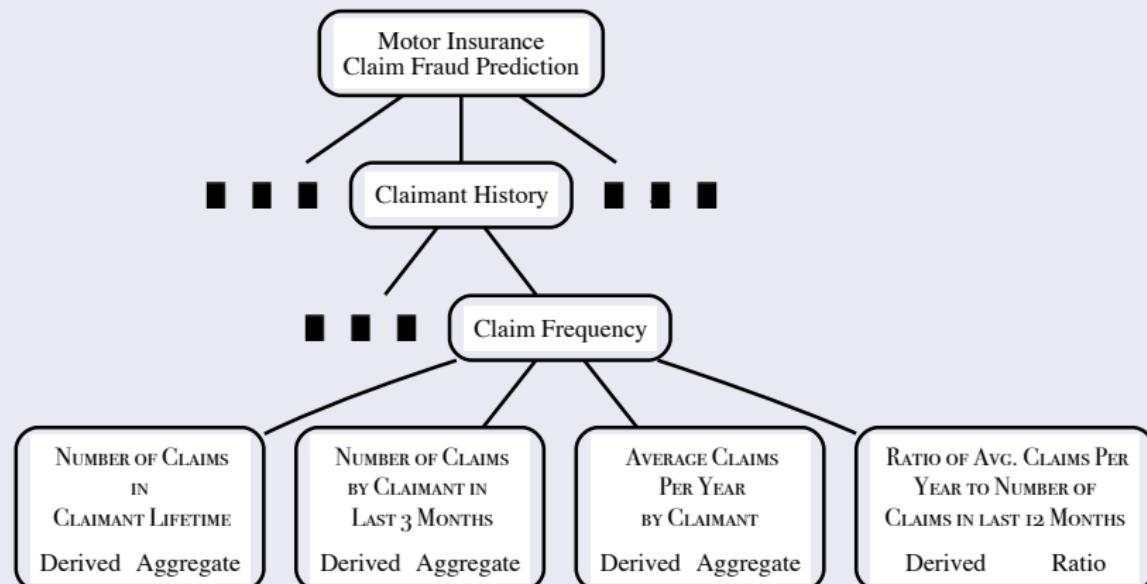


Figure: A subset of the domain concepts and related features for a motor insurance fraud prediction analytics solution.

Case Study: Motor Insurance Fraud

What features could you use to capture the Claim Types domain concept?

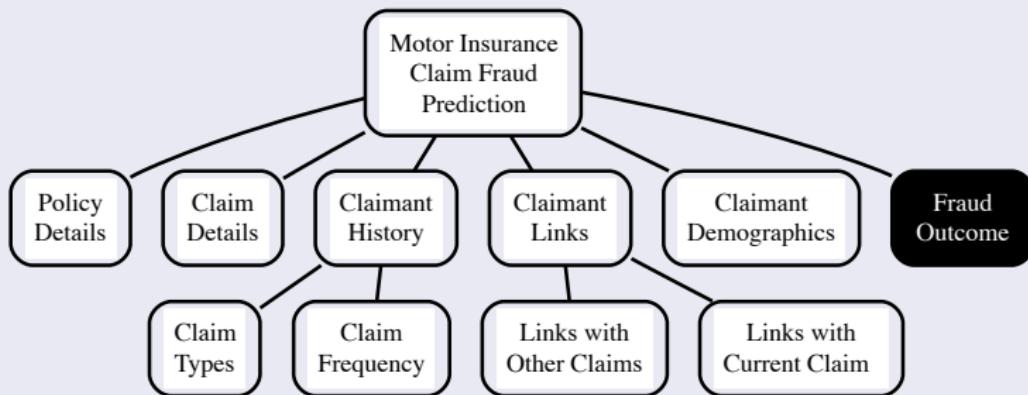


Figure: Example domain concepts for a motor insurance fraud prediction analytics solution.

Case Study: Motor Insurance Fraud

What features could you use to capture the Claim Types domain concept?

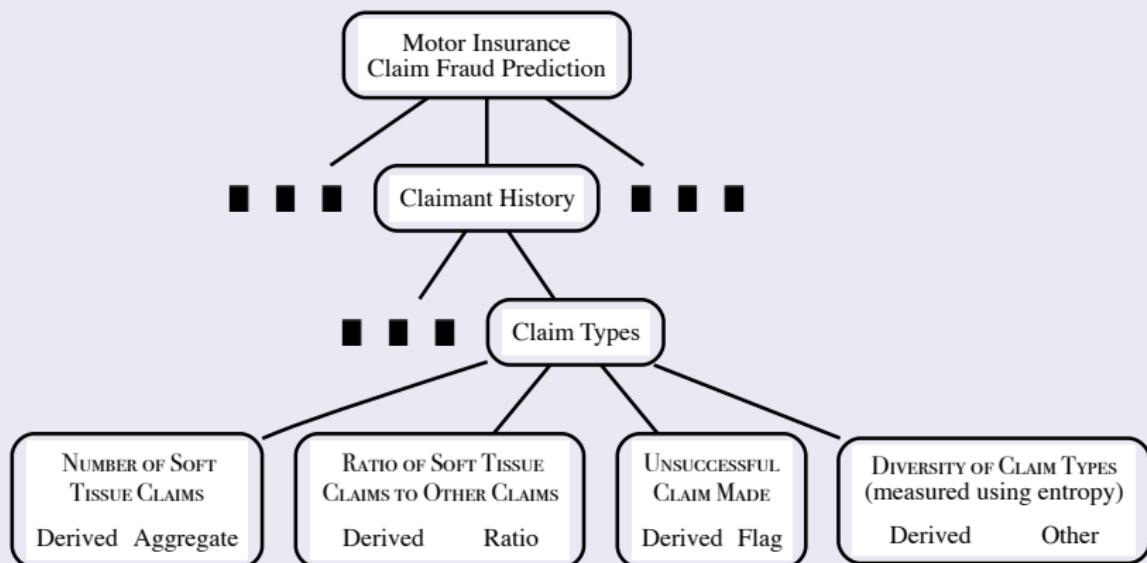


Figure: A subset of the domain concepts and related features for a motor insurance fraud prediction analytics solution.

Case Study: Motor Insurance Fraud

What features could you use to capture the Claim Details domain concept?

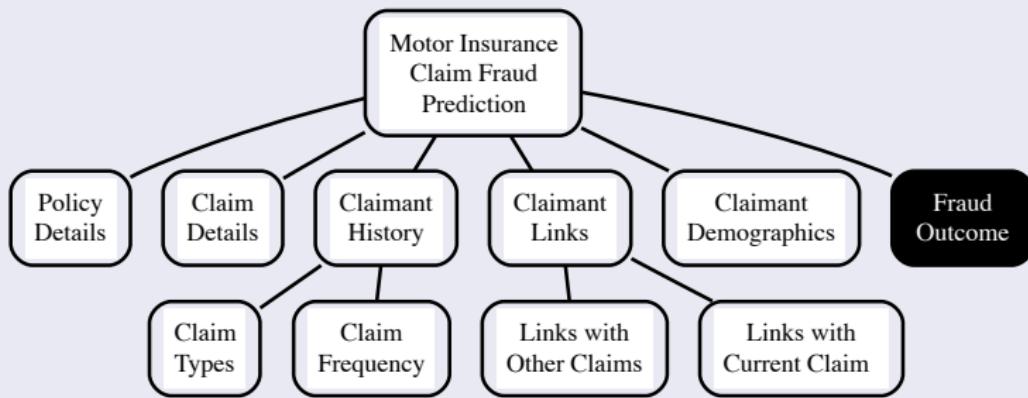


Figure: Example domain concepts for a motor insurance fraud prediction analytics solution.

Case Study: Motor Insurance Fraud

What features could you use to capture the Claim Details domain concept?

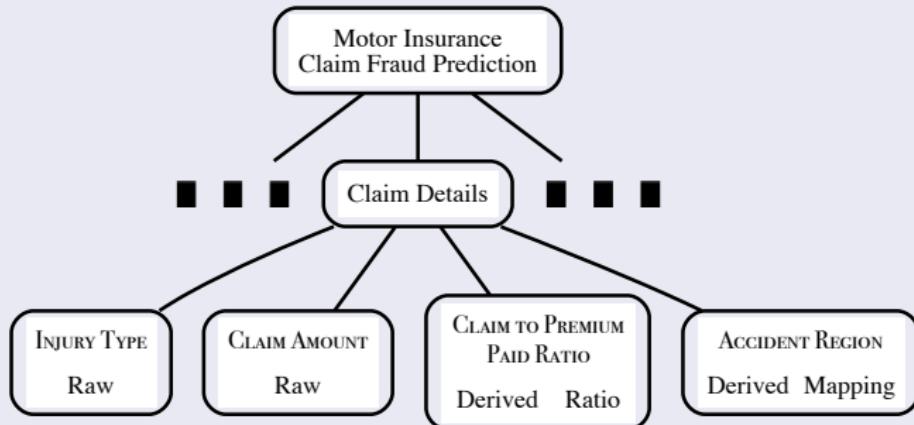


Figure: A subset of the domain concepts and related features for a motor insurance fraud prediction analytics solution.

Case Study: Motor Insurance Fraud

- The following table illustrates the structure of the final ABT that was designed for the motor insurance claims fraud detection solution.
- The table contains more descriptive features than the ones we have discussed
- The table also shows the first four instances.
- If we examine the table closely, we see a number of strange values (for example, -9 999) and a number of missing values—we will return to these in Chapter 3.

Table: The ABT for the motor insurance claims fraud detection solution.

ID	TYPE	INC.	MARITAL STATUS	NUM. CLMNTS.	INJURY TYPE	HOSPITAL STAY	CLAIM AMT.
1	CI	0		2	Soft Tissue	No	1625
2	CI	0		2	Back	Yes	15028
3	CI	54613	Married	1	Broken Limb	No	-9999
4	CI	0		3	Serious	Yes	270200
				:		:	
				:		:	

ID	TOTAL CLAIMED	NUM. CLAIMS	NUM. CLAIMS 3 MONTHS	AVG. CLAIMS PER YEAR	AVG. CLAIMS RATIO	NUM. SOFT TISSUE	% SOFT TISSUE
1	3250	2	0	1	1	2	1
2	60112	1	0	1	1	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
				:		:	
				:		:	

ID	UNSUCC. CLAIMS	CLAIM AMT. REC.	CLAIM DIV.	CLAIM TO PREM.	REGION	FRAUD FLAG
1	2	0	0	32.5	MN	1
2	0	15028	0	57.14	DL	0
3	0	572	0	-89.27	WAT	0
4	0	270200	0	30.186	DL	0
				:		
				:		

Summary

- Predictive data analytics models built using machine learning techniques are tools that we can use to help make better decisions within an organization, not an end in themselves.
- It is important to fully understand the business problem that a model is being constructed to address—this is the goal behind *converting business problems into analytics solutions*

- Predictive data analytics models are reliant on the data that is used to build them—the **analytics base table (ABT)**.
- The first step in designing an ABT is to decide on the **prediction subject**.
- An effective way in which to design ABTs is to start by defining a set of **domain concepts** in collaboration with the business, and then designing **features** that express these concepts in order to form the actual ABT.

- Features (both descriptive and target) are concrete numeric or symbolic representations of domain concepts.
- It is useful to distinguish between **raw features** that come directly from existing data sources and **derived features** that are constructed by manipulating values from existing data sources.
- Common manipulations used in this process include aggregates, flags, ratios, and mappings, although any manipulation is valid.

- The techniques described here cover the **Business Understanding**, **Data Understanding**, and (partially) **Data Preparation** phases of the **CRISP-DM** process.

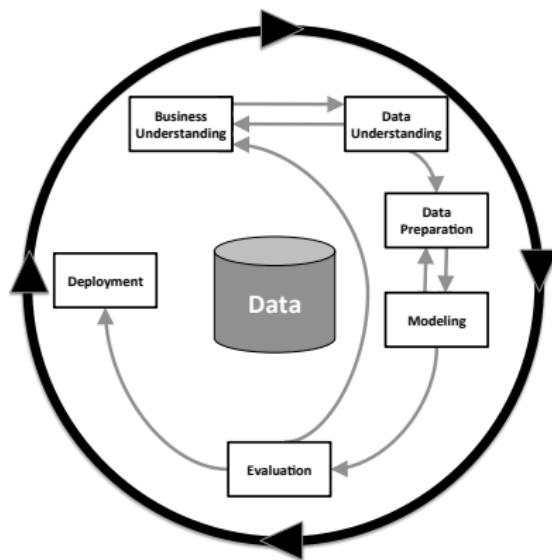


Figure: A diagram of the CRISP-DM process.

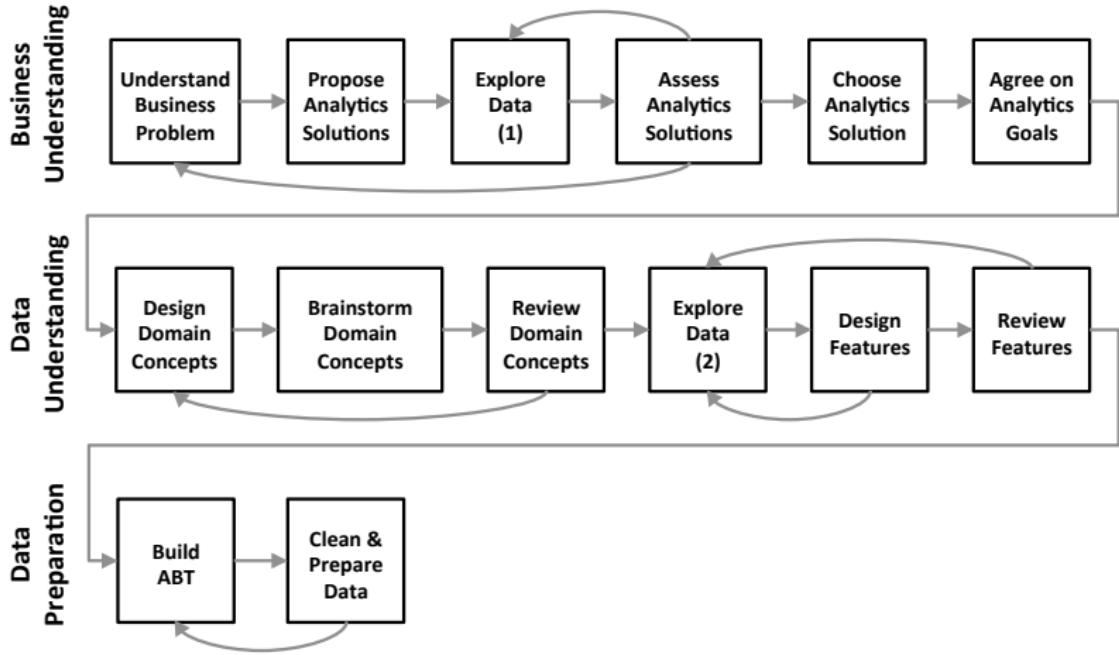


Figure: A summary of the tasks in the Business Understanding, Data Understanding, and Data Preparation phases of the **CRISP-DM** process.

- 1 **Converting Business Problems into Analytics Solutions**
 - Case Study: Motor Insurance Fraud
- 2 **Assessing Feasibility**
 - Case Study: Motor Insurance Fraud
- 3 **Designing the Analytics Base Table**
 - Case Study: Motor Insurance Fraud
- 4 **Designing & Implementing Features**
 - Different Types of Data
 - Different Types of Features
 - Handling Time
 - Legal Issues
 - Implementing Features
 - Case Study: Motor Insurance Fraud
- 5 **Summary**

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 3: Data Exploration
Sections 3.1, 3.2, 3.3, 3.4

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

1 The Data Quality Report

- Case Study: Motor Insurance Fraud

2 Getting To Know The Data

- Case Study: Motor Insurance Fraud

3 Identifying Data Quality Issues

- Case Study: Motor Insurance Fraud

4 Handling Data Quality Issues

- Handling Missing Values
- Handling Outliers
- Case Study: Motor Insurance Fraud

5 Summary

The Data Quality Report



- A data quality report includes tabular reports that describe the characteristics of each feature in an ABT using standard statistical measures of **central tendency** and **variation**.
- The tabular reports are accompanied by data visualizations:
 - A **histogram** for each continuous feature in an ABT.
 - A **bar plot** for each categorical feature in an ABT.

oooooooo

○○○○○

Table: The structures of the tables included in a data quality report to describe (a) continuous features and (b) categorical features.

(a) Continuous Features

(b) Categorical Features

Case Study: Motor Insurance Fraud

The following slides show a portion of the ABT that has been developed for the motor insurance claims fraud detection.

A portion of the ABT developed for this solution is shown first.

Table: Portions of the ABT for the motor insurance claims fraud detection problem.

ID	TYPE	INC.	MARITAL STATUS	NUM CLMNTS.	INJURY TYPE	HOSPITAL STAY	CLAIM AMNT.	TOTAL CLAIMED	NUM CLAIMS	NUM SOFT TISS.	% SOFT TISS.	CLAIM AMT RCVD.	FRAUD FLAG
1	CI	0		2	Soft Tissue	No	1,625	3250	2	2	1.0	0	1
2	CI	0		2	Back	Yes	15,028	60,112	1	0	0	15,028	0
3	CI	54,613	Married	1	Broken Limb	No	-99,999	0	0	0	0	572	0
4	CI	0		4	Broken Limb	Yes	5,097	11,661	1	1	1.0	7,864	0
5	CI	0		4	Soft Tissue	No	8869	0	0	0	0	0	1
6	CI	0		1	Broken Limb	Yes	17,480	0	0	0	0	17,480	0
7	CI	52,567	Single	3	Broken Limb	No	3,017	18,102	2	1	0.5	0	1
8	CI	0		2	Back	Yes	7463	0	0	0	0	7,463	0
9	CI	0		1	Soft Tissue	No	2,067	0	0	0	0	2,067	0
10	CI	42,300	Married	4	Back	No	2,260	0	0	0	0	2,260	0
300	CI	0		2	Broken Limb	No	2,244	0	0	0	0	2,244	0
301	CI	0		1	Broken Limb	No	1,627	92,283	3	0	0	1,627	0
302	CI	0		3	Serious	Yes	270,200	0	0	0	0	270,200	0
303	CI	0		1	Soft Tissue	No	7,668	92,806	3	0	0	7,668	0
304	CI	46,365	Married	1	Back	No	3,217	0	0	0	0	1,653	0
458	CI	48,176	Married	3	Soft Tissue	Yes	4,653	8,203	1	0	0	4,653	0
459	CI	0		1	Soft Tissue	Yes	881	51,245	3	0	0	0	1
460	CI	0		3	Back	No	8,688	729,792	56	5	0.08	8,688	0
461	CI	47,371	Divorced	1	Broken Limb	Yes	3,194	11,668	1	0	0	3,194	0
462	CI	0		1	Soft Tissue	No	6,821	0	0	0	0	0	1
491	CI	40,204	Single	1	Back	No	75,748	11,116	1	0	0	0	1
492	CI	0		1	Broken Limb	No	6,172	6,041	1	0	0	6,172	0
493	CI	0		1	Soft Tissue	Yes	2,569	20,055	1	0	0	2,569	0
494	CI	31,951	Married	1	Broken Limb	No	5,227	22,095	1	0	0	5,227	0
495	CI	0		2	Back	No	3,813	9,882	3	0	0	0	1
496	CI	0		1	Soft Tissue	No	2,118	0	0	0	0	0	1
497	CI	29,280	Married	4	Broken Limb	Yes	3,199	0	0	0	0	0	1
498	CI	0		1	Broken Limb	Yes	32,469	0	0	0	0	16,763	0
499	CI	46,683	Married	1	Broken Limb	No	179,448	0	0	0	0	179,448	0
500	CI	0		1	Broken Limb	No	8,259	0	0	0	0	0	1

Table: A data quality report for the motor insurance claims fraud detection ABT

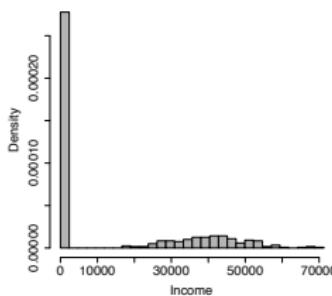
(a) Continuous Features

Feature	Count	Miss.	Card.	1 st				3 rd		Std. Dev.
				Min	Qrt.	Mean	Median	Qrt.	Max	
INCOME	500	0.0	171	0.0	0.0	13,740.0	0.0	33,918.5	71,284.0	20,081.5
NUM CLAIMANTS	500	0.0	4	1.0	1.0	1.9	2	3.0	4.0	1.0
CLAIM AMOUNT	500	0.0	493	-99,999	3,322.3	16,373.2	5,663.0	12,245.5	270,200.0	29,426.3
TOTAL CLAIMED	500	0.0	235	0.0	0.0	9,597.2	0.0	11,282.8	729,792.0	35,655.7
NUM CLAIMS	500	0.0	7	0.0	0.0	0.8	0.0	1.0	56.0	2.7
NUM SOFT TISSUE	500	2.0	6	0.0	0.0	0.2	0.0	0.0	5.0	0.6
% SOFT TISSUE	500	0.0	9	0.0	0.0	0.2	0.0	0.0	2.0	0.4
AMOUNT RECEIVED	500	0.0	329	0.0	0.0	13,051.9	3,253.5	8,191.8	295,303.0	30,547.2
FRAUD FLAG	500	0.0	2	0.0	0.0	0.3	0.0	1.0	1.0	0.5

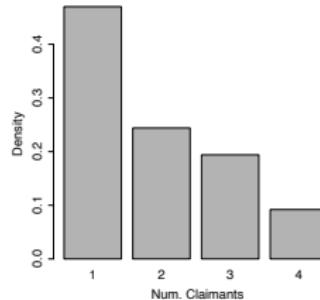
Table: A data quality report for the motor insurance claims fraud detection ABT.

(a) Categorical Features

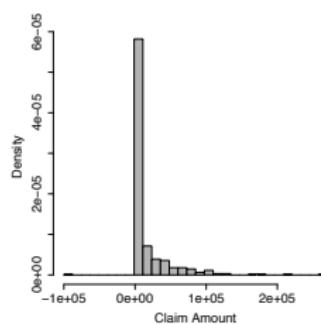
Feature	Count	Miss.	Card.	Mode	Mode	Mode	<i>2nd</i> Mode	<i>2nd</i> Mode	<i>2nd</i> Mode
					Freq.	%		Freq.	%
INSURANCE TYPE	500	0.0	1	CI	500	1.0	—	—	—
MARITAL STATUS	500	61.2	4	Married	99	51.0	Single	48	24.7
INJURY TYPE	500	0.0	4	Broken Limb	177	35.4	Soft Tissue	172	34.4
HOSPITAL STAY	500	0.0	2	No	354	70.8	Yes	146	29.2



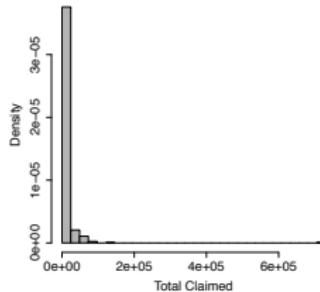
(a) INCOME



(b) NUM CLAIMANTS

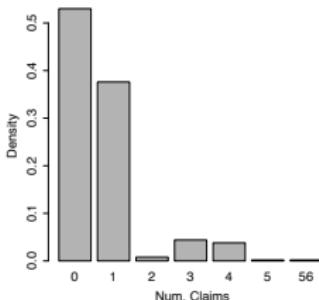


(c) CLAIM AMOUNT

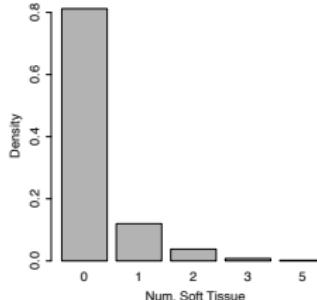


(d) TOTAL CLAIMED

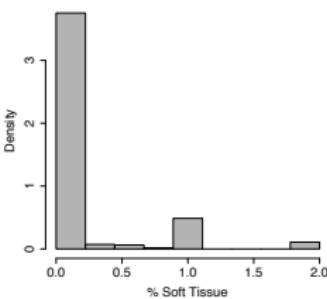
Figure: Visualizations of the continuous and categorical features in the motor insurance claims fraud detection ABT in Table 2 [7].



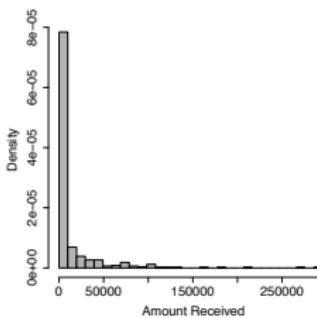
(a) NUM CLAIMS



(b) NUM SOFT TISSUE



(c) % SOFT TISSUE



(d) AMOUNT RECEIVED

Figure: Visualizations of the continuous and categorical features in the motor insurance claims fraud detection ABT in Table 2 [7].

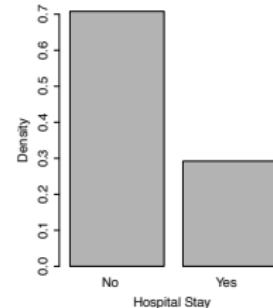
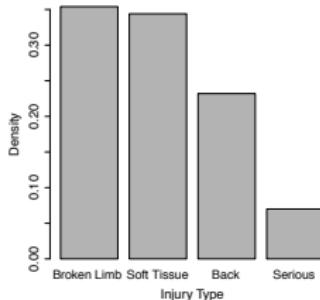
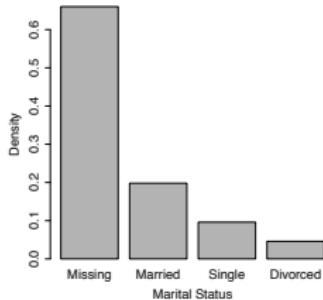
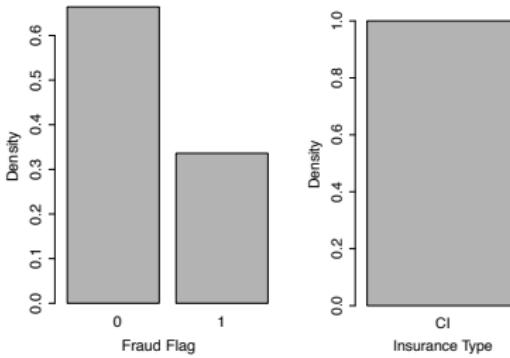


Figure: Visualizations of the continuous and categorical features in the motor insurance claims fraud detection ABT in Table 2 [7].



(a) FRAUD FLAG

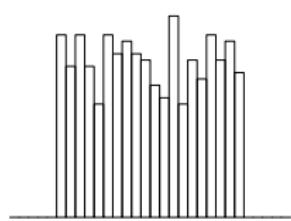
(b) INSURANCE
TYPE

Figure: Visualizations of the continuous and categorical features in the motor insurance claims fraud detection ABT in Table 2 [7].

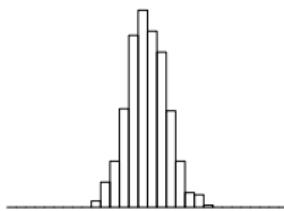
Getting To Know The Data

- For categorical features, we should:
 - Examine the mode, 2nd mode, mode %, and 2nd mode % as these tell us the most common levels within these features and will identify if any levels dominate the dataset.
- For continuous features we should:
 - Examine the mean and standard deviation of each feature to get a sense of the central tendency and variation of the values within the dataset for the feature.
 - Examine the minimum and maximum values to understand the range that is possible for each feature.

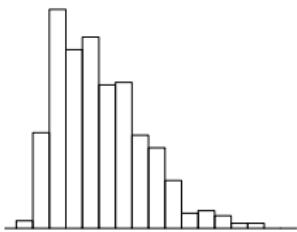
- When we generate histograms of features there are a number of common, well understood shapes that we should look out for.



(a) Uniform

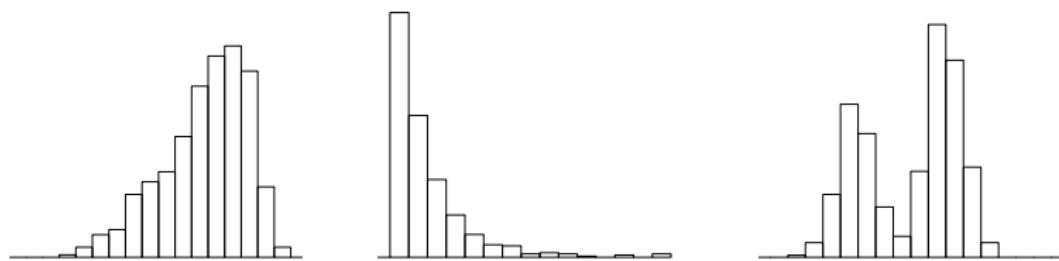


(b) Normal (Unimodal)



(c) Unimodal (skewed right)

Figure: Histograms for different sets of data each of which exhibit well-known, common characteristics.



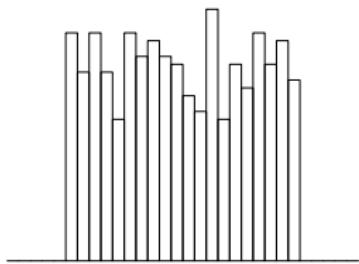
(a) Unimodal (skewed left)

(b) Exponential

(c) Multimodal

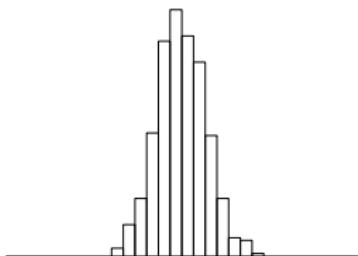
Figure: Histograms for different sets of data each of which exhibit well-known, common characteristics.

- A uniform distribution indicates that a feature is equally likely to take a value in any of the ranges present.



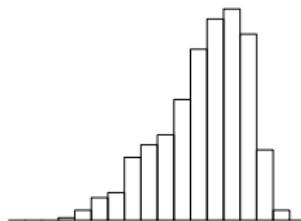
Uniform

- Features following a normal distribution are characterized by a strong tendency towards a central value and symmetrical variation to either side of this.

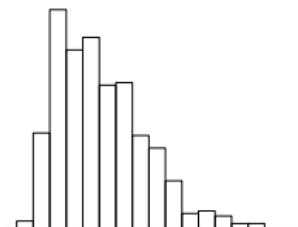


Normal (Unimodal)

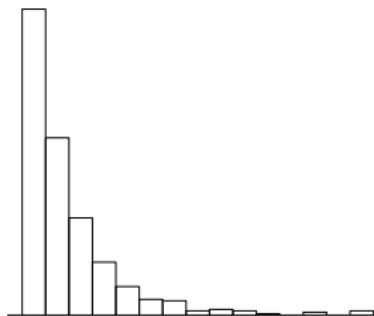
- Skew is simply a tendency towards very high (**right skew**) or very low (**left skew**) values.



Unimodal (skewed left)

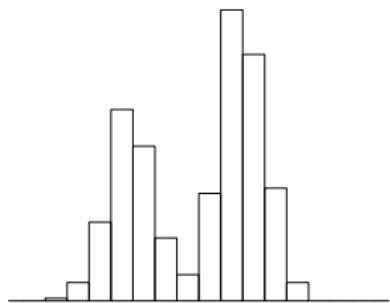


Unimodal (skewed right)



Exponential

- In a feature following an **exponential distribution** the likelihood of occurrence of a small number of low values is very high, but sharply diminishes as values increase.



Multimodal

- A feature characterized by a **multimodal distribution** has two or more very commonly occurring ranges of values that are clearly separated.

- The probability density function for the **normal** distribution (or **Gaussian distribution**) is

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

where x is any value, and μ and σ are parameters that define the shape of the distribution: the **population mean** and **population standard deviation**.

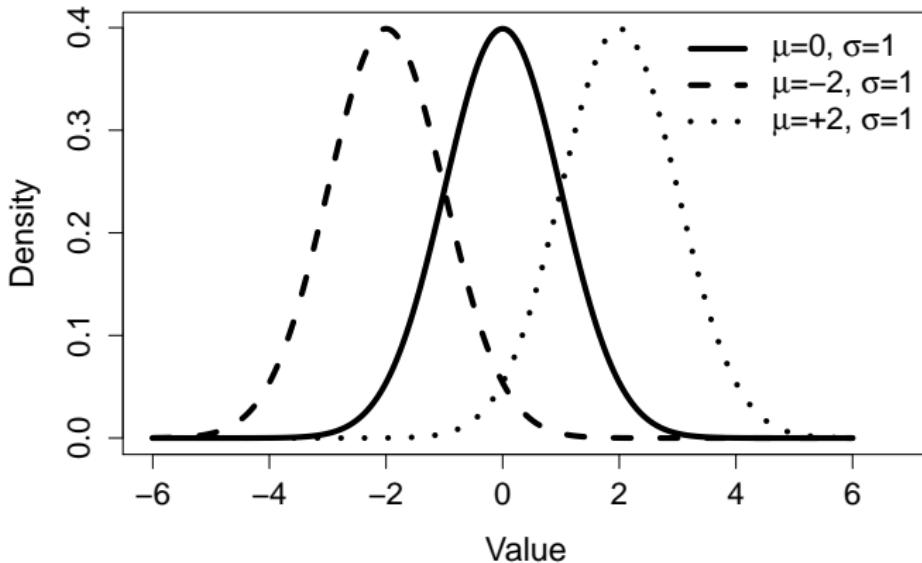


Figure: Three normal distributions with different means but identical standard deviations.

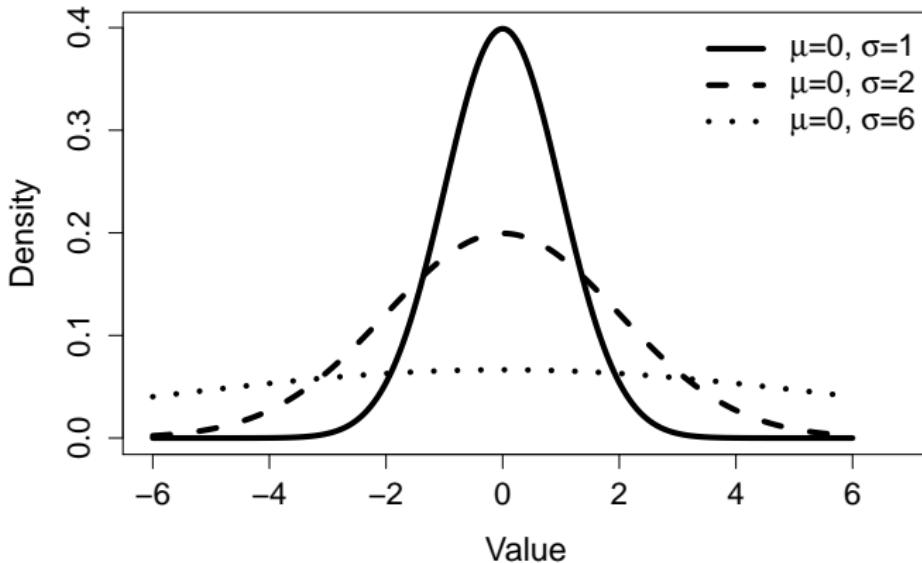


Figure: Three normal distributions with identical means but different standard deviations.

- The 68 – 95 – 99.7 rule is a useful characteristic of the normal distribution.
- The rule states that approximately:
 - 68% of the observations will be within one σ of μ
 - 95% of observations will be within two σ of μ
 - 99.7% of observations will be within three σ of μ .

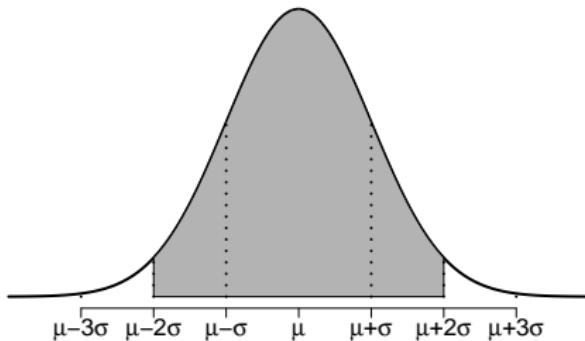


Figure: An illustration of the 68 – 95 – 99.7 percentage rule that a normal distribution defines as the expected distribution of observations. The grey region defines the area where 95% of observations are expected.

Case Study: Motor Insurance Fraud

Case Study: Motor Insurance Fraud

Examine the data quality report for the motor insurance fraud prediction scenario and comment on the central tendency and variation of each feature.

Identifying Data Quality Issues

- A **data quality issue** is loosely defined as anything *unusual* about the data in an ABT.
- The most common data quality issues are:
 - **missing values**
 - **irregular cardinality**
 - **outliers**

- The data quality issues we identify from a data quality report will be of two types:
 - Data quality issues due to **invalid data**.
 - Data quality issues due to **valid data**.

Table: The structure of a data quality plan.

Feature	Data Quality Issue	Potential Handling Strategies

Case Study: Motor Insurance Fraud

Table: The data quality plan for the motor insurance fraud prediction ABT.

Feature	Data Quality Issue	Potential Handling Strategies
NUM SOFT TISSUE	Missing values (2%)	
CLAIM AMOUNT	Outliers (high)	
AMOUNT RECEIVED	Outliers (high)	

Handling Data Quality Issues

Handling Missing Values

- Approach 1: Drop any features that have missing value.
- Approach 2: Apply **complete case analysis**.
- Approach 3: Derive a **missing indicator feature** from features with missing value.

Handling Missing Values

- **Imputation** replaces missing feature values with a plausible estimated value based on the feature values that are present.
- The most common approach to imputation is to replace missing values for a feature with a measure of the central tendency of that feature.
- We would be reluctant to use imputation on features missing in excess of 30% of their values and would strongly recommend against the use of imputation on features missing in excess of 50% of their values.

Handling Outliers

- The easiest way to handle outliers is to use a **clamp transformation** that clamps all values above an upper threshold and below a lower threshold to these threshold values, thus removing the offending outliers

$$a_i = \begin{cases} \text{lower} & \text{if } a_i < \text{lower} \\ \text{upper} & \text{if } a_i > \text{upper} \\ a_i & \text{otherwise} \end{cases} \quad (2)$$

where a_i is a specific value of feature a , and lower and upper are the lower and upper thresholds.

Case Study: Motor Insurance Fraud

Case Study: Motor Insurance Fraud

What handling strategies would you recommend for the data quality issues found in the motor Insurance fraud ABT?

Case Study: Motor Insurance Fraud

Table: The data quality plan for the motor insurance fraud prediction ABT.

Feature	Data Quality Issue	Potential Handling Strategies
NUM SOFT TISSUE	Missing values (2%)	Imputation (median: 0.0)
CLAIM AMOUNT	Outliers (high)	Clamp transformation (manual: 0, 80 000)
AMOUNT RECEIVED	Outliers (high)	Clamp transformation (manual: 0, 80 000)

Summary

- The key outcomes of the **data exploration** process are that the practitioner should
 - 1 Have *gotten to know* the features within the ABT, especially their central tendencies, variations, and **distributions**.
 - 2 Have identified any **data quality issues** within the ABT, in particular **missing values**, **irregular cardinality**, and **outliers**.
 - 3 Have corrected any data quality issues due to **invalid data**.
 - 4 Have recorded any data quality issues due to **valid data** in a **data quality plan** along with potential handling strategies.
 - 5 Be confident that enough good quality data exists to continue with a project.

1 The Data Quality Report

- Case Study: Motor Insurance Fraud

2 Getting To Know The Data

- Case Study: Motor Insurance Fraud

3 Identifying Data Quality Issues

- Case Study: Motor Insurance Fraud

4 Handling Data Quality Issues

- Handling Missing Values
- Handling Outliers
- Case Study: Motor Insurance Fraud

5 Summary

Data Quality Report – Case Study

Fundamentals of Machine Learning for Predictive Data Analytics,

John Kelleher and Brian Mac Namee and Aoife D'Arcy

Chapter 3

Case Study: Analytical Base Table

ID	TYPE	INC.	MARITAL STATUS	NUM CLMNTS.	INJURY TYPE	HOSPITAL STAY	CLAIM AMNT.	TOTAL CLAIMED	NUM CLAIMS	NUM SOFT TISS.	% SOFT TISS.	CLAIM AMT RCVD.	FRAUD FLAG
1	CI	0		2	Soft Tissue	No	1,625	3250	2	2	1.0	0	1
2	CI	0		2	Back	Yes	15,028	60,112	1	0	0	15,028	0
3	CI	54,613	Married	1	Broken Limb	No	-99,999	0	0	0	0	572	0
4	CI	0		4	Broken Limb	Yes	5,097	11,661	1	1	1.0	7,864	0
5	CI	0		4	Soft Tissue	No	8869	0	0	0	0	0	1
6	CI	0		1	Broken Limb	Yes	17,480	0	0	0	0	17,480	0
7	CI	52,567	Single	3	Broken Limb	No	3,017	18,102	2	1	0.5	0	1
8	CI	0		2	Back	Yes	7463	0	0	0	0	7,463	0
9	CI	0		1	Soft Tissue	No	2,067	0	0	0	0	2,067	0
10	CI	42,300	Married	4	Back	No	2,260	0	0	0	0	2,260	0
⋮													
300	CI	0		2	Broken Limb	No	2,244	0	0	0	0	2,244	0
301	CI	0		1	Broken Limb	No	1,627	92,283	3	0	0	1,627	0
302	CI	0		3	Serious	Yes	270,200	0	0	0	0	270,200	0
303	CI	0		1	Soft Tissue	No	7,668	92,806	3	0	0	7,668	0
304	CI	46,365	Married	1	Back	No	3,217	0	0	0	0	1,653	0
⋮													
458	CI	48,176	Married	3	Soft Tissue	Yes	4,653	8,203	1	0	0	4,653	0
459	CI	0		1	Soft Tissue	Yes	881	51,245	3	0	0	0	1
460	CI	0		3	Back	No	8,688	729,792	56	5	0.08	8,688	0
461	CI	47,371	Divorced	1	Broken Limb	Yes	3,194	11,668	1	0	0	3,194	0
462	CI	0		1	Soft Tissue	No	6,821	0	0	0	0	0	1
⋮													
491	CI	40,204	Single	1	Back	No	75,748	11,116	1	0	0	0	1
492	CI	0		1	Broken Limb	No	6,172	6,041	1	0	0	6,172	0
493	CI	0		1	Soft Tissue	Yes	2,569	20,055	1	0	0	2,569	0
494	CI	31,951	Married	1	Broken Limb	No	5,227	22,095	1	0	0	5,227	0
495	CI	0		2	Back	No	3,813	9,882	3	0	0	0	1
496	CI	0		1	Soft Tissue	No	2,118	0	0	0	0	0	1
497	CI	29,280	Married	4	Broken Limb	Yes	3,199	0	0	0	0	0	1
498	CI	0		1	Broken Limb	Yes	32,469	0	0	0	0	16,763	0
499	CI	46,683	Married	1	Broken Limb	No	179,448	0	0	0	0	179,448	0
500	CI	0		1	Broken Limb	No	8,259	0	0	0	0	0	1

Case study: Data Quality Reports

(a) Continuous Features

Feature	Count	Miss.	Card.	1 st			3 rd		Std. Dev.	
				Min	Qrt.	Mean	Median	Qrt.		
INCOME	500	0.0	171	0.0	0.0	13,740.0	0.0	33,918.5	71,284.0	20,081.5
NUM CLAIMANTS	500	0.0	4	1.0	1.0	1.9	2	3.0	4.0	1.0
CLAIM AMOUNT	500	0.0	493	-99,999	3,322.3	16,373.2	5,663.0	12,245.5	270,200.0	29,426.3
TOTAL CLAIMED	500	0.0	235	0.0	0.0	9,597.2	0.0	11,282.8	729,792.0	35,655.7
NUM CLAIMS	500	0.0	7	0.0	0.0	0.8	0.0	1.0	56.0	2.7
NUM SOFT TISSUE	500	2.0	6	0.0	0.0	0.2	0.0	0.0	5.0	0.6
% SOFT TISSUE	500	0.0	9	0.0	0.0	0.2	0.0	0.0	2.0	0.4
AMOUNT RECEIVED	500	0.0	329	0.0	0.0	13,051.9	3,253.5	8,191.8	295,303.0	30,547.2
FRAUD FLAG	500	0.0	2	0.0	0.0	0.3	0.0	1.0	1.0	0.5

(a) Categorical Features

Feature	Count	Miss.	Card.	Mode		Mode	% Freq.	2 nd	
				Mode	Freq.			Mode	Mode
INSURANCE TYPE	500	0.0	1	CI	500	1.0	—	—	—
MARITAL STATUS	500	61.2	4	Married	99	51.0	Single	48	24.7
INJURY TYPE	500	0.0	4	Broken Limb	177	35.4	Soft Tissue	172	34.4
HOSPITAL STAY	500	0.0	2	No	354	70.8	Yes	146	29.2

Analysis – Missing values

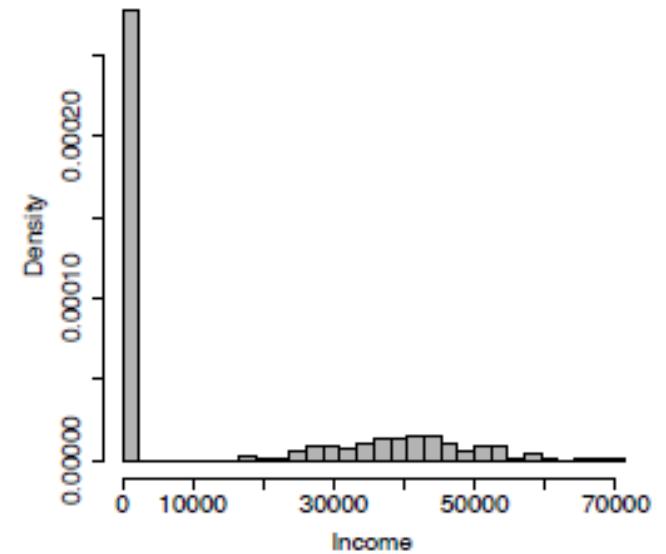
NUM SOFT TISSUE – 2%

Possible solution: leave it as it is, document it in the data quality plan

MARITAL STATUS – 61.2%

Possible solution – remove the feature

- Examining the raw data we notice that the fields where MARITAL STATUS is missing we have income value 0
- Unusual pattern for INCOME from the histogram large number of 0s
- Discussion with the business revealed that MARITAL STATUS and INCOME were collected together, and INCOME 0 represents a missing value



(a) INCOME

Decision → remove both MARITAL STATUS and INCOME from ABT

Analysis – Irregular cardinality

INSURANCE TYPE – cardinality 1

Nothing wrong with the data, refers to the type of claim (CI = Car Insurance).
However, not useful for predictions

Decision → feature is removed from the ABT

Analysis – Irregular cardinality

Continuous features with low cardinality:

NUM CLAIMANTS, NUM CLAIMS, SOFT TISSUE, SOFT TISSUE %, FRAUD FLAG –
all have cardinality <10 (dataset of 500 instances)

- Discussing with the business indicates that NUM CLAIMANTS, NUM CLAIMS,
SOFT TISSUE, SOFT TISSUE% naturally take low number of values

Decision → no issues

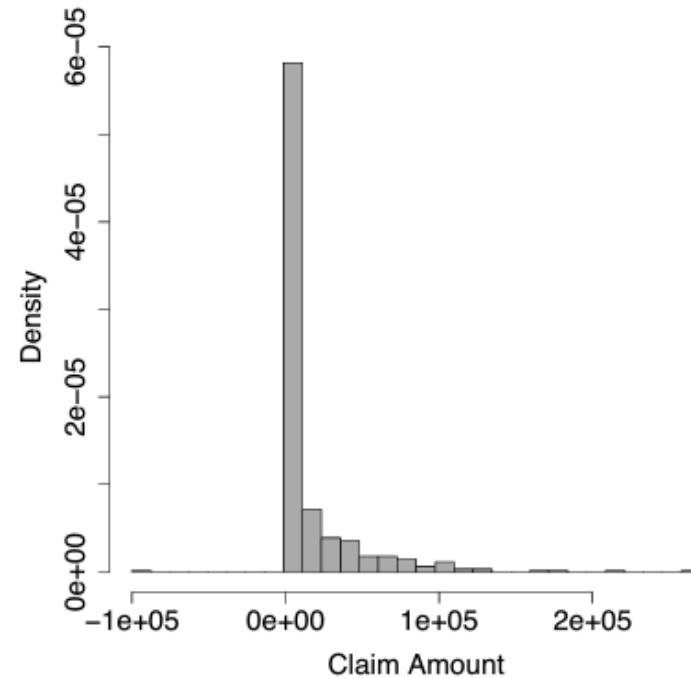
- FRAUD FLAG has cardinality 2, indicates categorical feature labelled as continuous
with values 0 and 1.

Decision → change FRAUD FLAG to categorical feature (note – this is also our
target feature, so the type will have large impact on the ML approach we take)

Analysis - Outliers

CLAIM AMOUNT – min value -99,999

- Examining the raw data we notice this value comes from the d_3 in our table.
- Examining the histogram for CLAIM AMOUNT we don't see a large bar at that value, so this is isolated instance
- Value looks like input error, which was confirmed with the business



Decision → invalid outlier, remove it and replace it with missing value

Analysis - Outliers

CLAIM AMOUT, TOTAL CLAIMED, NUM CLAIMS, AMOUNT RECIEVED –
all have unusually high maximum values, especially compared to
median and 3rd quartile

TOTAL CLAIMED, NUM CLAIMS – both high values are from d_{460} in ABT

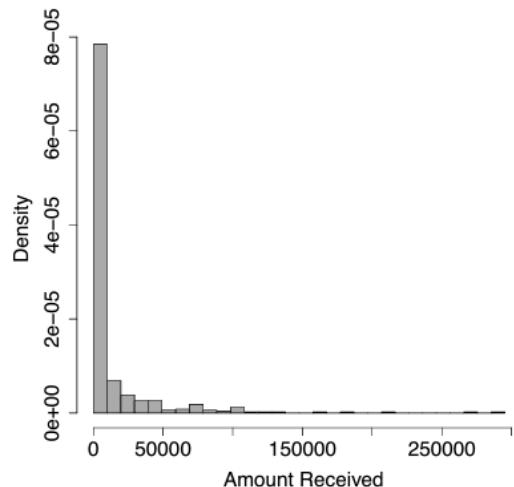
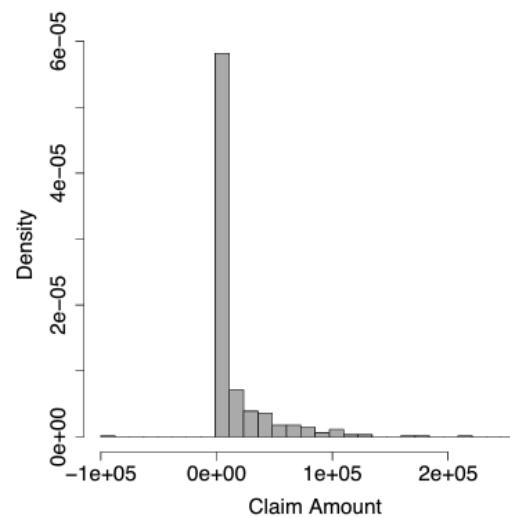
- This policy seems to have made many more claims compared to the others
- Investigation with the business shows this is a valid outlier, however this was a company policy and was included in the ABT by mistake.

Decision → d_{460} was removed from the ABT

Analysis - Outliers

CLAIM AMOUT and AMOUNT RECEIVED – both from d_{302} in ABT

- Examining the raw data we note this is a large claim for a serious injury
- Analysis of the histograms show that the large values are no unique (several small bars in the right hand side of the histograms)



Decision → document the outliers and possibly handle them later

Handling missing values

Approach 1: Drop any features that have missing value

Rule of thumb – only use if 60% or more of values missing, otherwise look for different approach to handle

Approach 2: Apply **complete case analysis**.

Delete any instances where one or more features are missing values

Can result in significant data loss, can introduce bias (if distribution of missing values is not random)

Recommendation: Remove only instances with missing value for target feature

Approach 3: Derive a **missing indicator feature** from features with missing value.

Binary feature that flags whether the value was present or missing in the original data.
May be useful if the reason the features are missing is related to the target feature

Handling missing values

Imputation replaces missing feature values with a plausible estimated value based on the feature values that are present.

The most common approach to imputation is to replace missing values for a feature with a measure of the central tendency of that feature.

- Continuous features – usually mean and median
- Categorical features – usually the mode

We would be reluctant to use imputation on features missing in excess of 30% of their values and would strongly recommend against the use of imputation on features missing in excess of 50% of their values.

Handling outliers – clamp transformation

The easiest way to handle outliers is to use a **clamp transformation** that clamps all values above an upper threshold and below a lower threshold to these threshold values, thus removing the offending outliers

$$a_i = \begin{cases} \text{lower} & \text{if } a_i < \text{lower} \\ \text{upper} & \text{if } a_i > \text{upper} \\ a_i & \text{otherwise} \end{cases} \quad (2)$$

where a_i is a specific value of feature a , and lower and upper are the lower and upper thresholds.

Handling outliers - clamp transformation

Upper and lower limits can be set manually based on domain knowledge, or can be calculated from the data

Method 1:

lower = 1st quartile value – 1.5*inter-quartile range

higher = 3rd quartile value + 1.5*inter-quartile range

Method 2:

Use the mean value of a feature \pm 2 times standard deviation

Handling outliers – clamp transformation

Clamp transformation – for and against

Performing clamp transformation may remove the most interesting (and predictive) data from the dataset

However, some machine learning techniques perform poorly in presence of outliers

Clamp transformations are only recommended when you suspect the outliers will affect the performance of the model

Case study: Data Quality Plan

Table: The data quality plan for the motor insurance fraud prediction ABT.

Feature	Data Quality Issue	Potential Handling Strategies
NUM SOFT TISSUE	Missing values (2%)	
CLAIM AMOUNT	Outliers (high)	
AMOUNT RECEIVED	Outliers (high)	

Case study: Data Quality Plan

NUM SOFT TISSUE

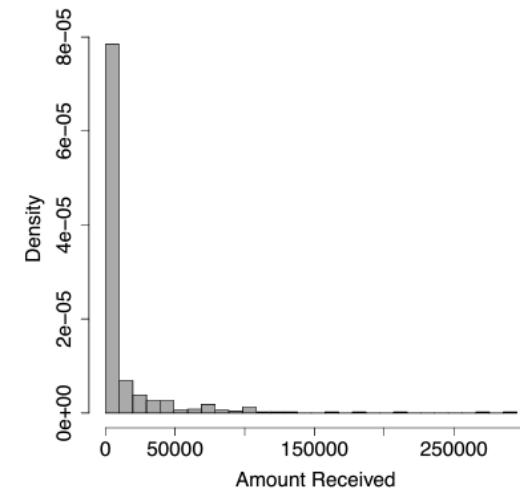
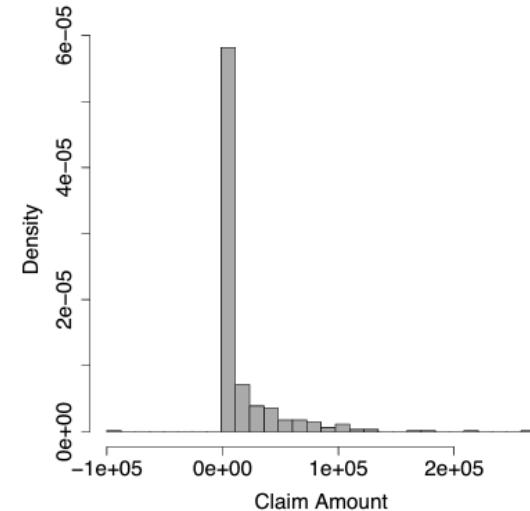
Missing only 2% of the values, we'll use imputation.

We'll use the mean or the median (value 0.2 doesn't naturally occur in the data set, so we'll use 0)

Case study: Data Quality Plan

CLAIM AMOUNT, AMOUNT RECEIVED

- we'll use clamp transformation
- exponential distribution, so methods we discussed won't work too well
- following a discussion with the business we were advised that lower limit of 0 and upper limit of 80,000 make sense for the clamp transformation



Case study: Data Quality Plan

Case Study: Motor Insurance Fraud

Table: The data quality plan for the motor insurance fraud prediction ABT.

Feature	Data Quality Issue	Potential Handling Strategies
NUM SOFT TISSUE	Missing values (2%)	Imputation (median: 0.0)
CLAIM AMOUNT	Outliers (high)	Clamp transformation (manual: 0, 80 000)
AMOUNT RECEIVED	Outliers (high)	Clamp transformation (manual: 0, 80 000)

Visualising relationship between features

In preparation to create predictive models it's always a good idea to investigate the relationship between the variables

This can identify pairs of closely-related features (which in turn can be used to reduce the size of the ABT)

Table 3.7

The details of a professional basketball team.

ID	POSITION	HEIGHT	WEIGHT	CAREER	STAGE	AGE	SPONSORSHIP	SHOE	SPONSOR
1	forward	192	218	veteran	29		561	yes	
2	center	218	251	mid-career	35		60	no	
3	forward	197	221	rookie	22		1,312	no	
4	forward	192	219	rookie	22		1,359	no	
5	forward	198	223	veteran	29		362	yes	
6	guard	166	188	rookie	21		1,536	yes	
7	forward	195	221	veteran	25		694	no	
8	guard	182	199	rookie	21		1,678	yes	
9	guard	189	199	mid-career	27		385	yes	
10	forward	205	232	rookie	24		1,416	no	
11	center	206	246	mid-career	29		314	no	
12	guard	185	207	rookie	23		1,497	yes	
13	guard	172	183	rookie	24		1,383	yes	
14	guard	169	183	rookie	24		1,034	yes	
15	guard	185	197	mid-career	29		178	yes	
16	forward	215	232	mid-career	30		434	no	
17	guard	158	184	veteran	29		162	yes	
18	guard	190	207	mid-career	27		648	yes	
19	center	195	235	mid-career	28		481	no	
20	guard	192	200	mid-career	32		427	yes	
21	forward	202	220	mid-career	31		542	no	
22	forward	184	213	mid-career	32		12	no	
23	forward	190	215	rookie	22		1,179	no	
24	guard	178	193	rookie	21		1,078	no	
25	guard	185	200	mid-career	31		213	yes	
26	forward	191	218	rookie	19		1,855	no	
27	center	196	235	veteran	32		47	no	
28	forward	198	221	rookie	22		1,409	no	
29	center	207	247	veteran	27		1,065	no	
30	center	201	244	mid-career	25		1,111	yes	

Scatter plot (continuous features)

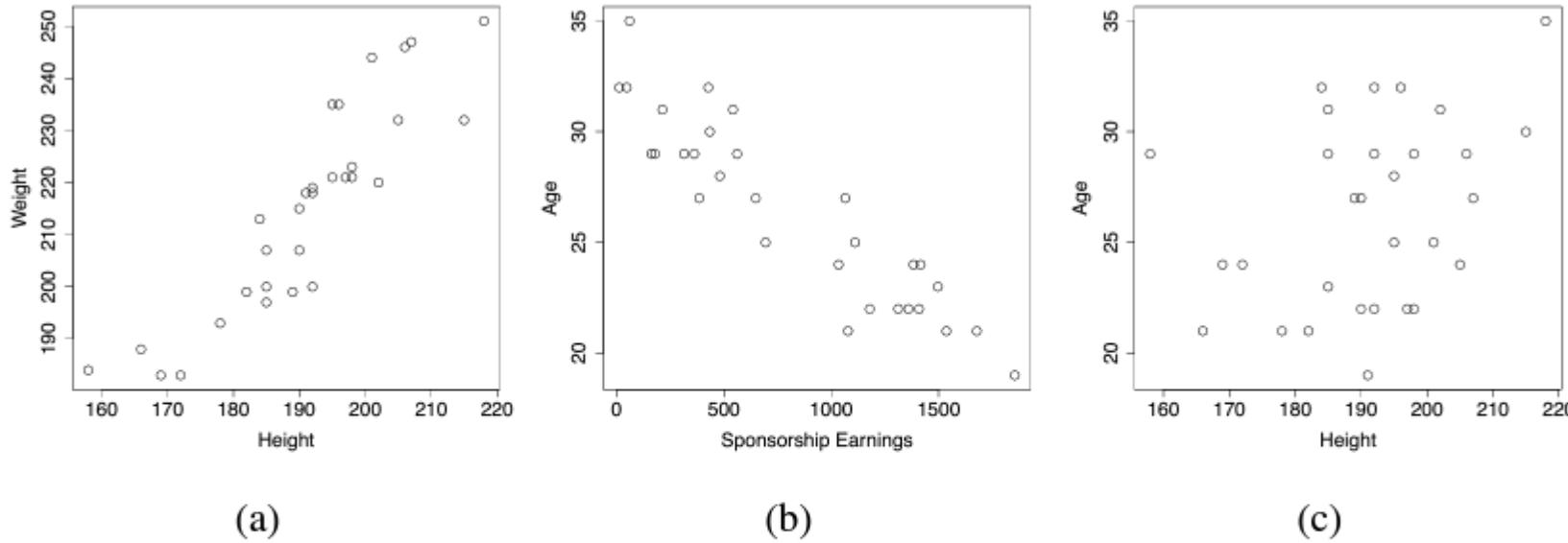


Figure 3.5

Example scatter plots for pairs of features from the dataset in Table 3.7^[78], showing (a) the strong positive covariance between HEIGHT and WEIGHT; (b) the strong negative covariance between SPONSORSHIP EARNINGS and AGE; and (c) the lack of strong covariance between HEIGHT and AGE.

SPLOM (Scatter plot matrix)

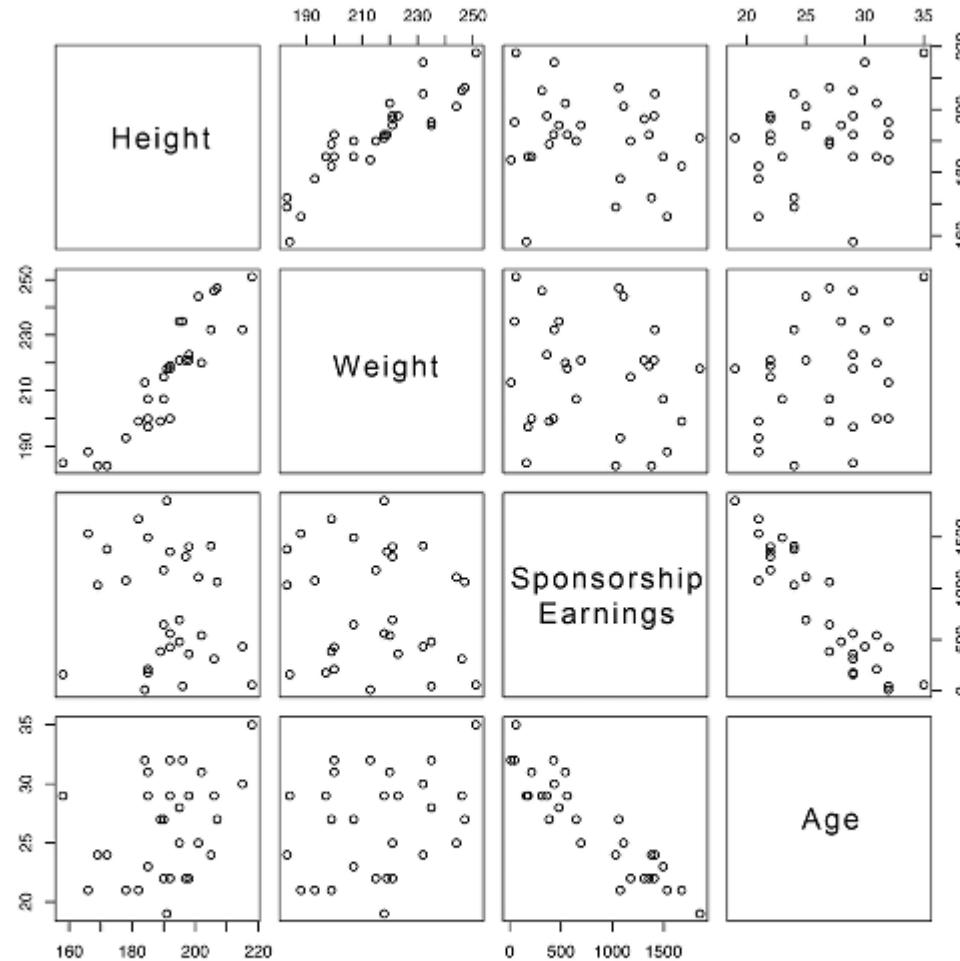
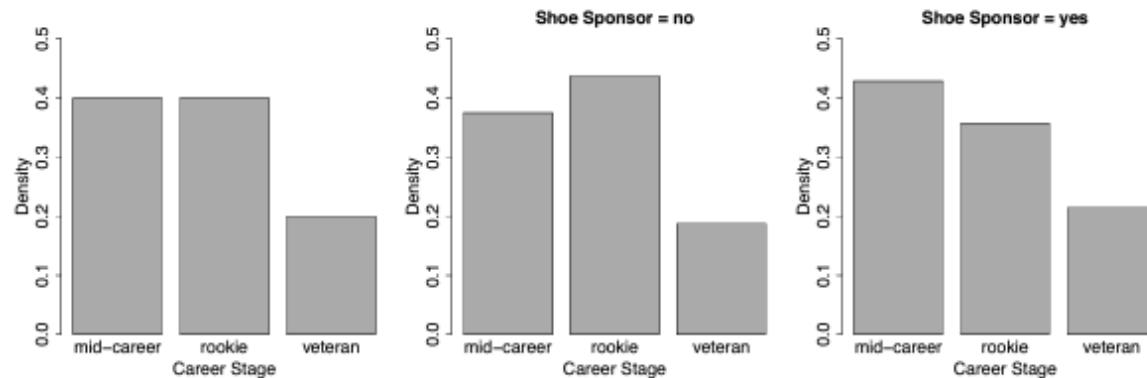


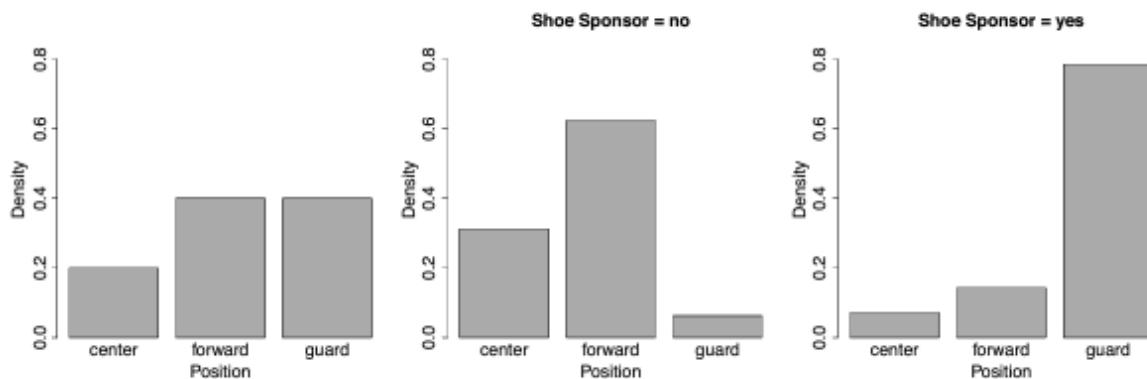
Figure 3.6

A scatter plot matrix showing scatter plots of the continuous features from the professional basketball team dataset in Table 3.7^[78].

Bar plot (categorical features)



(a) Career Stage and Shoe Sponsor



(b) Position and Shoe Sponsor

"Small multiples approach"

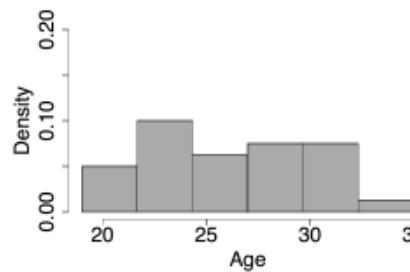
If the two features being visualised have a strong relationship, then the bar plots for each level of the second feature will look noticeably different to one another, and to the overall bar plot.

Figure 3.7

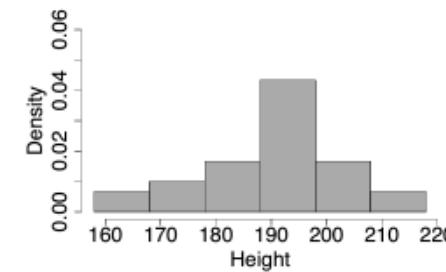
Examples of using small multiple bar plot visualizations to illustrate the relationship between two categorical features: (a) the CAREER STAGE and SHOE SPONSOR features; and (b) the POSITION and SHOE SPONSOR features. All data comes from Table 3.7^[78].

Small multiple histograms

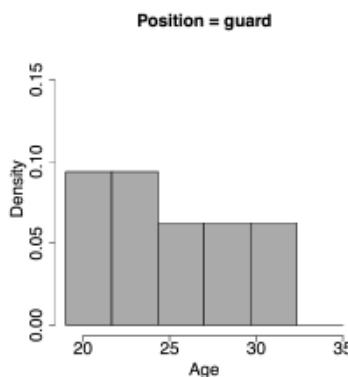
“Small multiple” histograms can be used to compare a categorical with a continuous feature



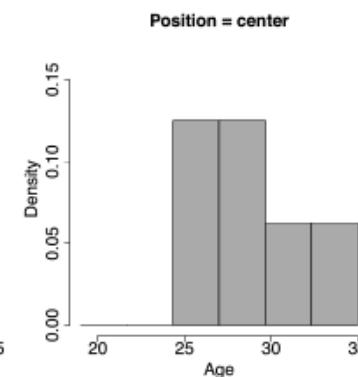
(a) Age



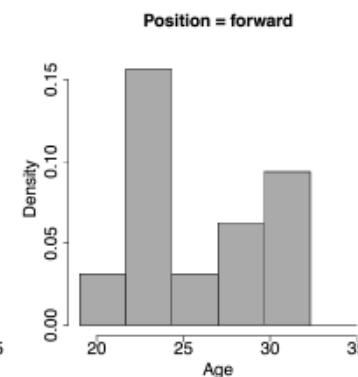
(b) Height



Position = guard



Position = center



Position = forward

(c) Age and Position

Small multiple histograms (cont.)

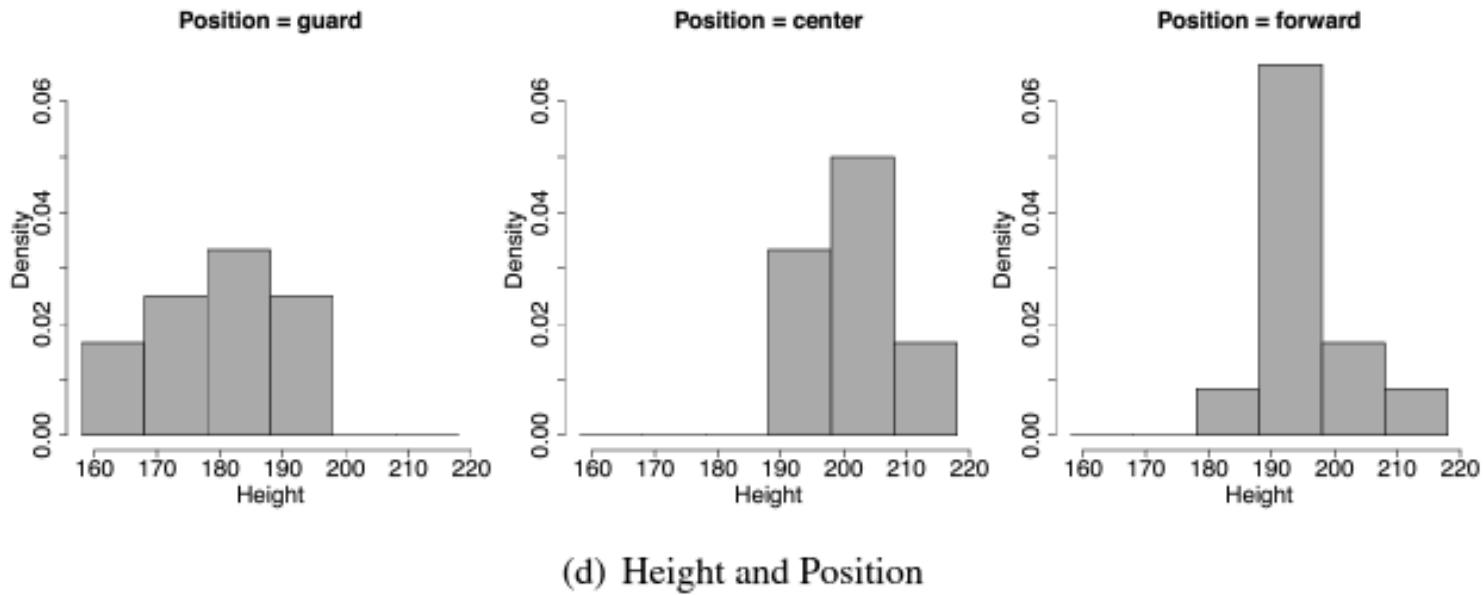


Figure 3.9

Example of using small multiple histograms to visualize the relationship between a categorical feature and a continuous feature. All examples use data from the professional basketball team dataset in Table 3.7^[78]: (a) a histogram of the AGE feature; (b) a histogram of the HEIGHT feature; (c) histograms of the AGE feature for instances displaying each level of the POSITION feature; and (d) histograms of the HEIGHT feature for instances displaying each level of the POSITION feature.

Covariance and correlation

Covariance and **correlation** provide us with formal measures of the relationship of two continuous variables

$$cov(a, b) = \frac{1}{n-1} \sum_{i=1}^n ((a_i - \bar{a}) \times (b_i - \bar{b}))$$

Covariance has a possible range of $[-\infty, \infty]$ where negative values indicate negative relationship, positive values indicate positive relationship, and values around 0 indicate little or no relationship

Correlation

Correlation is a normalized form of covariance

$$\text{corr}(a, b) = \frac{\text{cov}(a, b)}{\text{sd}(a) \times \text{sd}(b)}$$

Correlation has a range of [-1, 1]

Correlation matrix

$$\text{correlation matrix} = \begin{bmatrix} \text{corr}(a,a) & \text{corr}(a,b) & \cdots & \text{corr}(a,z) \\ \text{corr}(b,a) & \text{corr}(b,b) & \cdots & \text{corr}(b,z) \\ \vdots & \vdots & \ddots & \vdots \\ \text{corr}(z,a) & \text{corr}(z,b) & \cdots & \text{corr}(z,z) \end{bmatrix}_{\{a,b,\dots,z\}}$$

$$\text{correlation matrix} = \begin{bmatrix} 1.0 & 0.898 & 0.345 \\ 0.898 & 1.0 & 0.294 \\ 0.345 & 0.294 & 1.0 \end{bmatrix}_{\{\text{Height}, \text{Weight}, \text{Age}\}}$$

Summary (Data Quality Reports)

The key outcomes of the **data exploration** process are that the practitioner should

1. Have gotten to know the features within the ABT, especially their central tendencies, variations, and **distributions**.
2. Have identified any **data quality issues** within the ABT, in particular **missing values, irregular cardinality, and outliers**.
3. Have corrected any data quality issues due to **invalid data**.
4. Have recorded any data quality issues due to **valid data** in a **data quality plan** along with potential handling strategies.
5. Be confident that enough good quality data exists to continue with a project.

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 4: Information-based Learning
Sections 4.1, 4.2, 4.3

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

1 Big Idea

2 Fundamentals

- Decision Trees
- Shannon's Entropy Model
- Information Gain

3 Standard Approach: The ID3 Algorithm

- A Worked Example: Predicting Vegetation Distributions

4 Summary

- In this chapter we are going to introduce a machine learning algorithm that tries to build predictive models **using only the most informative features.**
- In this context an informative feature is a **descriptive feature** whose values split the instances in the dataset into **homogeneous sets** with respect to the target feature value.

Big Idea

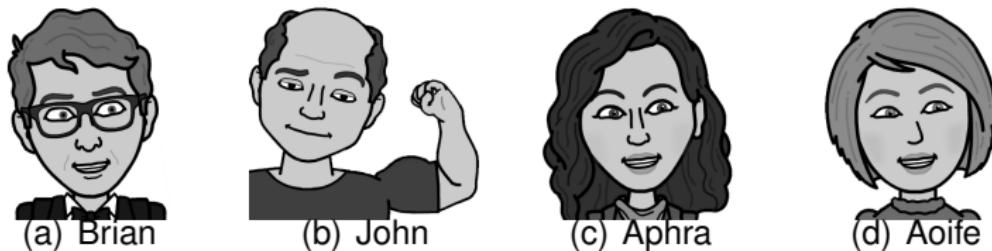


Figure: Cards showing character faces and names for the Guess-Who game

Man	Long Hair	Glasses	Name
Yes	No	Yes	Brian
Yes	No	No	John
No	Yes	No	Aphra
No	No	No	Aoife



(a) Brian



(b) John



(c) Aphra

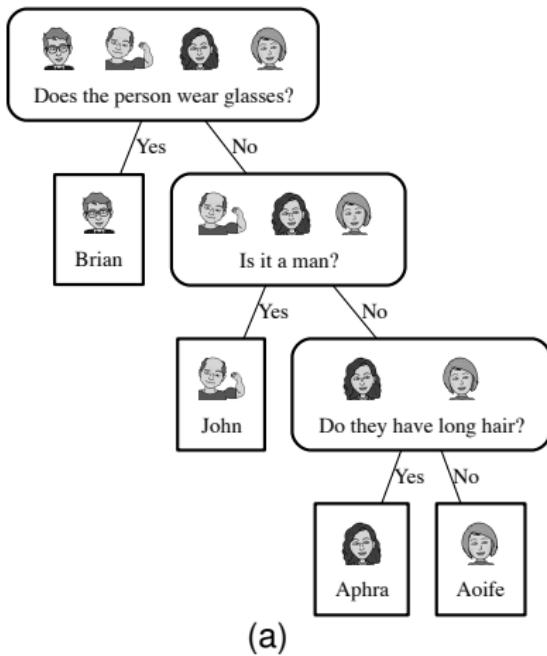


(d) Aoife

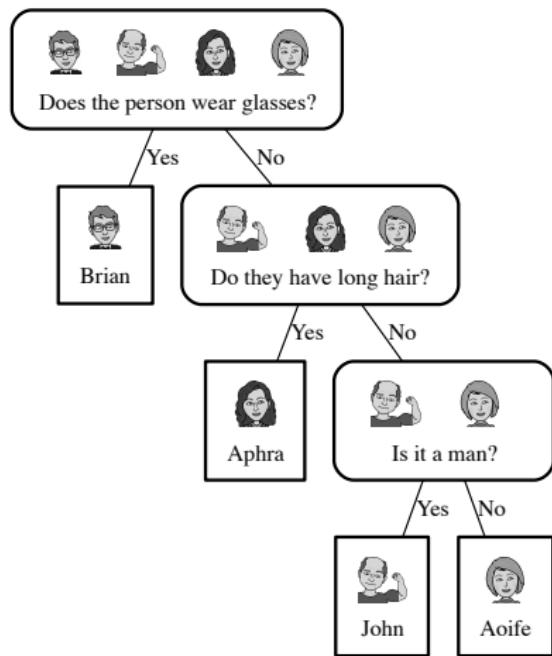
Figure: Cards showing character faces and names for the Guess-Who game

Which question would you ask first:

- ➊ Is it a man?
- ➋ Does the person wear glasses?



(a)



(b)

Figure: The different question sequences that can follow in a game of Guess Who beginning with the question Does the person wear

- In both of the diagrams:
 - one path is 1 question long,
 - one path is 2 questions long,
 - and two paths are 3 questions long.
- Consequently, if you ask Question (2) first the average number of questions you have to ask per game is:

$$\frac{1 + 2 + 3 + 3}{4} = 2.25$$

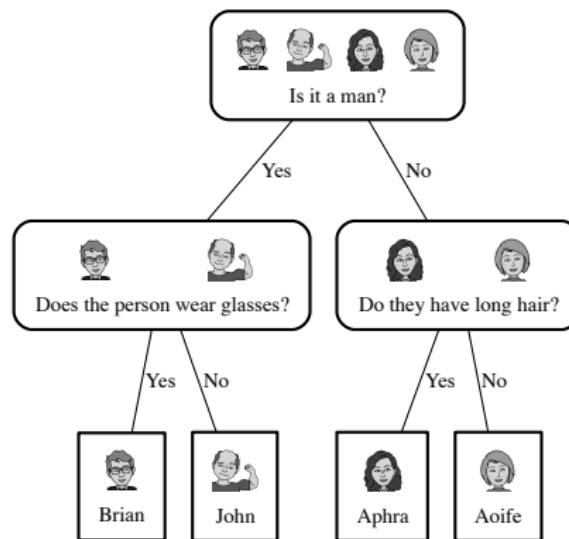


Figure: The different question sequences that can follow in a game of *Guess-Who* beginning with the question **Is it a man?**

- All the paths in this diagram are two questions long.
- So, on average if you ask Question (1) first the average number of questions you have to ask per game is:

$$\frac{2 + 2 + 2 + 2}{4} = 2$$

- On average getting an answer to Question (1) seems to give you more information than an answer to Question (2): less follow up questions.
- This is not because of the literal message of the answers: YES or NO.
- It is to do with how the answer to each question splits the domain into different sized sets based on the value of the descriptive feature the question is asked about and the likelihood of each possible answer to the question.

Big Idea

- So the big idea here is to figure out which features are the most informative ones to ask questions about by considering the effects of the different answers to the questions, in terms of:
 - 1 how the domain is split up after the answer is received,
 - 2 and the likelihood of each of the answers.

Fundamentals

- A decision tree consists of:
 - ① a **root node** (or starting node),
 - ② **interior nodes**
 - ③ and **leaf nodes** (or terminating nodes).
 - Each of the non-leaf nodes (root and interior) in the tree specifies a test to be carried out on one of the query's descriptive features.
 - Each of the leaf nodes specifies a predicted classification for the query.

Table: An email spam prediction dataset.

	SUSPICIOUS	UNKNOWN	CONTAINS	
ID	WORDS	SENDER	IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Decision Trees

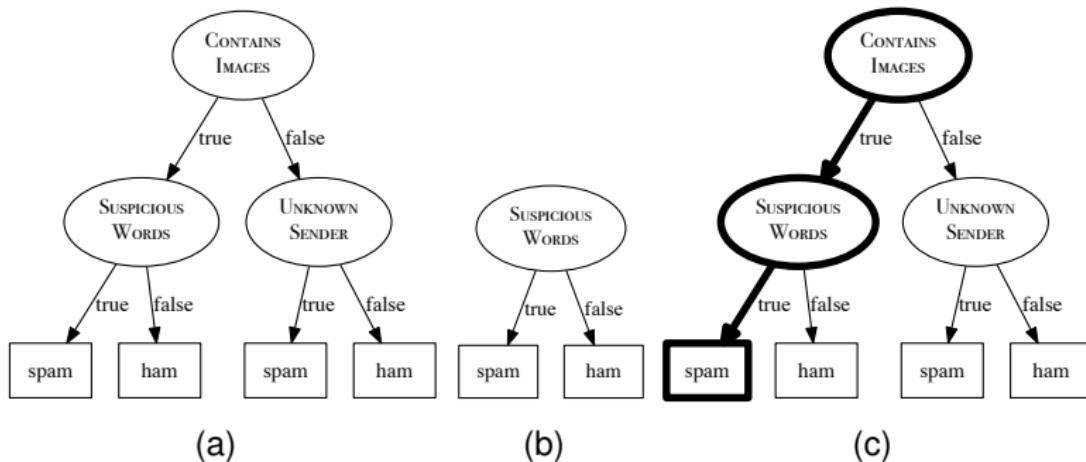


Figure: (a) and (b) show two decision trees that are consistent with the instances in the spam dataset. (c) shows the path taken through the tree shown in (a) to make a prediction for the query instance: SUSPICIOUS WORDS = 'true', UNKNOWN SENDER = 'true', CONTAINS IMAGES = 'true'.

- Both of these trees will return identical predictions for all the examples in the dataset.
 - So, which tree should we use?

- Apply the same approach as we used in the *Guess-Who* game: prefer decision trees that use less tests (shallower trees).
- This is an example of Occam's Razor.

How do we create shallow trees?

- The tree that tests SUSPICIOUS WORDS at the root is very shallow because the SUSPICIOUS WORDS feature perfectly splits the data into pure groups of '*spam*' and '*ham*'.
- Descriptive features that split the dataset into pure sets with respect to the target feature provide information about the target feature.
- So we can make shallow trees by testing the informative features early on in the tree.
- All we need to do that is a computational metric of the purity of a set: **entropy**

Shannon's Entropy Model

- Claude Shannon's entropy model defines a computational measure of the impurity of the elements of a set.
- An easy way to understand the entropy of a set is to think in terms of the uncertainty associated with guessing the result if you were to make a random selection from the set.

Shannon's Entropy Model

- Entropy is related to the probability of a outcome.
 - High probability → Low entropy
 - Low probability → High entropy
- If we take the **log** of a probability and multiply it by -1 we get this mapping!

What is a log?

Remember the *log* of a to the base b is the number to which we must raise b to get a .

- $\log_2(0.5) = -1$ because $2^{-1} = 0.5$
- $\log_2(1) = 0$ because $2^0 = 1$
- $\log_2(8) = 3$ because $2^3 = 8$
- $\log_5(25) = 2$ because $5^2 = 25$
- $\log_5(32) = 2.153$ because $5^{2.153} = 32$

Shannon's Entropy Model

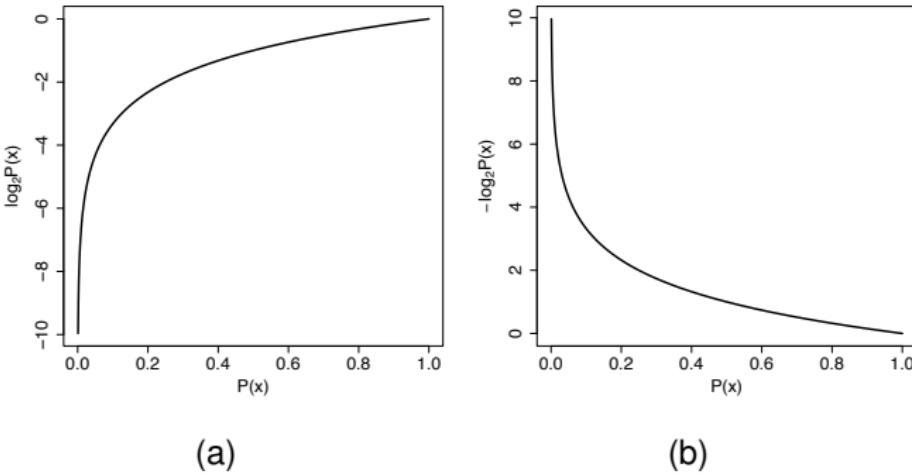


Figure: (a) A graph illustrating how the value of a binary log (the log to the base 2) of a probability changes across the range of probability values. (b) the impact of multiplying these values by -1 .

Shannon's Entropy Model

- Shannon's model of entropy is a weighted sum of the logs of the probabilities of each of the possible outcomes when we make a random selection from a set.

$$H(t) = - \sum_{i=1}^I (P(t = i) \times \log_s(P(t = i))) \quad (1)$$

Shannon's Entropy Model

- What is the entropy of a set of 52 different playing cards?

$$\begin{aligned} H(card) &= - \sum_{i=1}^{52} P(card = i) \times \log_2(P(card = i)) \\ &= - \sum_{i=1}^{52} 0.019 \times \log_2(0.019) = - \sum_{i=1}^{52} -0.1096 \\ &= 5.700 \text{ bits} \end{aligned}$$

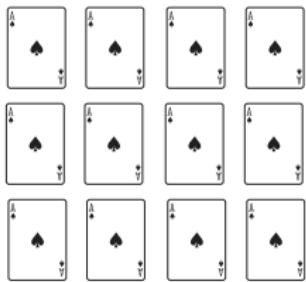
Shannon's Entropy Model

- What is the entropy of a set of 52 playing cards if we only distinguish between the cards based on their suit
 $\{\heartsuit, \clubsuit, \diamondsuit, \spadesuit\}$?

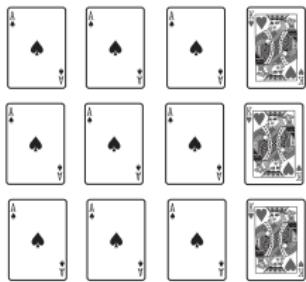
Shannon's Entropy Model

$$\begin{aligned}
 H(suit) &= - \sum_{I \in \{\heartsuit, \clubsuit, \diamondsuit, \spadesuit\}} P(suit = I) \times \log_2(P(suit = I)) \\
 &= -((P(\heartsuit) \times \log_2(P(\heartsuit))) + (P(\clubsuit) \times \log_2(P(\clubsuit))) \\
 &\quad + (P(\diamondsuit) \times \log_2(P(\diamondsuit))) + (P(\spadesuit) \times \log_2(P(\spadesuit)))) \\
 &= -((\frac{13}{52} \times \log_2(\frac{13}{52})) + (\frac{13}{52} \times \log_2(\frac{13}{52})) \\
 &\quad + (\frac{13}{52} \times \log_2(\frac{13}{52})) + (\frac{13}{52} \times \log_2(\frac{13}{52}))) \\
 &= -((0.25 \times -2) + (0.25 \times -2) \\
 &\quad + (0.25 \times -2) + (0.25 \times -2)) \\
 &= 2 \text{ bits}
 \end{aligned}$$

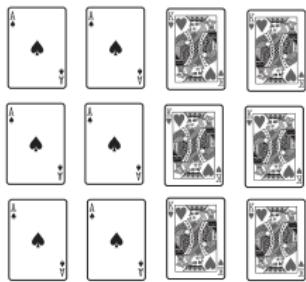
Shannon's Entropy Model



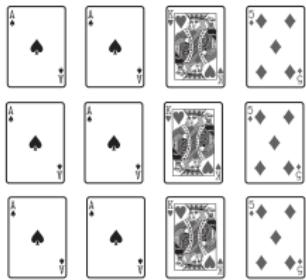
$$(a) H(card) = 0.00$$



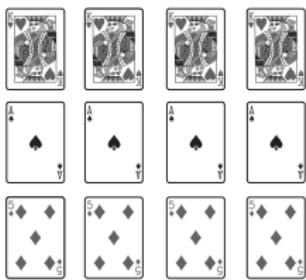
$$(b) H(card) = 0.81$$



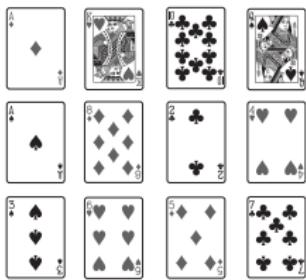
$$(c) H(card) = 1.00$$



$$(d) H(card) = 1.50$$



$$(e) H(card) = 1.58$$



$$(f) H(card) = 3.58$$

Figure: The entropy of different sets of playing cards measured in bits.

Shannon's Entropy Model

Table: The relationship between the entropy of a message and the set it was selected from.

Entropy of a Message	Properties of the Message Set
High	A large set of equally likely messages.
Medium	A large set of messages, some more likely than others.
Medium	A small set of equally likely messages.
Low	A small set of messages with one very likely message.

Information Gain

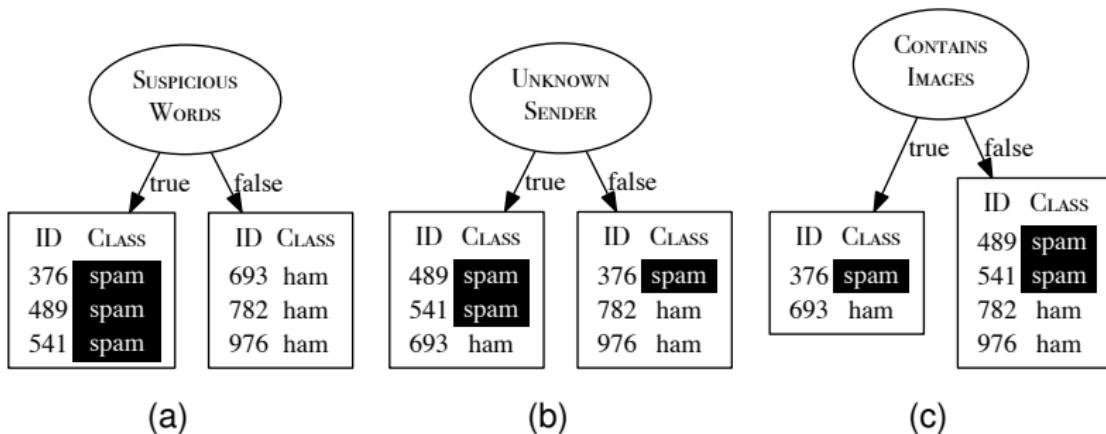


Figure: How the instances in the spam dataset split when we partition using each of the different descriptive features from the spam dataset in Table 1 [15]

Information Gain

- Our intuition is that the ideal discriminatory feature will partition the data into **pure** subsets where all the instances in each subset have the same classification.
 - SUSPICIOUS WORDS perfect split.
 - UNKNOWN SENDER mixture but some information (when 'true' most instances are 'spam').
 - CONTAINS IMAGES no information.
- One way to implement this idea is to use a metric called **information gain**.

Information Gain

Information Gain

- The information gain of a descriptive feature can be understood as a measure of the reduction in the overall entropy of a prediction task by testing on that feature.

Information Gain

Computing information gain involves the following 3 equations:

$$H(t, \mathcal{D}) = - \sum_{l \in \text{levels}(t)} (P(t = l) \times \log_2(P(t = l))) \quad (2)$$

$$\text{rem}(d, \mathcal{D}) = \sum_{l \in \text{levels}(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}} \quad (3)$$

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - \text{rem}(d, \mathcal{D}) \quad (4)$$

Information Gain

- As an example we will calculate the information gain for each of the descriptive features in the spam email dataset.

Information Gain

- Calculate the **entropy** for the target feature in the dataset.

$$H(t, \mathcal{D}) = - \sum_{l \in levels(t)} (P(t = l) \times \log_2(P(t = l)))$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Information Gain

Information Gain

- Calculate the **remainder** for the **SUSPICIOUS WORDS** feature in the dataset.

$$\text{rem}(d, \mathcal{D}) = \sum_{l \in \text{levels}(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Information Gain

 $rem(\text{WORDS}, \mathcal{D})$

$$\begin{aligned}
&= \left(\frac{|\mathcal{D}_{\text{WORDS}=T}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{WORDS}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{WORDS}=F}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{WORDS}=F}) \right) \\
&= \left(\frac{3}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\
&\quad + \left(\frac{3}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\
&= \left(\frac{3}{6} \times \left(- \left(\left(\frac{3}{3} \times \log_2(\frac{3}{3}) \right) + \left(\frac{0}{3} \times \log_2(\frac{0}{3}) \right) \right) \right) \right) \\
&\quad + \left(\frac{3}{6} \times \left(- \left(\left(\frac{0}{3} \times \log_2(\frac{0}{3}) \right) + \left(\frac{3}{3} \times \log_2(\frac{3}{3}) \right) \right) \right) \right) = 0 \text{ bits}
\end{aligned}$$

Information Gain

- Calculate the **remainder** for the UNKNOWN SENDER feature in the dataset.

$$rem(d, \mathcal{D}) = \sum_{l \in levels(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Information Gain

 $\text{rem}(\text{SENDER}, \mathcal{D})$

$$\begin{aligned}
&= \left(\frac{|\mathcal{D}_{\text{SENDER}=T}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{SENDER}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{SENDER}=F}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{SENDER}=F}) \right) \\
&= \left(\frac{3}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\
&\quad + \left(\frac{3}{6} \times \left(- \sum_{I \in \{\text{'spam'}, \text{'ham'}\}} P(t = I) \times \log_2(P(t = I)) \right) \right) \\
&= \left(\frac{3}{6} \times \left(- \left(\left(\frac{2}{3} \times \log_2(\frac{2}{3}) \right) + \left(\frac{1}{3} \times \log_2(\frac{1}{3}) \right) \right) \right) \right) \\
&\quad + \left(\frac{3}{6} \times \left(- \left(\left(\frac{1}{3} \times \log_2(\frac{1}{3}) \right) + \left(\frac{2}{3} \times \log_2(\frac{2}{3}) \right) \right) \right) \right) = 0.9183 \text{ bits}
\end{aligned}$$

Information Gain

- Calculate the **remainder** for the **CONTAINS IMAGES** feature in the dataset.

$$\text{rem}(d, \mathcal{D}) = \sum_{l \in \text{levels}(d)} \underbrace{\frac{|\mathcal{D}_{d=l}|}{|\mathcal{D}|}}_{\text{weighting}} \times \underbrace{H(t, \mathcal{D}_{d=l})}_{\substack{\text{entropy of} \\ \text{partition } \mathcal{D}_{d=l}}}$$

ID	SUSPICIOUS WORDS	UNKNOWN SENDER	CONTAINS IMAGES	CLASS
376	true	false	true	spam
489	true	true	false	spam
541	true	true	false	spam
693	false	true	true	ham
782	false	false	false	ham
976	false	false	false	ham

Information Gain

$$\begin{aligned}
& rem(\text{IMAGES}, \mathcal{D}) \\
&= \left(\frac{|\mathcal{D}_{\text{IMAGES}=T}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{IMAGES}=T}) \right) + \left(\frac{|\mathcal{D}_{\text{IMAGES}=F}|}{|\mathcal{D}|} \times H(t, \mathcal{D}_{\text{IMAGES}=F}) \right) \\
&= \left(\frac{2}{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t = l) \times \log_2(P(t = l)) \right) \right) \\
&\quad + \left(\frac{4}{6} \times \left(- \sum_{l \in \{\text{'spam'}, \text{'ham'}\}} P(t = l) \times \log_2(P(t = l)) \right) \right) \\
&= \left(\frac{2}{6} \times \left(- \left(\left(\frac{1}{2} \times \log_2(\frac{1}{2}) \right) + \left(\frac{1}{2} \times \log_2(\frac{1}{2}) \right) \right) \right) \right) \\
&\quad + \left(\frac{4}{6} \times \left(- \left(\left(\frac{2}{4} \times \log_2(\frac{2}{4}) \right) + \left(\frac{2}{4} \times \log_2(\frac{2}{4}) \right) \right) \right) \right) = 1 \text{ bit}
\end{aligned}$$

Information Gain

- Calculate the **information gain** for the three descriptive feature in the dataset.

$$IG(d, \mathcal{D}) = H(t, \mathcal{D}) - rem(d, \mathcal{D})$$

Information Gain

$$\begin{aligned}IG(\text{SUSPICIOUS WORDS}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - \text{rem}(\text{SUSPICIOUS WORDS}, \mathcal{D}) \\&= 1 - 0 = 1 \text{ bit}\end{aligned}$$

$$\begin{aligned}IG(\text{UNKNOWN SENDER}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - \text{rem}(\text{UNKNOWN SENDER}, \mathcal{D}) \\&= 1 - 0.9183 = 0.0817 \text{ bits}\end{aligned}$$

$$\begin{aligned}IG(\text{CONTAINS IMAGES}, \mathcal{D}) &= H(\text{CLASS}, \mathcal{D}) - \text{rem}(\text{CONTAINS IMAGES}, \mathcal{D}) \\&= 1 - 1 = 0 \text{ bits}\end{aligned}$$

- The results of these calculations match our intuitions.

Standard Approach: The ID3 Algorithm

- ID3 Algorithm (Iterative Dichotomizer 3)
- Attempts to create the shallowest tree that is consistent with the data that it is given.
- The ID3 algorithm builds the tree in a recursive, depth-first manner, beginning at the root node and working down to the leaf nodes.

- ➊ The algorithm begins by choosing the best descriptive feature to test (i.e., the best question to ask first) using **information gain**.
- ➋ A root node is then added to the tree and labelled with the selected test feature.
- ➌ The training dataset is then partitioned using the test.
- ➍ For each partition a branch is grown from the node.
- ➎ The process is then repeated for each of these branches using the relevant partition of the training set in place of the full training set and with the selected test feature excluded from further testing.

The algorithm defines three situations where the recursion stops and a leaf node is constructed:

- ① All of the instances in the dataset have the same classification (target feature value) then return a leaf node tree with that classification as its label.
- ② The set of features left to test is empty then return a leaf node tree with the majority class of the dataset as its classification.
- ③ The dataset is empty return a leaf node tree with the majority class of the dataset at the parent node that made the recursive call.

A Worked Example: Predicting Vegetation Distributions

Table: The vegetation classification dataset.

ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	high	chaparral
2	true	moderate	low	riparian
3	true	steep	medium	riparian
4	false	steep	medium	chaparral
5	false	flat	high	conifer
6	true	steep	highest	conifer
7	true	steep	high	chaparral

A Worked Example: Predicting Vegetation Distributions

$$H(\text{VEGETATION}, \mathcal{D})$$

$$\begin{aligned} &= - \sum_{I \in \{ \text{'chaparral'}, \text{'riparian'}, \text{'conifer'} \}} P(\text{VEGETATION} = I) \times \log_2 (P(\text{VEGETATION} = I)) \\ &= - \left(\left(\frac{3}{7} \times \log_2 \left(\frac{3}{7} \right) \right) + \left(\frac{2}{7} \times \log_2 \left(\frac{2}{7} \right) \right) + \left(\frac{2}{7} \times \log_2 \left(\frac{2}{7} \right) \right) \right) \\ &= 1.5567 \text{ bits} \end{aligned}$$

A Worked Example: Predicting Vegetation Distributions

Table: Partition sets (Part.), entropy, remainder (Rem.) and information gain (Info. Gain) by feature for the dataset in Table 3 [49].

Split By Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
STREAM	'true'	\mathcal{D}_1	d_2, d_3, d_6, d_7	1.5	1.2507	0.3060
	'false'	\mathcal{D}_2	d_1, d_4, d_5	0.9183		
SLOPE	'flat'	\mathcal{D}_3	d_5	0	0.9793	0.5774
	'moderate'	\mathcal{D}_4	d_2	0		
	'steep'	\mathcal{D}_5	d_1, d_3, d_4, d_6, d_7	1.3710		
ELEVATION	'low'	\mathcal{D}_6	d_2	0	0.6793	0.8774
	'medium'	\mathcal{D}_7	d_3, d_4	1.0		
	'high'	\mathcal{D}_8	d_1, d_5, d_7	0.9183		
	'highest'	\mathcal{D}_9	d_6	0		

A Worked Example: Predicting Vegetation Distributions

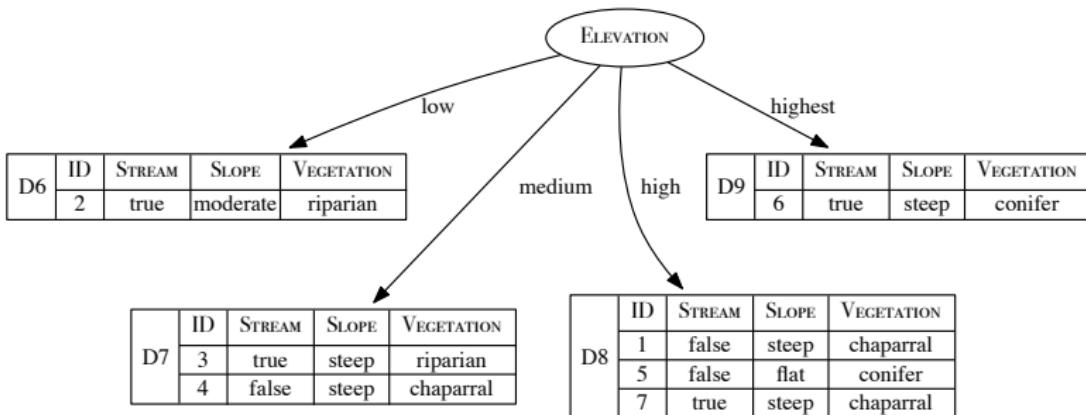


Figure: The decision tree after the data has been split using ELEVATION.

A Worked Example: Predicting Vegetation Distributions

$$H(V_{\text{EGETATION}}, \mathcal{D}_7)$$

$$\begin{aligned} &= - \sum_{I \in \{\text{'chaparral'}, \text{'riparian'}, \text{'conifer'}\}} P(V_{\text{EGETATION}} = I) \times \log_2 (P(V_{\text{EGETATION}} = I)) \\ &= - \left(\left(\frac{1}{2} \times \log_2(\frac{1}{2}) \right) + \left(\frac{1}{2} \times \log_2(\frac{1}{2}) \right) + \left(\frac{0}{2} \times \log_2(\frac{0}{2}) \right) \right) \\ &= 1.0 \text{ bits} \end{aligned}$$

A Worked Example: Predicting Vegetation Distributions

Table: Partition sets (Part.), entropy, remainder (Rem.) and information gain (Info. Gain) by feature for the dataset \mathcal{D}_7 in Figure 9 [52].

Split By Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
STREAM	'true'	\mathcal{D}_{10}	\mathbf{d}_3	0	0	1.0
	'false'	\mathcal{D}_{11}	\mathbf{d}_4	0		
SLOPE	'flat'	\mathcal{D}_{12}		0	1.0	0
	'moderate'	\mathcal{D}_{13}		0		
	'steep'	\mathcal{D}_{14}	$\mathbf{d}_3, \mathbf{d}_4$	1.0		

A Worked Example: Predicting Vegetation Distributions

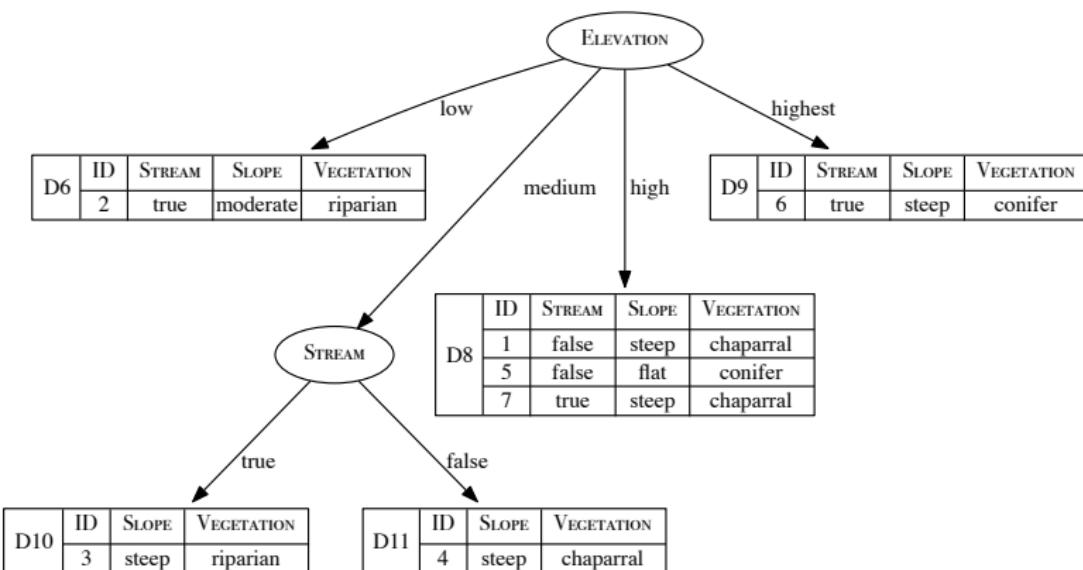


Figure: The state of the decision tree after the \mathcal{D}_7 partition has been split using STREAM.

A Worked Example: Predicting Vegetation Distributions

$$H(\text{VEGETATION}, \mathcal{D}_8)$$

$$\begin{aligned} &= - \sum_{I \in \{\text{'chaparral'}, \text{'riparian'}, \text{'conifer'}\}} P(\text{VEGETATION} = I) \times \log_2 (P(\text{VEGETATION} = I)) \\ &= - \left(\left(\frac{2}{3} \times \log_2 \left(\frac{2}{3} \right) \right) + \left(\frac{0}{3} \times \log_2 \left(\frac{0}{3} \right) \right) + \left(\frac{1}{3} \times \log_2 \left(\frac{1}{3} \right) \right) \right) \\ &= 0.9183 \text{ bits} \end{aligned}$$

A Worked Example: Predicting Vegetation Distributions

Table: Partition sets (Part.), entropy, remainder (Rem.) and information gain (Info. Gain) by feature for the dataset \mathcal{D}_8 in Figure 10 [55].

Split By Feature	Level	Part.	Instances	Partition Entropy	Rem.	Info. Gain
STREAM	'true'	\mathcal{D}_{15}	d_7	0	0.6666	0.2517
	'false'	\mathcal{D}_{16}	d_1, d_5	1.0		
SLOPE	'flat'	\mathcal{D}_{17}	d_5	0	0	0.9183
	'moderate'	\mathcal{D}_{18}		0		
	'steep'	\mathcal{D}_{19}	d_1, d_7	0		

A Worked Example: Predicting Vegetation Distributions

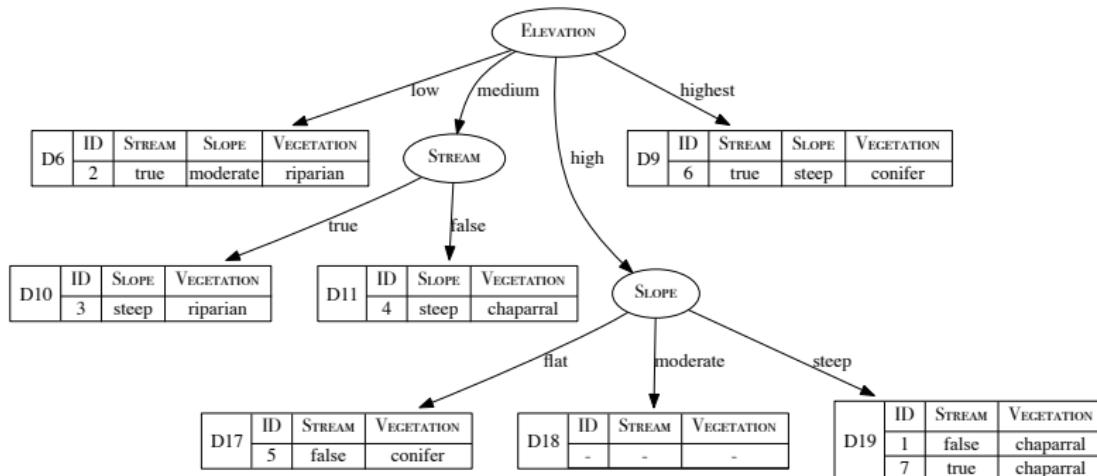


Figure: The state of the decision tree after the \mathcal{D}_8 partition has been split using SLOPE.

A Worked Example: Predicting Vegetation Distributions

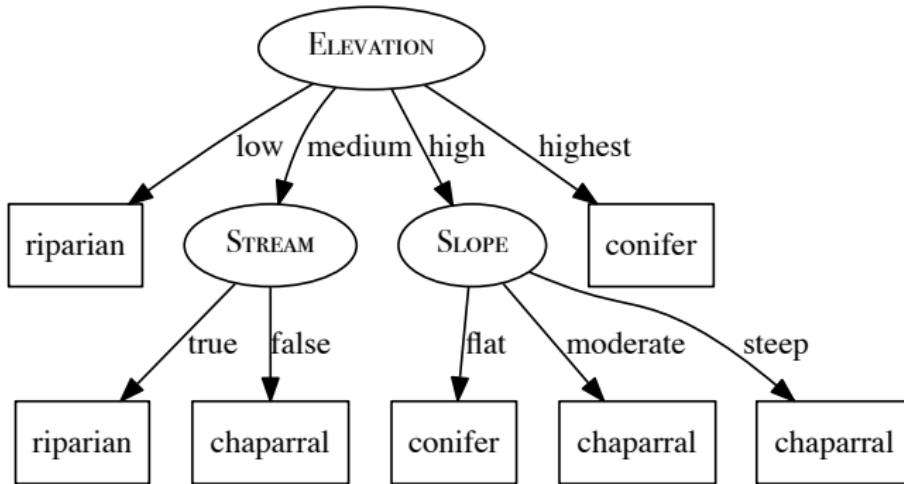
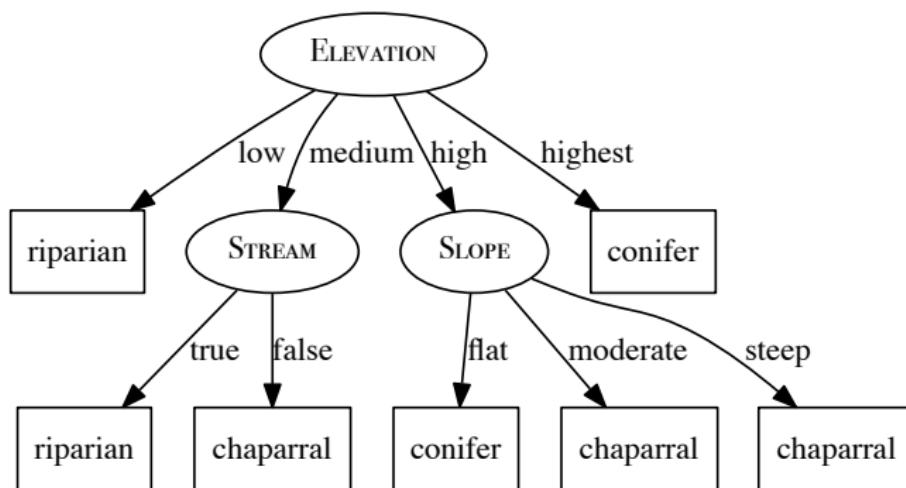


Figure: The final vegetation classification decision tree.

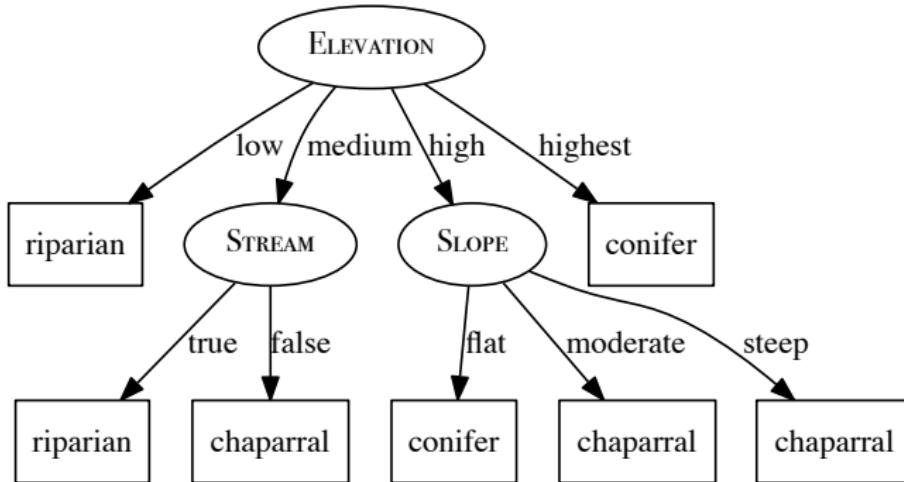
A Worked Example: Predicting Vegetation Distributions



- What prediction will this decision tree model return for the following query?

STREAM = 'true', SLOPE='Moderate', ELEVATION='High'

A Worked Example: Predicting Vegetation Distributions



- What prediction will this decision tree model return for the following query?

STREAM = 'true', SLOPE='Moderate', ELEVATION='High'

VEGETATION = '*Chaparral*'

A Worked Example: Predicting Vegetation Distributions

ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	high	chaparral
2	true	moderate	low	riparian
3	true	steep	medium	riparian
4	false	steep	medium	chaparral
5	false	flat	high	conifer
6	true	steep	highest	conifer
7	true	steep	high	chaparral

STREAM = '*true*', SLOPE= '*Moderate*', ELEVATION= '*High*'

VEGETATION = '*Chaparral*'

- This is an example where the model is attempting to **generalize** beyond the dataset.
- Whether or not the generalization is correct depends on whether the assumptions used in generating the model (i.e. the **inductive bias**) were appropriate.

Summary

- The ID3 algorithm works the same way for larger more complicated datasets.
- There have been many extensions and variations proposed for the ID3 algorithm:
 - using different impurity measures (Gini, Gain Ratio)
 - handling continuous descriptive features
 - to handle continuous targets
 - pruning to guard against over-fitting
 - using decision trees as part of an ensemble (Random Forests)
- We cover these extensions in Section 4.4.

1 Big Idea

2 Fundamentals

- Decision Trees
- Shannon's Entropy Model
- Information Gain

3 Standard Approach: The ID3 Algorithm

- A Worked Example: Predicting Vegetation Distributions

4 Summary

Fundamentals of Machine Learning for Predictive Data Analytics

**Chapter 4: Information-based Learning
Sections 4.4, 4.5**

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

- 1 Alternative Feature Selection Metrics
- 2 Handling Continuous Descriptive Features
- 3 Predicting Continuous Targets
- 4 Noisy Data, Overfitting and Tree Pruning
- 5 Model Ensembles
 - Boosting
 - Bagging
- 6 Summary

Alternative Feature Selection Metrics

- Entropy based information gain, prefers features with many values.
- One way of addressing this issue is to use **information gain ratio** which is computed by dividing the information gain of a feature by the amount of information used to determine the value of the feature:

$$GR(d, \mathcal{D}) = \frac{IG(d, \mathcal{D})}{-\sum_{l \in levels(d)} (P(d = l) \times \log_2(P(d = l)))} \quad (1)$$

$$\begin{aligned}IG(\text{STREAM}, \mathcal{D}) &= 0.3060 \\IG(\text{SLOPE}, \mathcal{D}) &= 0.5774 \\IG(\text{ELEVATION}, \mathcal{D}) &= 0.8774\end{aligned}$$

$$H(\text{STREAM}, \mathcal{D})$$

$$\begin{aligned} &= - \sum_{l \in \{\text{'true'}, \text{'false'}\}} P(\text{STREAM} = l) \times \log_2 (P(\text{STREAM} = l)) \\ &= - \left(\left(\frac{4}{7} \times \log_2 \left(\frac{4}{7} \right) \right) + \left(\frac{3}{7} \times \log_2 \left(\frac{3}{7} \right) \right) \right) \\ &= 0.9852 \text{ bits} \end{aligned}$$

$$H(\text{SLOPE}, \mathcal{D})$$

$$\begin{aligned} &= - \sum_{l \in \{\text{'flat'}, \text{'moderate'}, \text{'steep'}\}} P(\text{SLOPE} = l) \times \log_2 (P(\text{SLOPE} = l)) \\ &= - \left(\left(\frac{1}{7} \times \log_2 \left(\frac{1}{7} \right) \right) + \left(\frac{1}{7} \times \log_2 \left(\frac{1}{7} \right) \right) + \left(\frac{5}{7} \times \log_2 \left(\frac{5}{7} \right) \right) \right) \\ &= 1.1488 \text{ bits} \end{aligned}$$

$$H(\text{ELEVATION}, \mathcal{D})$$

$$\begin{aligned} &= - \sum_{l \in \{\text{'low'}, \text{'medium'}, \text{'high'}, \text{'highest'}\}} P(\text{ELEVATION} = l) \times \log_2 (P(\text{ELEVATION} = l)) \\ &= - \left(\left(\frac{1}{7} \times \log_2 \left(\frac{1}{7} \right) \right) + \left(\frac{2}{7} \times \log_2 \left(\frac{2}{7} \right) \right) + \left(\frac{3}{7} \times \log_2 \left(\frac{3}{7} \right) \right) + \left(\frac{1}{7} \times \log_2 \left(\frac{1}{7} \right) \right) \right) \\ &= 1.8424 \text{ bits} \end{aligned}$$

$$GR(\text{STREAM}, \mathcal{D}) = \frac{0.3060}{0.9852} = 0.3106$$

$$GR(\text{SLOPE}, \mathcal{D}) = \frac{0.5774}{1.1488} = 0.5026$$

$$GR(\text{ELEVATION}, \mathcal{D}) = \frac{0.8774}{1.8424} = 0.4762$$

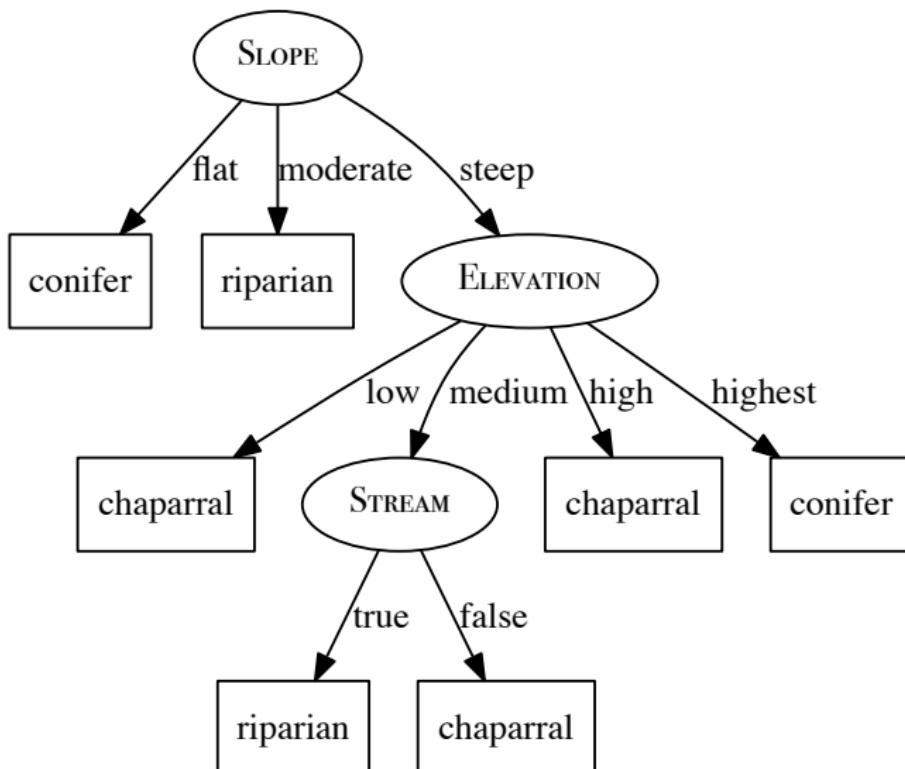


Figure: The vegetation classification decision tree generated using information gain ratio.

- Another commonly used measure of impurity is the **Gini index**:

$$Gini(t, \mathcal{D}) = 1 - \sum_{l \in levels(t)} P(t = l)^2 \quad (2)$$

- The Gini index can be thought of as calculating how often you would misclassify an instance in the dataset if you classified it based on the distribution of classifications in the dataset.
- Information gain can be calculated using the Gini index by replacing the entropy measure with the Gini index.

$Gini(V_{E}G_{E}TATI_{O}, \mathcal{D})$

$$\begin{aligned} &= 1 - \sum_{l \in \{ 'chapparal', 'riparian', 'conifer' \}} P(V_{E}G_{E}TATI_{O} = l)^2 \\ &= 1 - \left((3/7)^2 + (2/7)^2 + (2/7)^2 \right) \\ &= 0.6531 \end{aligned}$$

Table: Partition sets (Part.), entropy, Gini index, remainder (Rem.), and information gain (Info. Gain) by feature

Split by Feature	Level	Part.	Instances	Partition Gini Index	Rem.	Info. Gain
STREAM	'true'	\mathcal{D}_1	$\mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_6, \mathbf{d}_7$	0.625	0.5476	0.1054
	'false'	\mathcal{D}_2	$\mathbf{d}_1, \mathbf{d}_4, \mathbf{d}_5$	0.4444		
SLOPE	'flat'	\mathcal{D}_3	\mathbf{d}_5	0	0.4	0.2531
	'moderate'	\mathcal{D}_4	\mathbf{d}_2	0		
	'steep'	\mathcal{D}_5	$\mathbf{d}_1, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_6, \mathbf{d}_7$	0.56		
ELEVATION	'low'	\mathcal{D}_6	\mathbf{d}_2	0	0.3333	0.3198
	'medium'	\mathcal{D}_7	$\mathbf{d}_3, \mathbf{d}_4$	0.5		
	'high'	\mathcal{D}_8	$\mathbf{d}_1, \mathbf{d}_5, \mathbf{d}_7$	0.4444		
	'highest'	\mathcal{D}_9	\mathbf{d}_6	0		

Handling Continuous Descriptive Features

- The easiest way to handle continuous valued descriptive features is to turn them into boolean features by defining a threshold and using this threshold to partition the instances based their value of the continuous descriptive feature.
- How do we set the threshold?

- ➊ The instances in the dataset are sorted according to the continuous feature values.
- ➋ The adjacent instances in the ordering that have different classifications are then selected as possible threshold points.
- ➌ The optimal threshold is found by computing the information gain for each of these classification transition boundaries and selecting the boundary with the highest information gain as the threshold.

- Once a threshold has been set the dynamically created new boolean feature can compete with the other categorical features for selection as the splitting feature at that node.
- This process can be repeated at each node as the tree grows.

Table: Dataset for predicting the vegetation in an area with a continuous ELEVATION feature (measured in feet).

ID	STREAM	SLOPE	ELEVATION	VEGETATION
1	false	steep	3 900	chapparal
2	true	moderate	300	riparian
3	true	steep	1 500	riparian
4	false	steep	1 200	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer
7	true	steep	3 000	chapparal

Table: Dataset for predicting the vegetation in an area sorted by the continuous ELEVATION feature.

ID	STREAM	SLOPE	ELEVATION	VEGETATION
2	true	moderate	300	riparian
4	false	steep	1 200	chapparal
3	true	steep	1 500	riparian
7	true	steep	3 000	chapparal
1	false	steep	3 900	chapparal
5	false	flat	4 450	conifer
6	true	steep	5 000	conifer

Table: Partition sets (Part.), entropy, remainder (Rem.), and information gain (Info. Gain) for the candidate ELEVATION thresholds: ≥ 750 , $\geq 1\,350$, $\geq 2\,250$ and $\geq 4\,175$.

Split by Threshold	Part.	Instances	Partition Entropy	Rem.	Info. Gain
≥ 750	\mathcal{D}_1	d_2	0.0	1.2507	0.3060
	\mathcal{D}_2	$d_4, d_3, d_7, d_1, d_5, d_6$	1.4591		
$\geq 1\,350$	\mathcal{D}_3	d_2, d_4	1.0	1.3728	0.1839
	\mathcal{D}_4	d_3, d_7, d_1, d_5, d_6	1.5219		
$\geq 2\,250$	\mathcal{D}_5	d_2, d_4, d_3	0.9183	0.9650	0.5917
	\mathcal{D}_6	d_7, d_1, d_5, d_6	1.0		
$\geq 4\,175$	\mathcal{D}_7	d_2, d_4, d_3, d_7, d_1	0.9710	0.6935	0.8631
	\mathcal{D}_8	d_5, d_6	0.0		

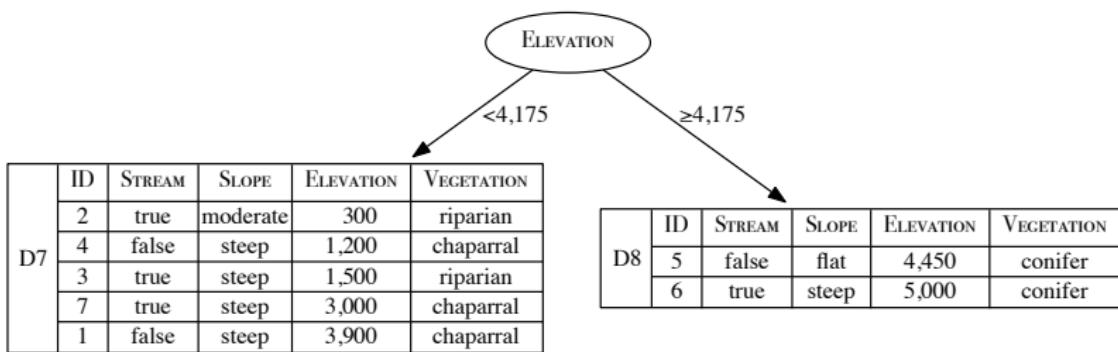


Figure: The vegetation classification decision tree after the dataset has been split using $ELEVATION \geq 4,175$.

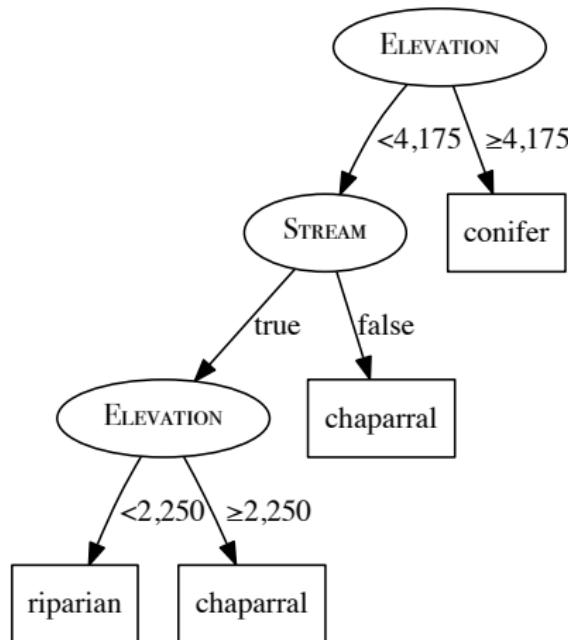


Figure: The decision tree that would be generated for the vegetation classification dataset listed in Table 3 [17] using information gain.

Predicting Continuous Targets

- Regression trees are constructed so as to reduce the **variance** in the set of training examples at each of the leaf nodes in the tree
- We can do this by adapting the ID3 algorithm to use a measure of variance rather than a measure of classification impurity (entropy) when selecting the best attribute

- The impurity (variance) at a node can be calculated using the following equation:

$$\text{var}(t, \mathcal{D}) = \frac{\sum_{i=1}^n (t_i - \bar{t})^2}{n - 1} \quad (3)$$

- We select the feature to split on at a node by selecting the feature that minimizes the weighted variance across the resulting partitions:

$$\mathbf{d}[best] = \operatorname{argmin}_{d \in \mathbf{d}} \sum_{I \in levels(d)} \frac{|\mathcal{D}_{d=I}|}{|\mathcal{D}|} \times \text{var}(t, \mathcal{D}_{d=I}) \quad (4)$$

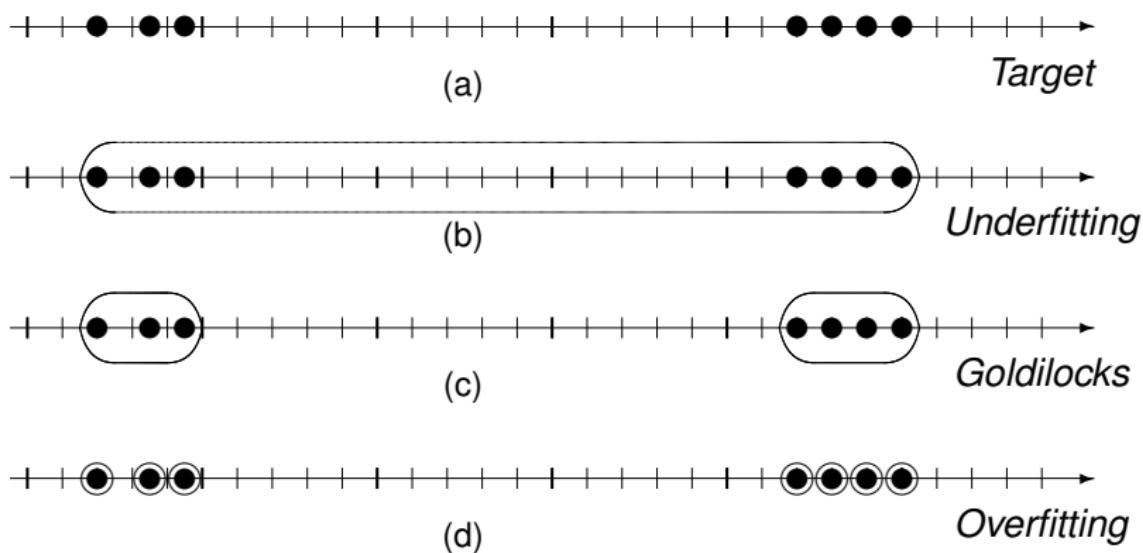


Figure: (a) A set of instances on a continuous number line; (b), (c), and (d) depict some of the potential groupings that could be applied to these instances.

Table: A dataset listing the number of bike rentals per day.

ID	SEASON	WORK DAY	RENTALS	ID	SEASON	WORK DAY	RENTALS
1	winter	false	800	7	summer	false	3 000
2	winter	false	826	8	summer	true	5 800
3	winter	true	900	9	summer	true	6 200
4	spring	false	2 100	10	autumn	false	2 910
5	spring	true	4 740	11	autumn	false	2 880
6	spring	true	4 900	12	autumn	true	2 820

Table: The partitioning of the dataset in Table 5 [25] based on SEASON and WORK DAY features and the computation of the weighted variance for each partitioning.

Split by Feature	Level	Part.	Instances	$\frac{ \mathcal{D}_{d=I} }{ \mathcal{D} }$	$\text{var}(t, \mathcal{D})$	Weighted Variance
SEASON	'winter'	\mathcal{D}_1	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3$	0.25	2 692	
	'spring'	\mathcal{D}_2	$\mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6$	0.25	2 472 533 $\frac{1}{3}$	
	'summer'	\mathcal{D}_3	$\mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_9$	0.25	3 040 000	1 379 331 $\frac{1}{3}$
	'autumn'	\mathcal{D}_4	$\mathbf{d}_{10}, \mathbf{d}_{11}, \mathbf{d}_{12}$	0.25	2 100	
WORK DAY	'true'	\mathcal{D}_5	$\mathbf{d}_3, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_8, \mathbf{d}_9, \mathbf{d}_{12}$	0.50	4 026 346 $\frac{1}{3}$	
	'false'	\mathcal{D}_6	$\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_7, \mathbf{d}_{10}, \mathbf{d}_{11}$	0.50	1 077 280	2 551 813 $\frac{1}{3}$

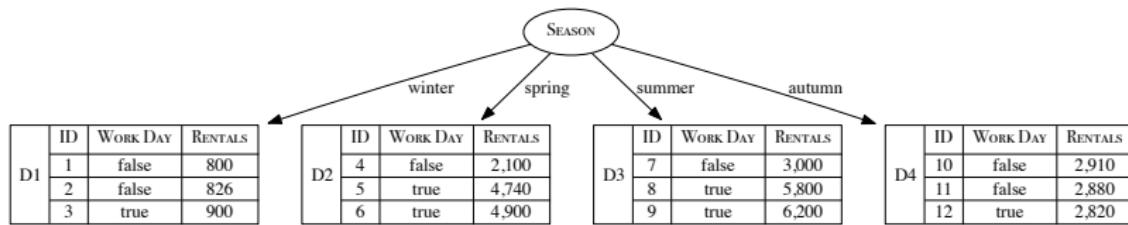


Figure: The decision tree resulting from splitting the data in Table 5 [25] using the feature SEASON.

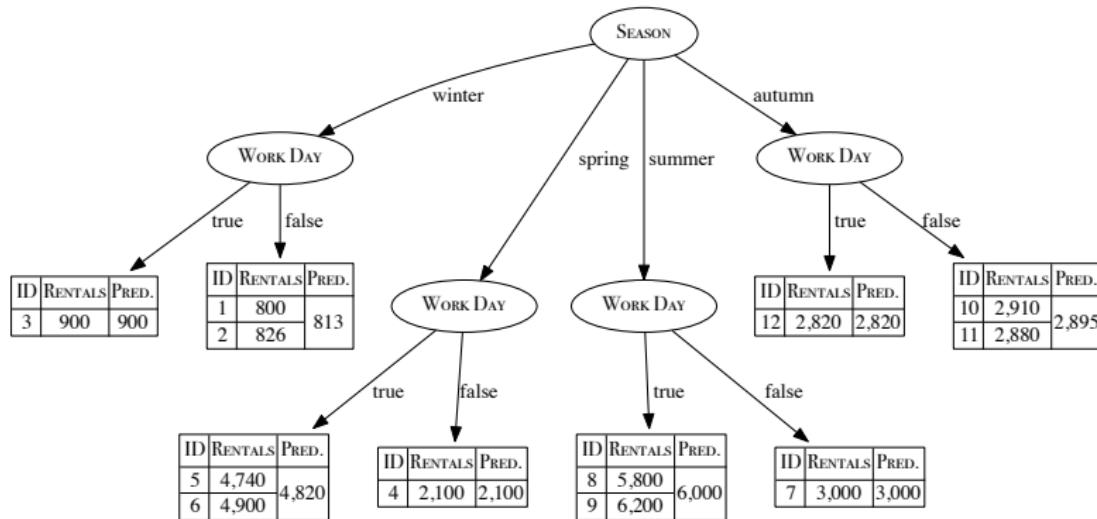


Figure: The final decision tree induced from the dataset in Table 5 [25]. To illustrate how the tree generates predictions, this tree lists the instances that ended up at each leaf node and the prediction (PRED.) made by each leaf node.

Noisy Data, Overfitting and Tree Pruning

- In the case of a decision tree, over-fitting involves splitting the data on an irrelevant feature.

The likelihood of over-fitting occurring increases as a tree gets deeper because the resulting classifications are based on smaller and smaller subsets as the dataset is partitioned after each feature test in the path.

- **Pre-pruning**: stop the recursive partitioning early.
Pre-pruning is also known as **forward pruning**.

Common Pre-pruning Approaches

- 1 **early stopping**
- 2 χ^2 **pruning**

- **Post-pruning**: allow the algorithm to grow the tree as much as it likes and then prune the tree of the branches that cause over-fitting.

Common Post-pruning Approach

- Using the validation set evaluate the prediction accuracy achieved by both the fully grown tree and the pruned copy of the tree. If the pruned copy of the tree performs no worse than the fully grown tree the node is a candidate for pruning.

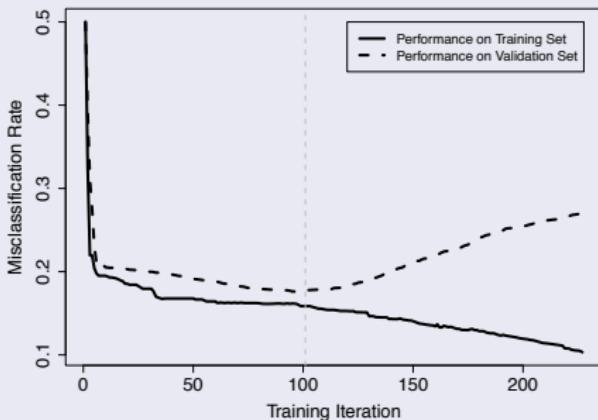


Table: An example validation set for the post-operative patient routing task.

ID	CORE-TEMP	STABLE-TEMP	GENDER	DECISION
1	high	true	male	gen
2	low	true	female	icu
3	high	false	female	icu
4	high	false	male	icu
5	low	false	female	icu
6	low	true	male	icu

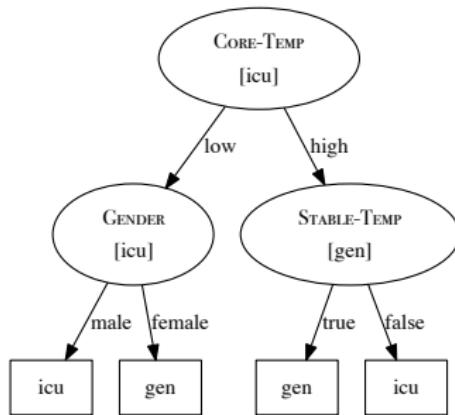


Figure: The decision tree for the post-operative patient routing task.

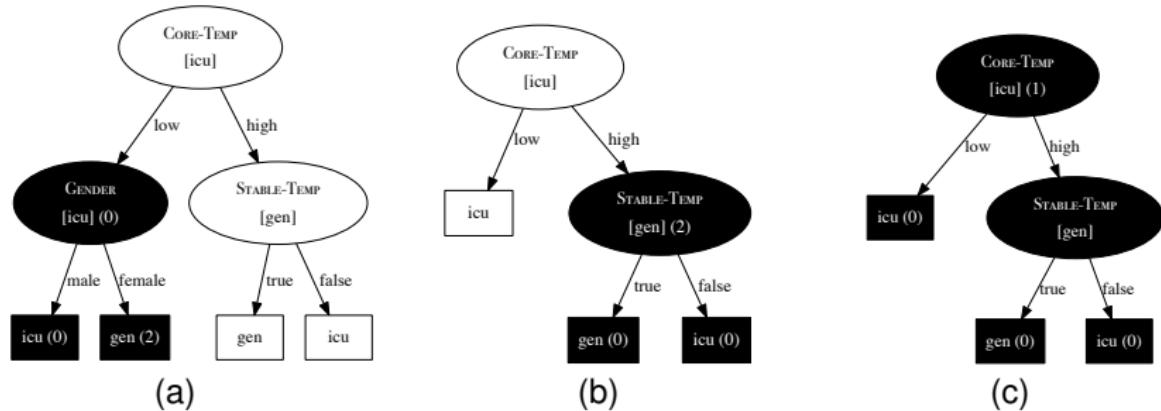


Figure: The iterations of reduced error pruning for the decision tree in Figure 7^[34] using the validation set in Table 7^[33]. The subtree that is being considered for pruning in each iteration is highlighted in black. The prediction returned by each non-leaf node is listed in square brackets. The error rate for each node is given in round brackets.

Advantages of pruning:

- Smaller trees are easier to interpret
- Increased generalization accuracy when there is noise in the training data (**noise dampening**).

Model Ensembles

- Rather than creating a single model they generate a set of models and then make predictions by aggregating the outputs of these models.
- A prediction model that is composed of a set of models is called a **model ensemble**.
- In order for this approach to work the models that are in the ensemble must be different from each other.

- There are two standard approaches to creating ensembles:
 - ➊ **boosting**
 - ➋ **bagging**.

Boosting

- Boosting works by iteratively creating models and adding them to the ensemble.
- The iteration stops when a predefined number of models have been added.
- When we use **boosting** each new model added to the ensemble is biased to pay more attention to instances that previous models miss-classified.
- This is done by incrementally adapting the dataset used to train the models. To do this we use a **weighted dataset**

Weighted Dataset

- Each instance has an associated weight $\mathbf{w}_i \geq 0$,
- Initially set to $\frac{1}{n}$ where n is the number of instances in the dataset.
- After each model is added to the ensemble it is tested on the **training data** and the weights of the instances the model gets correct are decreased and the weights of the instances the model gets incorrect are increased.
- These weights are used as a distribution over which the dataset is sampled to create a **replicated training set**, where the replication of an instance is proportional to its weight.

During each **training iteration** the algorithm:

- ➊ Induces a model and calculates the total error, ϵ , by summing the weights of the training instances for which the predictions made by the model are incorrect.
- ➋ Increases the weights for the instances misclassified using:

$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times \epsilon} \right) \quad (5)$$

- ➌ Decreases the weights for the instances correctly classified:

$$\mathbf{w}[i] \leftarrow \mathbf{w}[i] \times \left(\frac{1}{2 \times (1 - \epsilon)} \right) \quad (6)$$

- ➍ Calculate a **confidence factor**, α , for the model such that α increases as ϵ decreases:

$$\alpha = \frac{1}{2} \times \log_e \left(\frac{1 - \epsilon}{\epsilon} \right) \quad (7)$$

Boosting

- Once the set of models have been created the ensemble makes **predictions** using a weighted aggregate of the predictions made by the individual models.
- The weights used in this aggregation are simply the confidence factors associated with each model.

Bagging

- When we use **bagging** (or **bootstrap aggregating**) each model in the ensemble is trained on a random sample of the dataset known as **bootstrap samples**.
- Each random sample is the same size as the dataset and **sampling with replacement** is used.
- Consequently, every bootstrap sample will be missing some of the instances from the dataset so each bootstrap sample will be different and this means that models trained on different bootstrap samples will also be different

Bagging

- When bagging is used with decision trees each bootstrap sample only uses a randomly selected subset of the descriptive features in the dataset. This is known as **subspace sampling**.
- The combination of bagging, subspace sampling, and decision trees is known as a **random forest** model.

Bagging

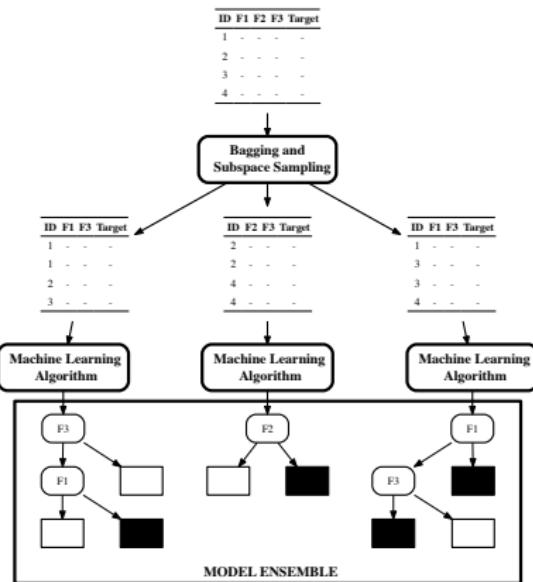


Figure: The process of creating a model ensemble using bagging and subspace sampling.

Bagging

- Which approach should we use? Bagging is simpler to implement and parallelize than boosting and, so, may be better with respect to ease of use and training time.
- Empirical results indicate:
 - boosted decision tree ensembles were the best performing model of those tested for datasets containing up to 4,000 descriptive features.
 - random forest ensembles (based on bagging) performed better for datasets containing more than 4,000 features.

Summary

- The **decision tree** model makes predictions based on sequences of tests on the descriptive feature values of a query
- The **ID3** algorithm as a standard algorithm for inducing decision trees from a dataset.

Decision Trees: Advantages

- interpretable.
- handle both categorical and continuous descriptive features.
- has the ability to model the interactions between descriptive features (diminished if **pre-pruning** is employed)
- relatively, robust to the **curse of dimensionality**.
- relatively, robust to noise in the dataset if **pruning** is used.

Decision Trees: Potential Disadvantages

- trees become large when dealing with continuous features.
- decision trees are very expressive and sensitive to the dataset, as a result they can overfit the data if there are a lot of features (curse of dimensionality)
- eager learner (concept drift).

- 1 Alternative Feature Selection Metrics
- 2 Handling Continuous Descriptive Features
- 3 Predicting Continuous Targets
- 4 Noisy Data, Overfitting and Tree Pruning
- 5 Model Ensembles
 - Boosting
 - Bagging
- 6 Summary

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 8: Evaluation

Sections 8.1, 8.2, 8.3

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

1 Big Idea

2 Fundamentals

3 Standard Approach: Measuring Misclassification Rate on a Hold-out Test Set

4 Summary

- The most important part of the design of an evaluation experiment for a predictive model is ensuring that the data used to evaluate the model is not the same as the data used to train the model.

- The purpose of evaluation is threefold:
 - ➊ to determine which model is the most suitable for a task
 - ➋ to estimate how the model will perform
 - ➌ to convince users that the model will meet their needs

Standard Approach: Measuring Misclassification Rate on a Hold-out Test Set

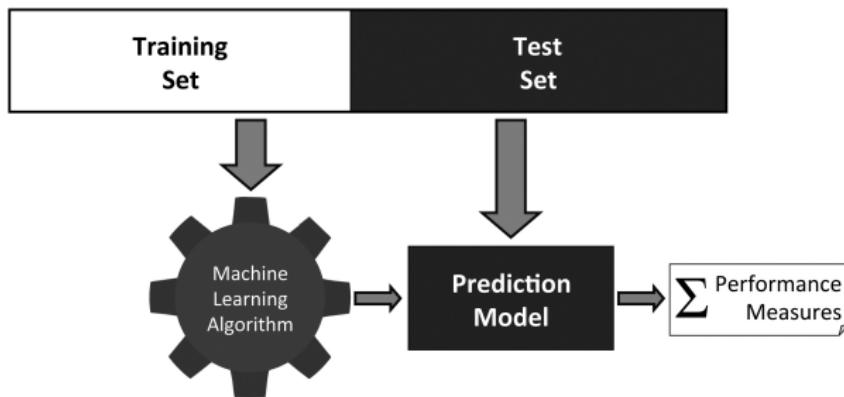


Figure: The process of building and evaluating a model using a **hold-out test set**.

Table: A sample test set with model predictions.

ID	Target	Pred.	Outcome	ID	Target	Pred.	Outcome
1	spam	ham	FN	11	ham	ham	TN
2	spam	ham	FN	12	spam	ham	FN
3	ham	ham	TN	13	ham	ham	TN
4	spam	spam	TP	14	ham	ham	TN
5	ham	ham	TN	15	ham	ham	TN
6	spam	spam	TP	16	ham	ham	TN
7	ham	ham	TN	17	ham	spam	FP
8	spam	spam	TP	18	spam	spam	TP
9	spam	spam	TP	19	ham	ham	TN
10	spam	spam	TP	20	ham	spam	FP

$$\text{misclassification rate} = \frac{\text{number incorrect predictions}}{\text{total predictions}} \quad (1)$$

$$\text{misclassification rate} = \frac{\text{number incorrect predictions}}{\text{total predictions}} \quad (1)$$

$$\text{misclassification rate} = \frac{(2 + 3)}{(6 + 9 + 2 + 3)} = 0.25$$

- For binary prediction problems there are 4 possible outcomes:
 - 1 True Positive (TP)
 - 2 True Negative (TN)
 - 3 False Positive (FP)
 - 4 False Negative (FN)

Table: The structure of a confusion matrix.

		Prediction	
		positive	negative
Target	positive	TP	FN
	negative	FP	TN

Table: A confusion matrix for the set of predictions shown in Table 1 [7].

		Prediction	
		'spam'	'ham'
Target	'spam'	6	3
	'ham'	2	9

$$\text{misclassification accuracy} = \frac{(FP + FN)}{(TP + TN + FP + FN)} \quad (2)$$

$$\text{misclassification accuracy} = \frac{(FP + FN)}{(TP + TN + FP + FN)} \quad (2)$$

$$\text{misclassification accuracy} = \frac{(2 + 3)}{(6 + 9 + 2 + 3)} = 0.25$$

$$\text{classification accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (3)$$

$$\text{classification accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (3)$$

$$\text{classification accuracy} = \frac{(6 + 9)}{(6 + 9 + 2 + 3)} = 0.75$$

Summary

1 Big Idea

2 Fundamentals

3 Standard Approach: Measuring Misclassification Rate on a Hold-out Test Set

4 Summary

Design
ooooooo

Cat. Targets
oooooooooooooo

Pred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
ooo

Deployment
oooooooooooo

Sum.

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 8: Evaluation
Sections 8.4, 8.5

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

1 Designing Evaluation Experiments

- Hold-out Sampling
- k-Fold Cross Validation
- Leave-one-out Cross Validation
- Bootstrapping
- Out-of-time Sampling

2 Performance Measures: Categorical Targets

- Confusion Matrix-based Performance Measures
- Precision, Recall and F_1 Measure
- Average Class Accuracy
- Measuring Profit and Loss

3 Performance Measures: Prediction Scores

- Receiver Operating Characteristic Curves
- Kolmogorov-Smirnov Statistic
- Measuring Gain and Lift

4 Performance Measures: Multinomial Targets

5 Performance Measures: Continuous Targets

- Basic Measures of Error
- Domain Independent Measures of Error

6 Evaluating Models after Deployment

- Monitoring Changes in Performance Measures
- Monitoring Model Output Distributions
- Monitoring Descriptive Feature Distribution Changes
- Comparative Experiments Using a Control Group

7 Summary

Design



Cat. Targets



Pred. Scores



Multinomial



Cont. Targets



Deployment



Sum.

Designing Evaluation Experiments

Design

Cat. Targets

Pred. Scores

Multinomial

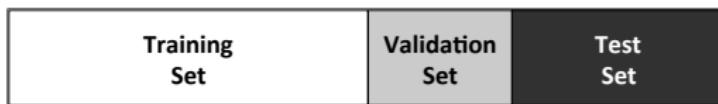
Cont. Targets

Deployment

Sum.



Hold-out Sampling



(a) A 50:20:30 split



(b) A 40:20:40 split

Figure: Hold-out sampling can divide the full data into training, validation, and test sets.

Design

Cat. Targets

Pred. Scores

Multinomial

Cont. Targets

Deployment

Sum.

○●○○○○

Hold-out Sampling

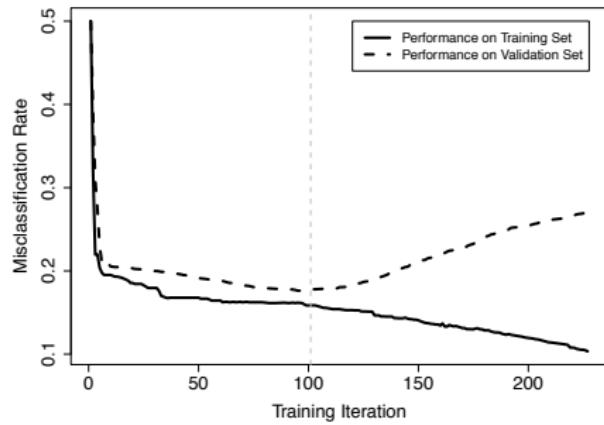


Figure: Using a validation set to avoid overfitting in iterative machine learning algorithms.

Design

Cat. Targets

Pred. Scores

Multinomial

Cont. Targets

Deployment

Sum.

k-Fold Cross Validation



Figure: The division of data during the **k-fold cross validation** process. Black rectangles indicate test data, and white spaces indicate training data.

Fold	Confusion Matrix				Class Accuracy
		Prediction			
		'lateral' 'frontal'			
1	Target	'lateral'	43	9	81%
		'frontal'	10	38	
2	Target	'lateral'	46	9	88%
		'frontal'	3	42	
3	Target	'lateral'	51	10	82%
		'frontal'	8	31	
4	Target	'lateral'	51	8	85%
		'frontal'	7	34	
5	Target	'lateral'	46	9	84%
		'frontal'	7	38	
Overall	Target	'lateral'	237	45	84%
		'frontal'	35	183	

Design

Cat. Targets

Pred. Scores

Multinomial

Cont. Targets

Deployment

Sum.

Leave-one-out Cross Validation

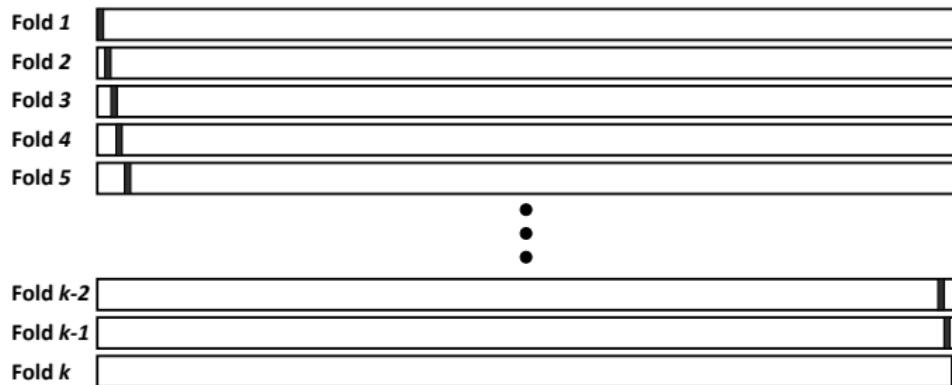


Figure: The division of data during the **leave-one-out cross validation** process. Black rectangles indicate instances in the test set, and white spaces indicate training data.

Design

Cat. Targets

Pred. Scores

Multinomial

Cont. Targets

Deployment

Sum.

oooooooooooo

oooooooooooooooooooo

oooooooooooooooooooooooo

ooo

oooooooooooo

Bootstrapping

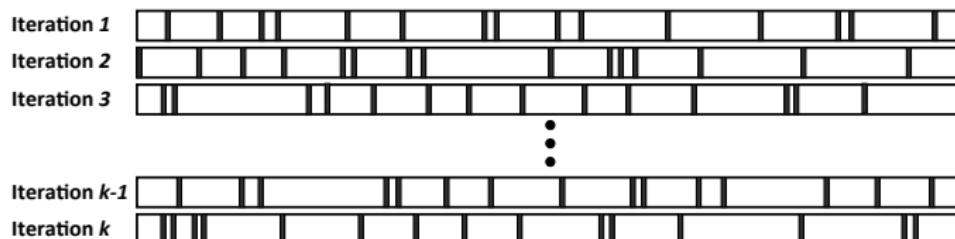


Figure: The division of data during the ϵ_0 bootstrap process. Black rectangles indicate test data, and white spaces indicate training data.

A random selection of m instances is taken from the dataset to generate the test set, the remaining instances are used for training. A performance measure is calculated for this iteration.

This process is repeated for k iterations.

Design

Cat. Targets

Pred. Scores

Multinomial

Cont. Targets

Deployment

Sum.

oooooooo●

oooooooooooooo

oooooooooooooooooooo

ooo

oooooooooooo

Out-of-time Sampling

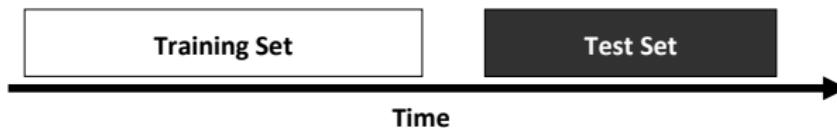


Figure: The **out-of-time sampling** process.

Design



Cat. Targets



Pred. Scores

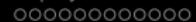


Multinomial

Cont. Targets



Deployment



Sum.

Performance Measures: Categorical Targets

Design

Cat. Targets



Pred. Scores
oooooooooooo

Multinomial

Cont. Targets

Deployment



Sum.

Confusion Matrix-based Performance Measures

$$\text{TPR} = \frac{TP}{(TP + FN)} \quad (1)$$

$$\text{TNR} = \frac{TN}{(TN + FP)} \quad (2)$$

$$FPR = \frac{FP}{(TN + FP)} \quad (3)$$

$$FNR = \frac{FN}{(TP + FN)} \quad (4)$$

Confusion Matrix-based Performance Measures

Table: A sample test set with model predictions.

ID	Target	Pred.	Outcome	ID	Target	Pred.	Outcome
1	spam	ham	FN	11	ham	ham	TN
2	spam	ham	FN	12	spam	ham	FN
3	ham	ham	TN	13	ham	ham	TN
4	spam	spam	TP	14	ham	ham	TN
5	ham	ham	TN	15	ham	ham	TN
6	spam	spam	TP	16	ham	ham	TN
7	ham	ham	TN	17	ham	spam	FP
8	spam	spam	TP	18	spam	spam	TP
9	spam	spam	TP	19	ham	ham	TN
10	spam	spam	TP	20	ham	spam	FP

$$TPR = \frac{TP}{(TP + FN)}$$

$$TPR = \frac{6}{(6+3)} = 0.667$$

$$TNR = \frac{TN}{(TN + FP)}$$

$$TNR = \frac{9}{(9+2)} = 0.818$$

$$FPR = \frac{FP}{(TN + FP)}$$

$$FPR = \frac{2}{(9+2)} = 0.182$$

$$FNR = \frac{FN}{(TP + FN)}$$

$$FNR = \frac{3}{(6+3)} = 0.333$$



Precision, Recall and F_1 Measure

$$\text{precision} = \frac{TP}{(TP + FP)} \quad (5)$$

$$\text{recall} = \frac{TP}{(TP + FN)} \quad (6)$$



Precision, Recall and F₁ Measure

$$\text{precision} = \frac{6}{(6 + 2)} = 0.75$$

$$\text{recall} = \frac{6}{(6 + 3)} = 0.667$$

Design Cat. Targets Pred. Scores Multinomial Cont. Targets Deployment Sum.
oooooooooooo●oooooooooooo ooooooooooooooooooooo

Precision, Recall and F_1 Measure

$$F_1\text{-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \quad (7)$$

Design
ooooooooCat. Targets
oooo●ooooooooooooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Precision, Recall and F₁ Measure

$$F_1\text{-measure} = 2 \times \frac{(\text{precision} \times \text{recall})}{(\text{precision} + \text{recall})} \quad (7)$$

$$\begin{aligned} F_1\text{-measure} &= 2 \times \frac{\left(\frac{6}{(6+2)} \times \frac{6}{(6+3)} \right)}{\left(\frac{6}{(6+2)} + \frac{6}{(6+3)} \right)} \\ &= 0.706 \end{aligned}$$

Average Class Accuracy

Table: A confusion matrix for a k -NN model trained on a churn prediction problem.

		Prediction		91% accuracy
		'non-churn'	'churn'	
Target	'non-churn'	90	0	
	'churn'	9	1	

Table: A confusion matrix for a naive Bayes model trained on a churn prediction problem.

		Prediction		78% accuracy
		'non-churn'	'churn'	
Target	'non-churn'	70	20	
	'churn'	2	8	

Design
ooooooooCat. Targets
oooooooo●ooooooooooooPred. Scores
oooooooooooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Average Class Accuracy

We'll use **average class accuracy** instead of **classification accuracy** to deal with the imbalanced data in the first table:

$$\text{average class accuracy} = \frac{1}{|levels(t)|} \sum_{l \in levels(t)} \text{recall}_l \quad (8)$$

$$\text{recall} = \frac{TP}{(TP + FN)}$$

Design

Cat. Targets

Pred. Scores
oooooooooooo

N

tinomial

Cont. Targets
ooo

Deployment
oooooooooooooo

Sum.

Average Class Accuracy

Alternative: use **harmonic mean** instead of **arithmetic mean**

$$\text{average class accuracy}_{\text{HM}} = \frac{1}{|levels(t)|} \sum_{l \in levels(t)} \frac{1}{\text{recall}_l} \quad (9)$$



Average Class Accuracy

$$\frac{1}{\frac{1}{2} \left(\frac{1}{1.0} + \frac{1}{0.1} \right)} = \frac{1}{5.5} = 18.2\%$$

$$\frac{1}{\frac{1}{2} \left(\frac{1}{0.778} + \frac{1}{0.800} \right)} = \frac{1}{1.268} = 78.873\%$$

Design

Cat. Targets

Pred. Scores

Multinomial

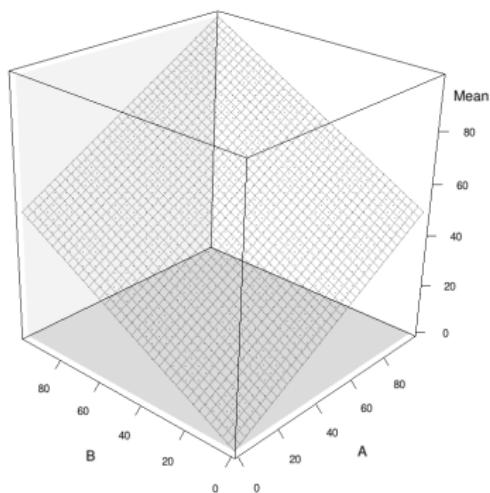
Cont. Targets

Deployment

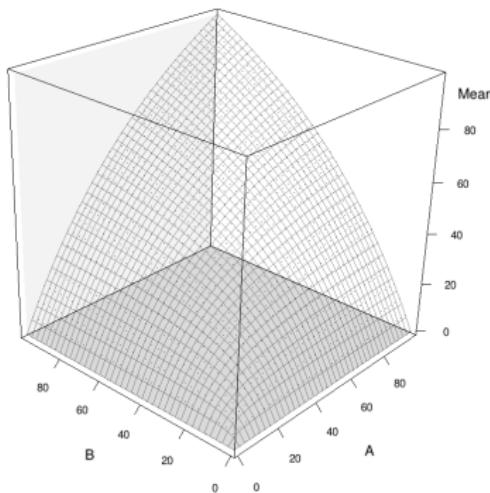
Sum.



Average Class Accuracy



(a)



(b)

Figure: Surfaces generated by calculating (a) the **arithmetic mean** and (b) the **harmonic mean** of all combinations of features A and B that range from 0 to 100.



Measuring Profit and Loss

- It is not always correct to treat all outcomes equally
- In these cases, it is useful to take into account the cost of the different outcomes when evaluating models

Design
oooooooCat. Targets
oooooooooooo●oooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Measuring Profit and Loss

Table: The structure of a **profit matrix**.

		Prediction	
		positive	negative
Target	positive	TP_{Profit}	FN_{Profit}
	negative	FP_{Profit}	TN_{Profit}

Design
ooooooooCat. Targets
oooooooooooo●ooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Measuring Profit and Loss

Table: The **profit matrix** for the pay-day loan credit scoring problem.

		Prediction	
		'good'	'bad'
Target	'good'	140	-140
	'bad'	-700	0

Design
ooooooooCat. Targets
oooooooooooo●●Pred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Measuring Profit and Loss

Table: (a) The confusion matrix for a k -NN model trained on the pay-day loan credit scoring problem (average class accuracy_{HM} = 83.824%); (b) the confusion matrix for a decision tree model trained on the pay-day loan credit scoring problem (average class accuracy_{HM} = 80.761%).

(a) k -NN model

		Prediction 'good' 'bad'	
		'good'	'bad'
Target	'good'	57	3
	'bad'	10	30

(b) decision tree

		Prediction 'good' 'bad'	
		'good'	'bad'
Target	'good'	43	17
	'bad'	3	37

Measuring Profit and Loss

Table: (a) Overall profit for the k -NN model using the profit matrix in Table 4 [25] and the **confusion matrix** in Table 5(a) [26]; (b) overall profit for the decision tree model using the profit matrix in Table 4 [25] and the **confusion matrix** in Table 5(b) [26].

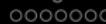
(a) k -NN model

		Prediction	
		'good'	'bad'
Target	'good'	7980	-420
	'bad'	-7000	0
Profit		560	

(b) decision tree

		Prediction	
		'good'	'bad'
Target	'good'	6020	-2380
	'bad'	-2100	0
Profit		1540	

Design



Cat. Targets



Pred. Scores



Multinomial



Cont. Targets



Deployment



Sum.

Performance Measures: Prediction Scores

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

- All our classification prediction models return a score which is then thresholded.

Example

$$\text{threshold}(\text{score}, 0.5) = \begin{cases} \text{positive} & \text{if } \text{score} \geq 0.5 \\ \text{negative} & \text{otherwise} \end{cases} \quad (10)$$

Table: A sample test set with model predictions and scores
 (threshold= 0.5.

ID	Target	Pred-iction	Score	Out-come	ID	Target	Pred-iction	Score	Out-come
7	ham	ham	0.001	TN	5	ham	ham	0.302	TN
11	ham	ham	0.003	TN	14	ham	ham	0.348	TN
15	ham	ham	0.059	TN	17	ham	spam	0.657	FP
13	ham	ham	0.064	TN	8	spam	spam	0.676	TP
19	ham	ham	0.094	TN	6	spam	spam	0.719	TP
12	spam	ham	0.160	FN	10	spam	spam	0.781	TP
2	spam	ham	0.184	FN	18	spam	spam	0.833	TP
3	ham	ham	0.226	TN	20	ham	spam	0.877	FP
16	ham	ham	0.246	TN	9	spam	spam	0.960	TP
1	spam	ham	0.293	FN	4	spam	spam	0.963	TP

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

- We have ordered the examples by score so the threshold is apparent in the predictions.
- Note that, in general, instances that actually should get a prediction of '*ham*' generally have a low score, and those that should get a prediction of '*spam*' generally get a high score.

Design
ooooooo

Cat. Targets
oooooooooooooo

Pred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
ooo

Deployment
oooooooooooo

Sum.

- There are a number of performance measures that use this ability of a model to rank instances that should get predictions of one target level higher than the other, to assess how well the model is performing.
- The basis of most of these approaches is measuring **how well the distributions of scores produced by the model for different target levels are separated**

Design Cat. Targets Pred. Scores Multinomial Cont. Targets Deployment Sum.
ooooooooooooooo ooooooooooooooooooooo ooooooooooooooooooooo
ooo ooooooooooooo

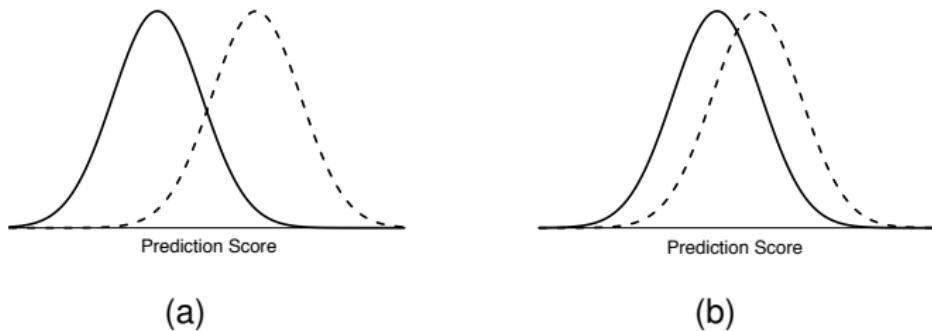
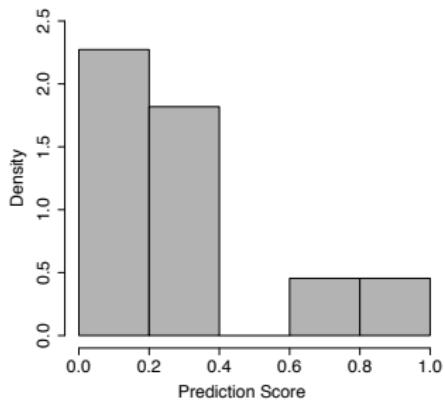
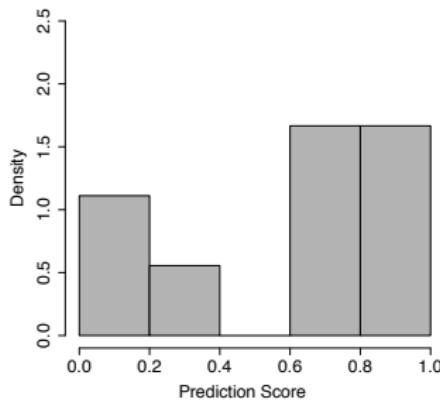


Figure: Prediction score distributions for two different prediction models. The distributions in (a) are much better separated than those in (b).



(a) spam



(b) ham

Figure: Prediction score distributions for the (a) 'spam' and (b) 'ham' target levels based on the data in Table 7 [30].

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
●oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Receiver Operating Characteristic Curves

- The **receiver operating characteristic index (ROC index)**, which is based on the **receiver operating characteristic curve (ROC curve)**, is a widely used performance measure that is calculated using prediction scores.
- TPR and TNR are intrinsically tied to the threshold used to convert prediction scores into target levels.
- This threshold can be changed, however, which leads to different predictions and a different confusion matrix.

Receiver Operating Characteristic Curves

Table: Confusion matrices for the set of predictions shown in Table 7^[30] using (a) a prediction score threshold of 0.75 and (b) a prediction score threshold of 0.25.

(a) Threshold: 0.75

		Prediction	
		'spam'	'ham'
Target	'spam'	4	4
	'ham'	2	10

(b) Threshold: 0.25

		Prediction	
		'spam'	'ham'
Target	'spam'	7	2
	'ham'	4	7

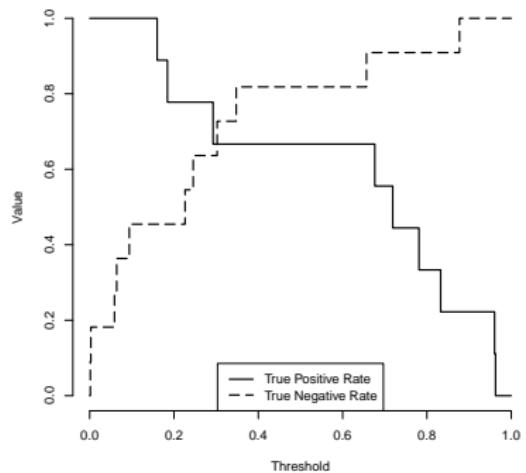
ID	Target	Score	Pred. (0.10)	Pred. (0.25)	Pred. (0.50)	Pred. (0.75)	Pred. (0.90)
7	ham	0.001	ham	ham	ham	ham	ham
11	ham	0.003	ham	ham	ham	ham	ham
15	ham	0.059	ham	ham	ham	ham	ham
13	ham	0.064	ham	ham	ham	ham	ham
19	ham	0.094	ham	ham	ham	ham	ham
12	spam	0.160	spam	ham	ham	ham	ham
2	spam	0.184	spam	ham	ham	ham	ham
3	ham	0.226	spam	ham	ham	ham	ham
16	ham	0.246	spam	ham	ham	ham	ham
1	spam	0.293	spam	spam	ham	ham	ham
5	ham	0.302	spam	spam	ham	ham	ham
14	ham	0.348	spam	spam	ham	ham	ham
17	ham	0.657	spam	spam	spam	ham	ham
8	spam	0.676	spam	spam	spam	ham	ham
6	spam	0.719	spam	spam	spam	ham	ham
10	spam	0.781	spam	spam	spam	spam	ham
18	spam	0.833	spam	spam	spam	spam	ham
20	ham	0.877	spam	spam	spam	spam	ham
9	spam	0.960	spam	spam	spam	spam	spam
4	spam	0.963	spam	spam	spam	spam	spam
Misclassification Rate			0.300	0.300	0.250	0.300	0.350
True Positive Rate (TPR)			1.000	0.778	0.667	0.444	0.222
True Negative rate (TNR)			0.455	0.636	0.818	0.909	1.000
False Positive Rate (FPR)			0.545	0.364	0.182	0.091	0.000
False Negative Rate (FNR)			0.000	0.222	0.333	0.556	0.778

Design Cat. Targets Pred. Scores Multinomial Cont. Targets Deployment Sum.
oooooooooooo ooooo●oooooooooooooooooooo ooo ooooooooooooo

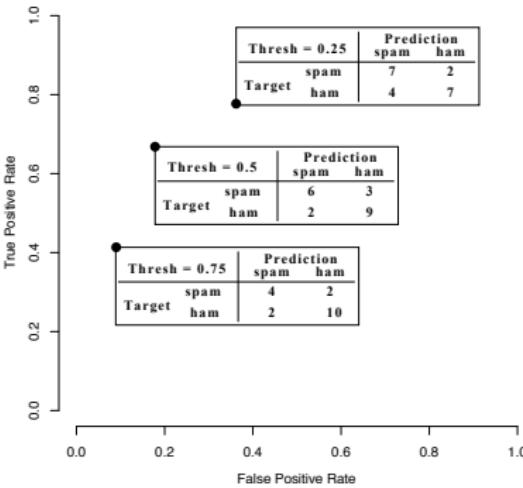
Receiver Operating Characteristic Curves

- Note: as the threshold increases TPR decreases and TNR increases (and vice versa).
- Capturing this tradeoff is the basis of the ROC curve.

Receiver Operating Characteristic Curves



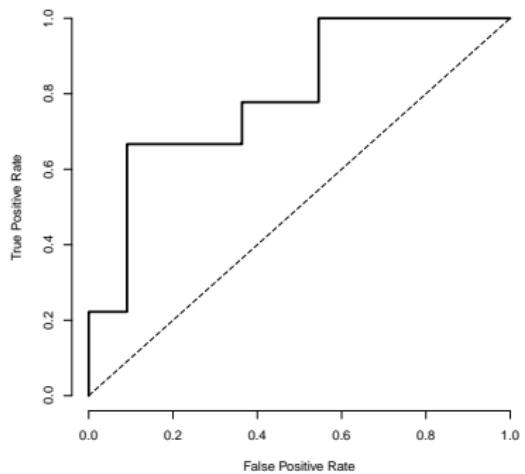
(a)



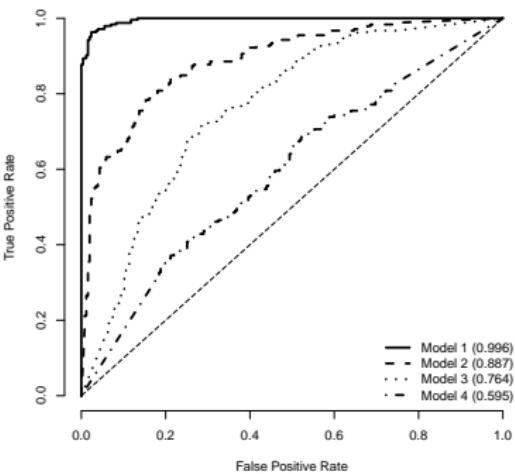
(b)

Figure: (a) The changing values of TPR and TNR for the test data shown in Table 36 [37] as the threshold is altered; (b) points in ROC space for thresholds of 0.25, 0.5, and 0.75.

Receiver Operating Characteristic Curves



(a)



(b)

Figure: (a) A complete ROC curve for the email classification example; (b) a selection of ROC curves for different models trained on the same prediction task.

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooo●oooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Receiver Operating Characteristic Curves

- We can also calculate a single performance measure from an ROC curve
- The **ROC Index** measures the area underneath an ROC curve.

ROC index =

$$\sum_{i=2}^{|T|} \frac{(FPR(T[i]) - FPR(T[i-1])) \times (TPR(T[i]) + TPR(T[i-1]))}{2} \quad (11)$$

Receiver Operating Characteristic Curves

- The **Gini coefficient** is a linear rescaling of the ROC index

$$\text{Gini coefficient} = (2 \times \text{ROC index}) - 1 \quad (12)$$

The Gini coefficient takes value in the range [0,1], the higher the value, the better the performance of the model

Design
00000

Cat. Targets

Pred. Scores

Pred. Scores

Multinomial

Cont. Targets

Deployment

Sum.

Kolmogorov-Smirnov Statistic

- The **Kolmogorov-Smirnov statistic** (K-S statistic) is another performance measure that captures the separation between the distribution of prediction scores for the different target levels in a classification problem.

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooooooo●ooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Kolmogorov-Smirnov Statistic

- To calculate the K-S statistic, we first determine the cumulative probability distributions of the prediction scores for the positive and negative target levels:

$$CP(\text{positive}, ps) = \frac{\text{num positive test instances with score} \leq ps}{\text{num positive test instances}} \quad (13)$$

$$CP(\text{negative}, ps) = \frac{\text{num negative test instances with score} \leq ps}{\text{num negative test instances}} \quad (14)$$

Design
00000

Cat. Targets

Pred. Scores

Multinomial

Cont. Targets

Deployment

Sum.

Kolmogorov-Smirnov Statistic

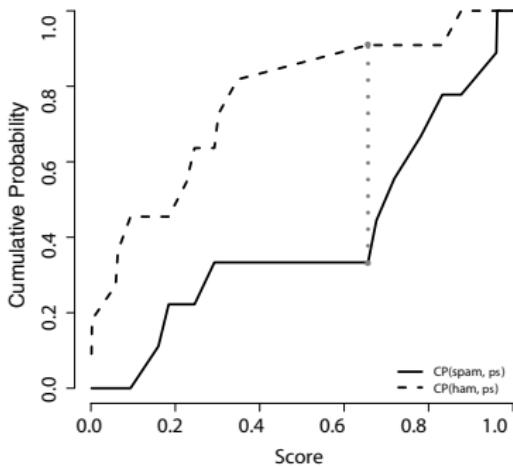


Figure: The K-S chart for the email classification predictions shown in Table 7 [30].

Kolmogorov-Smirnov Statistic

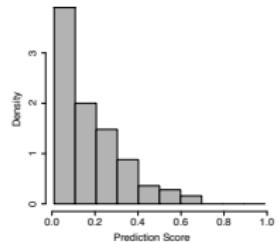
- The K-S statistic is calculated by determining the maximum difference between the cumulative probability distributions for the positive and negative target levels.

$$K-S = \max_{ps} (CP(\text{positive}, ps) - CP(\text{negative}, ps)) \quad (15)$$

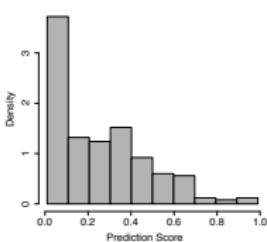
ID	Prediction Score	Positive ('spam')	Negative ('ham')	Positive ('spam')	Negative ('ham')	Distance
		Cumulative Count	Cumulative Count	Cumulative Probability	Cumulative Probability	
7	0.001	0	1	0.000	0.091	0.091
11	0.003	0	2	0.000	0.182	0.182
15	0.059	0	3	0.000	0.273	0.273
13	0.064	0	4	0.000	0.364	0.364
19	0.094	0	5	0.000	0.455	0.455
12	0.160	1	5	0.111	0.455	0.343
2	0.184	2	5	0.222	0.455	0.232
3	0.226	2	6	0.222	0.545	0.323
16	0.246	2	7	0.222	0.636	0.414
1	0.293	3	7	0.333	0.636	0.303
5	0.302	3	8	0.333	0.727	0.394
14	0.348	3	9	0.333	0.818	0.485
17	0.657	3	10	0.333	0.909	0.576*
8	0.676	4	10	0.444	0.909	0.465
6	0.719	5	10	0.556	0.909	0.354
10	0.781	6	10	0.667	0.909	0.242
18	0.833	7	10	0.778	0.909	0.131
20	0.877	7	11	0.778	1.000	0.222
9	0.960	8	11	0.889	1.000	0.111
4	0.963	9	11	1.000	1.000	0.000

(*much better model)

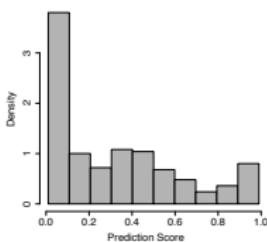
(a) Model 1



(b) Model 2



(c) Model 3



(d) Model 4

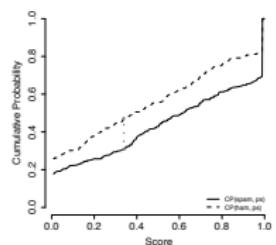
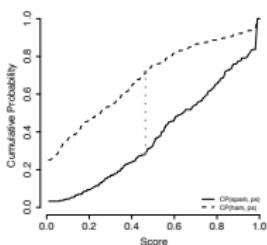
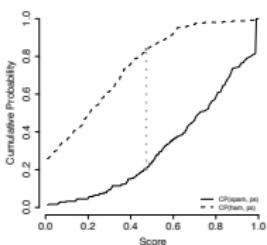
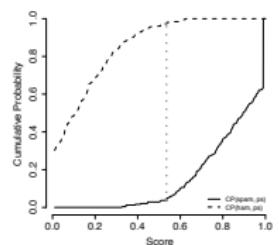
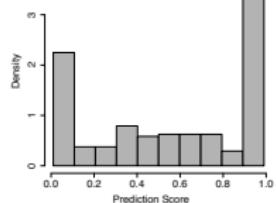
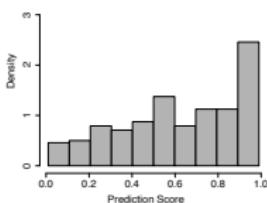
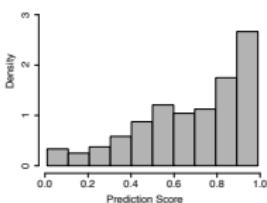
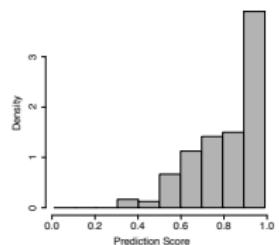
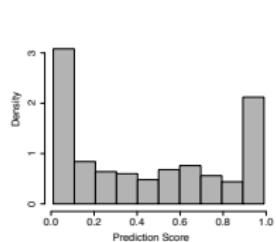


Figure: A series of charts for different model performance on the same large email classification test set used to generate the ROC

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Measuring Gain and Lift

Measuring gain and lift

If we are to rank the instances in the test data in descending order of prediction scores, we would expect the majority of the positive instances to be toward the top of this ranking.

The gain and lift attempt to measure to what extent a set of predictions made by a model meet this assumption

$$\text{Gain}(\text{dec}) = \frac{\text{num positive test instances in decile } \text{dec}}{\text{num positive test instances}} \quad (16)$$

Table: The test set with model predictions and scores from Table 7 [30] extended to include deciles.

Decile	ID	Target	Prediction	Score	Outcome
1 st	9	spam	spam	0.960	TP
	4	spam	spam	0.963	TP
2 nd	18	spam	spam	0.833	TP
	20	ham	spam	0.877	FP
3 rd	6	spam	spam	0.719	TP
	10	spam	spam	0.781	TP
4 th	17	ham	spam	0.657	FP
	8	spam	spam	0.676	TP
5 th	5	ham	ham	0.302	TN
	14	ham	ham	0.348	TN
6 th	16	ham	ham	0.246	TN
	1	spam	ham	0.293	FN
7 th	2	spam	ham	0.184	FN
	3	ham	ham	0.226	TN
8 th	19	ham	ham	0.094	TN
	12	spam	ham	0.160	FN
9 th	15	ham	ham	0.059	TN
	13	ham	ham	0.064	TN
10 th	7	ham	ham	0.001	TN
	11	ham	ham	0.003	TN

Design
ooooooooCat. Targets
oooooooooooooooPred. Scores
oooooooooooooooooooo●oooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

Measuring Gain and Lift

Table: Tabulating the workings required to calculate **gain**, **cumulative gain**, **lift**, and **cumulative lift** for the data given in Table 7 [30].

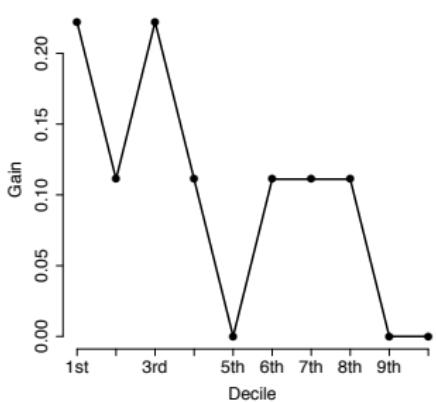
Decile	Positive ('spam')	Negative ('ham')	Gain	Cum. Gain	Lift	Cum. Lift
	Count	Count				
1 st	2	0	0.222	0.222	2.222	2.222
2 nd	1	1	0.111	0.333	1.111	1.667
3 rd	2	0	0.222	0.556	2.222	1.852
4 th	1	1	0.111	0.667	1.111	1.667
5 th	0	2	0.000	0.667	0.000	1.333
6 th	1	1	0.111	0.778	1.111	1.296
7 th	1	1	0.111	0.889	1.111	1.270
8 th	1	1	0.111	1.000	1.111	1.250
9 th	0	2	0.000	1.000	0.000	1.111
10 th	0	2	0.000	1.000	0.000	1.000



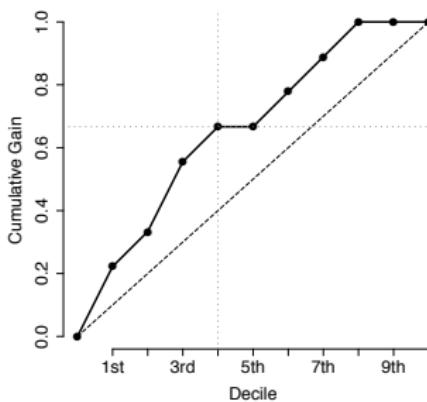
Measuring Gain and Lift

$$\text{Cumulative gain}(dec) = \frac{\text{num positive test instances in all deciles up to } dec}{\text{num positive test instances}} \quad (17)$$

Measuring Gain and Lift



(a)



(b)

Figure: The (a) **gain** and (b) **cumulative gain** at each decile for the email predictions given in Table 7 [30].



Measuring Gain and Lift

$$\text{Lift}(dec) = \frac{\% \text{ of positive test instances in decile } dec}{\% \text{ of positive test instances}} \quad (18)$$

Measuring Gain and Lift

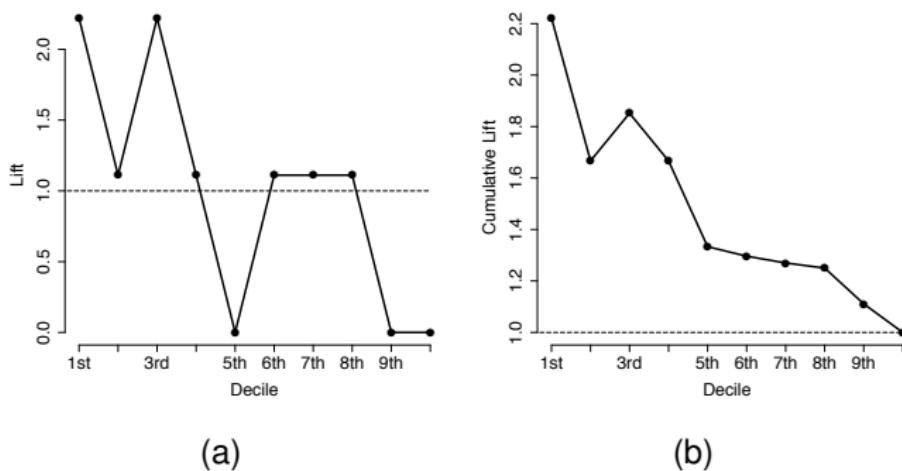
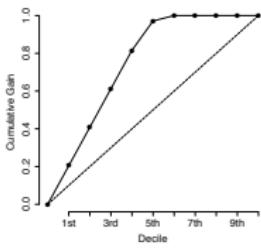


Figure: The (a) **lift** and (b) **cumulative lift** at each decile for the email predictions given in Table 7 [30].

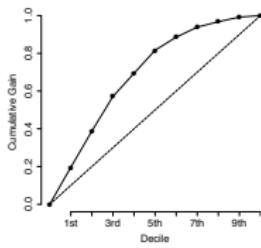
Measuring Gain and Lift

$$\text{Cumulative lift}(dec) = \frac{\% \text{ of positive instances in all deciles up to } dec}{\% \text{ of positive test instances}} \quad (19)$$

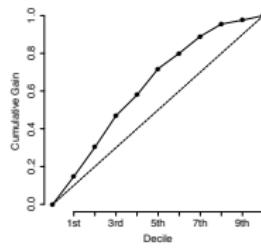
(a) Model 1



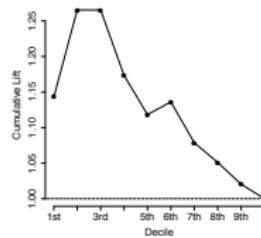
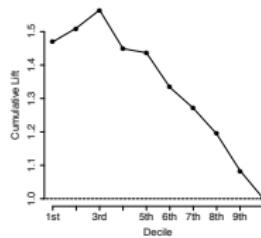
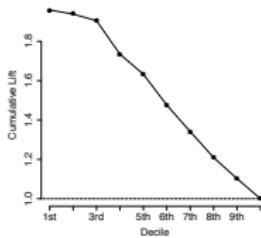
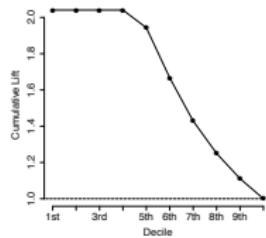
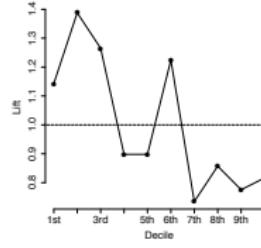
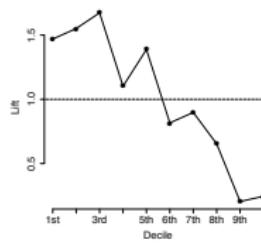
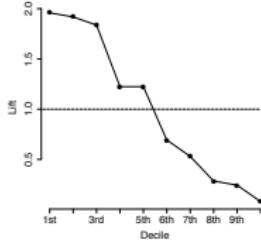
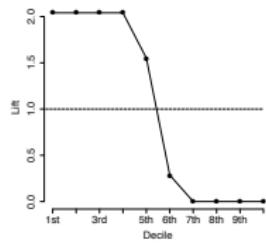
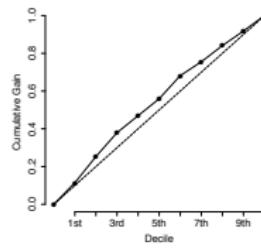
(b) Model 2



(c) Model 3



(d) Model 4



Design
ooooooo

Cat. Targets
oooooooooooooo

Pred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
ooo

Deployment
oooooooooooo

Sum.

Performance Measures: Multinomial Targets

Table: The structure of a confusion matrix for a multinomial prediction problem with l target levels.

		Prediction					Recall
		<i>level1</i>	<i>level2</i>	<i>level3</i>	...	<i>levell</i>	
Target	<i>level1</i>	-	-	-	-	-	-
	<i>level2</i>	-	-	-	-	-	-
	<i>level3</i>	-	-	-	-	-	-
	:				.	.	:
	<i>levell</i>	-	-	-	-	-	-
	Precision	-	-	-	-	...	-



$$\text{precision}(I) = \frac{TP(I)}{TP(I) + FP(I)} \quad (20)$$

$$\text{recall}(I) = \frac{TP(I)}{TP(I) + FN(I)} \quad (21)$$

Table: A sample test set with model predictions for a bacterial species identification problem.

ID	Target	Prediction	ID	Target	Prediction
1	durionis	fructosus	16	ficulneus	ficulneus
2	ficulneus	fructosus	17	ficulneus	ficulneus
3	fructosus	fructosus	18	fructosus	fructosus
4	ficulneus	ficulneus	19	durionis	durionis
5	durionis	durionis	20	fructosus	fructosus
6	pseudo.	pseudo.	21	fructosus	fructosus
7	durionis	fructosus	22	durionis	durionis
8	ficulneus	ficulneus	23	fructosus	fructosus
9	pseudo.	pseudo.	24	pseudo.	fructosus
10	pseudo.	fructosus	25	durionis	durionis
11	fructosus	fructosus	26	pseudo.	pseudo.
12	ficulneus	ficulneus	27	fructosus	fructosus
13	durionis	durionis	28	ficulneus	ficulneus
14	fructosus	fructosus	29	fructosus	fructosus
15	fructosus	ficulneus	30	fructosus	fructosus

Table: A confusion matrix for a model trained on the bacterial species identification problem.

		Prediction				Recall
		'durionis'	'ficalneus'	'fructosus'	'pseudo.'	
Target	'durionis'	5	0	2	0	0.714
	'ficalneus'	0	6	1	0	0.857
	'fructosus'	0	1	10	0	0.909
	'pseudo.'	0	0	2	3	0.600
	Precision	1.000	0.857	0.667	1.000	

Design
ooooooooCat. Targets
oooooooooooooooPred. Scores
oooooooooooooooooooo

Multinomial

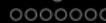
Cont. Targets
oooDeployment
oooooooooooo

Sum.

- The average class accuracy_{HM} for this problem is:

$$\frac{1}{4} \left(\frac{1}{0.714} + \frac{1}{0.857} + \frac{1}{0.909} + \frac{1}{0.600} \right) = \frac{1}{1.333} = 75.000\%$$

Design



Cat. Targets



Pred. Scores



Multinomial



Cont. Targets



Deployment



Sum.

Performance Measures: Continuous Targets

Basic Measures of Error

$$\text{sum of squared errors} = \frac{1}{2} \sum_{i=1}^n (t_i - \mathbb{M}(\mathbf{d}_i))^2 \quad (22)$$

$$\text{mean squared error} = \frac{\sum_{i=1}^n (t_i - \mathbb{M}(\mathbf{d}_i))^2}{n} \quad (23)$$

$$\text{root mean squared error} = \sqrt{\frac{\sum_{i=1}^n (t_i - \mathbb{M}(\mathbf{d}_i))^2}{n}} \quad (24)$$

$$\text{mean absolute error} = \frac{\sum_{i=1}^n abs(t_i - \mathbb{M}(\mathbf{d}_i))}{n} \quad (25)$$

ID	Target	Linear Regression		<i>k</i> -NN	
		Prediction	Error	Prediction	Error
1	10.502	10.730	0.228	12.240	1.738
2	18.990	17.578	-1.412	21.000	2.010
3	20.000	21.760	1.760	16.973	-3.027
4	6.883	7.001	0.118	7.543	0.660
5	5.351	5.244	-0.107	8.383	3.032
6	11.120	10.842	-0.278	10.228	-0.892
7	11.420	10.913	-0.507	12.921	1.500
8	4.836	7.401	2.565	7.588	2.752
9	8.177	8.227	0.050	9.277	1.100
10	19.009	16.667	-2.341	21.000	1.991
11	13.282	14.424	1.142	15.496	2.214
12	8.689	9.874	1.185	5.724	-2.965
13	18.050	19.503	1.453	16.449	-1.601
14	5.388	7.020	1.632	6.640	1.252
15	10.646	10.358	-0.288	5.840	-4.805
16	19.612	16.219	-3.393	18.965	-0.646
17	10.576	10.680	0.104	8.941	-1.634
18	12.934	14.337	1.403	12.484	-0.451
19	10.492	10.366	-0.126	13.021	2.529
20	13.439	14.035	0.596	10.920	-2.519
21	9.849	9.821	-0.029	9.920	0.071
22	18.045	16.639	-1.406	18.526	0.482
23	6.413	7.225	0.813	7.719	1.307
24	9.522	9.565	0.043	8.934	-0.588
25	12.083	13.048	0.965	11.241	-0.842
26	10.104	10.085	-0.020	10.010	-0.095
27	8.924	9.048	0.124	8.157	-0.767
28	10.636	10.876	0.239	13.409	2.773
29	5.457	4.080	-1.376	9.684	4.228
30	3.538	7.090	3.551	5.553	2.014
MSE		1.905		4.394	
RMSE		1.380		2.096	
MAE		0.975		1.750	
R²		0.889		0.776	

Domain Independent Measures of Error

$$R^2 = 1 - \frac{\text{sum of squared errors}}{\text{total sum of squares}} \quad (26)$$

$$\text{total sum of squares} = \frac{1}{2} \sum_{i=1}^n (t_i - \bar{t})^2 \quad (27)$$

coefficient of determination
range [0,1), the larger the value the better the performance

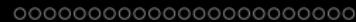
Design



Cat. Targets



Pred. Scores



Multinomial



Cont. Targets



Deployment



Sum.

Evaluating Models after Deployment

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooooooo

Sum.

To monitor the on-going performance of a model, we need a signal that indicates that something has changed. There are three sources from which we can extract such a signal:

- ➊ The performance of the model measured using appropriate performance measures
- ➋ The distributions of the outputs of a model
- ➌ The distributions of the descriptive features in query instances presented to the model

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
●oooooooooooo

Sum.

Monitoring Changes in Performance Measures

- The simplest way to get a signal that concept drift has occurred is to repeatedly evaluate models with the same performance measures used to evaluate them before deployment.
- We can calculate performance measures for a deployed model and compare these to the performance achieved in evaluations before the model was deployed.
- If the performance changes significantly, this is a strong indication that **concept drift** has occurred and that the model has gone stale.

Design
oooooooo

Cat. Targets
oooooooooooo

Pred. Scores
oooooooooooooooooooo

Multinomial
ooo

Cont. Targets
ooo

Deployment
o●oooooooooooo

Sum.

Monitoring Changes in Performance Measures

- Although monitoring changes in the performance of a model is the easiest way to tell whether it has gone stale, this method makes the rather large assumption that the correct target feature value for a query instance will be made available shortly after the query has been presented to a deployed model.

Monitoring Model Output Distributions

- An alternative to using changing model performance is to use changes in the distribution of model outputs as a signal for concept drift.

$$\text{stability index} = \sum_{l \in levels(t)} \left(\left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} - \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \times \log_e \left(\frac{|\mathcal{A}_{t=l}|}{|\mathcal{A}|} / \frac{|\mathcal{B}_{t=l}|}{|\mathcal{B}|} \right) \right) \quad (28)$$

Design
ooooooo

Cat. Targets
oooooooooooooo

Pred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
ooo

Deployment
ooo●oooo

Sum.

Monitoring Model Output Distributions

In general,

- stability index < 0.1 , then the distribution of the newly collected test set is broadly similar to the distribution in the original test set.
- stability index is between 0.1 and 0.25, then some change has occurred and further investigation may be useful.
- stability index > 0.25 suggests that a significant change has occurred and corrective action is required.

Table: Calculating the **stability index** for the bacterial species identification problem given new test data for two periods after model deployment. The frequency and percentage of each target level are shown for the original test set and for two samples collected after deployment. The column marked SI_t shows the different parts of the stability index sum based on Equation (28)^[72].

Target	Original		New Sample 1			New Sample 2		
	Count	%	Count	%	SI_t	Count	%	SI_t
'durionis'	7	0.233	12	0.267	0.004	12	0.200	0.005
'ficalneus'	7	0.233	8	0.178	0.015	9	0.150	0.037
'fructosus'	11	0.367	16	0.356	0.000	14	0.233	0.060
'pseudo.'	5	0.167	9	0.200	0.006	25	0.417	0.229
Sum	30		45		0.026	60		0.331

Design
ooooooooCat. Targets
ooooooooooooooooooooPred. Scores
oooooooooooooooooooooooooooo

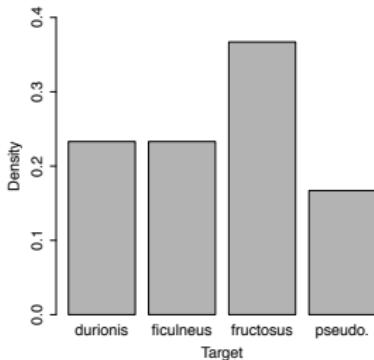
Multinomial

Cont. Targets
oooDeployment
oooo●●○○○○○

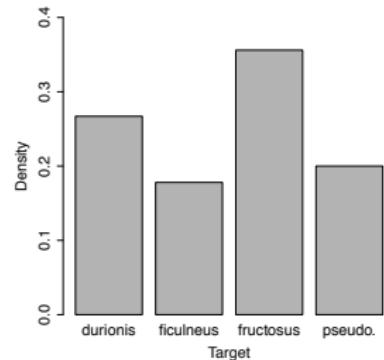
Sum.

Monitoring Model Output Distributions

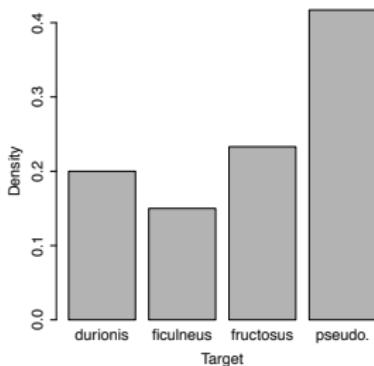
$$\begin{aligned}\text{stability index} &= \left(\frac{7}{30} - \frac{12}{45} \right) \times \log_e \left(\frac{7}{30} / \frac{12}{45} \right) \\ &\quad + \left(\frac{7}{30} - \frac{8}{45} \right) \times \log_e \left(\frac{7}{30} / \frac{8}{45} \right) \\ &\quad + \left(\frac{11}{30} - \frac{16}{45} \right) \times \log_e \left(\frac{11}{30} / \frac{16}{45} \right) \\ &\quad + \left(\frac{5}{30} - \frac{9}{45} \right) \times \log_e \left(\frac{5}{30} / \frac{9}{45} \right) \\ &= 0.026\end{aligned}$$



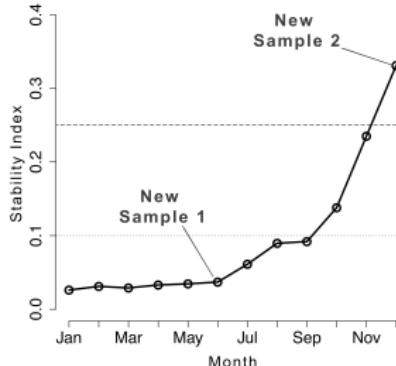
(a) Original



(b) New Sample 1



(c) New Sample 2



(d) Monitoring Over Time

Design
oooooooCat. Targets
ooooooooooooooooooooPred. Scores
oooooooooooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooo•oooo

Sum.

Monitoring Descriptive Feature Distribution Changes

- In the same way we can compare the distributions of model outputs between the time that the model was built and after deployment, we can also make the same type of comparison for the distributions of the descriptive features used by the model.
- We can use any appropriate measure that captures the difference between two different distributions for this, including the stability index, the χ^2 statistic, and the K-S statistic.

Design
oooooooCat. Targets
ooooooooooooooPred. Scores
oooooooooooooooooooo

Multinomial

Cont. Targets
oooDeployment
oooooooo●ooo

Sum.

Monitoring Descriptive Feature Distribution Changes

- There is, however, a challenge here, as usually, there are a large number of descriptive features for which measures need to be calculated and tracked.
- Furthermore, it is unlikely that a change in the distribution of just one descriptive feature in a multi-feature model will have a large impact on model performance.
- For this reason, unless a model uses a very small number of descriptive features (generally fewer than 10), we do not recommend this approach.



Comparative Experiments Using a Control Group

- We use control groups not to evaluate the predictive power of the models themselves, but rather to evaluate how good they are at helping with the business problem when they are deployed.

Table: The number of customers who left the mobile phone network operator each week during the comparative experiment from both the control group (random selection) and the treatment group (model selection).

Week	Control Group (Random Selection)	Treatment Group (Model Selection)
1	21	23
2	18	15
3	28	18
4	19	20
5	18	15
6	17	17
7	23	18
8	24	20
9	19	18
10	20	19
11	18	13
12	21	16
Mean	20.500	17.667
Std. Dev.	3.177	2.708



Comparative Experiments Using a Control Group

- These figures show that, on average, fewer customers churn when the churn prediction model is used to select which customers to call.



Summary

1 Designing Evaluation Experiments

- Hold-out Sampling
- k-Fold Cross Validation
- Leave-one-out Cross Validation
- Bootstrapping
- Out-of-time Sampling

2 Performance Measures: Categorical Targets

- Confusion Matrix-based Performance Measures
- Precision, Recall and F_1 Measure
- Average Class Accuracy
- Measuring Profit and Loss

3 Performance Measures: Prediction Scores

- Receiver Operating Characteristic Curves
- Kolmogorov-Smirnov Statistic
- Measuring Gain and Lift

4 Performance Measures: Multinomial Targets

5 Performance Measures: Continuous Targets

- Basic Measures of Error
- Domain Independent Measures of Error

6 Evaluating Models after Deployment

- Monitoring Changes in Performance Measures
- Monitoring Model Output Distributions
- Monitoring Descriptive Feature Distribution Changes
- Comparative Experiments Using a Control Group

7 Summary

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 5: Similarity-based Learning
Sections 5.1, 5.2, 5.3

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

1 Big Idea**2 Fundamentals**

- Feature Space
- Distance Metrics

3 Standard Approach: The Nearest Neighbor Algorithm

- A Worked Example

4 Epilogue**5 Summary**

Big Idea

- The year is 1798 and you are Lieutenant-Colonel David Collins of HMS Calcutta who is exploring the region around Hawkesbury River, in New South Wales.
- After an expedition up the river one of the men tells you that he saw a strange animal near the river.
- You ask him to describe the animal to you and he explains that he didn't see it very well but that he did notice that it had webbed feet and a duck-bill snout, and that it growled at him.
- In order to plan the expedition for the next day you decide that you need to classify the animal so that you can figure out whether it is dangerous to approach it or not.

	<i>Grrrh!</i>			Score
	✓	✗	✗	1
	✗	✓	✗	1
	✗	✓	✓	2

Figure: Matching animals you remember to the features of the unknown animal described by the sailor. Note: The images used in this figure were created by Jan Gillbank for the English for the Australian Curriculum website (<http://www.e4ac.edu.au>) and are used under the Create Commons Attribution 3.0 Unported licence (<http://creativecommons.org/licenses/by/3.0>). The images were sourced via Wikimedia Commons.

- The process of classifying an unknown animal by matching the features of the animal against the features of animals you can remember neatly encapsulates the big idea underpinning similarity-based learning:

if you are trying to classify something then you should search your memory to find things that are similar and label it with the same class as the most similar thing in your memory

- One of the simplest and best known machine learning algorithms for this type of reasoning is called the **nearest neighbor** algorithm.

Big Idea

Fundamentals
oooooooooooo

Standard Approach
ooooooo

Epilogue

Summary

Fundamentals

- The fundamentals of similarity-based learning are:
 - Feature space
 - Similarity metrics

Feature Space

Table: The speed and agility ratings for 20 college athletes labelled with the decisions for whether they were drafted or not.

ID	Speed	Agility	Draft	ID	Speed	Agility	Draft
1	2.50	6.00	No	11	2.00	2.00	No
2	3.75	8.00	No	12	5.00	2.50	No
3	2.25	5.50	No	13	8.25	8.50	No
4	3.25	8.25	No	14	5.75	8.75	Yes
5	2.75	7.50	No	15	4.75	6.25	Yes
6	4.50	5.00	No	16	5.50	6.75	Yes
7	3.50	5.25	No	17	5.25	9.50	Yes
8	3.00	3.25	No	18	7.00	4.25	Yes
9	4.00	4.00	No	19	7.50	8.00	Yes
10	4.25	3.75	No	20	7.25	5.75	Yes

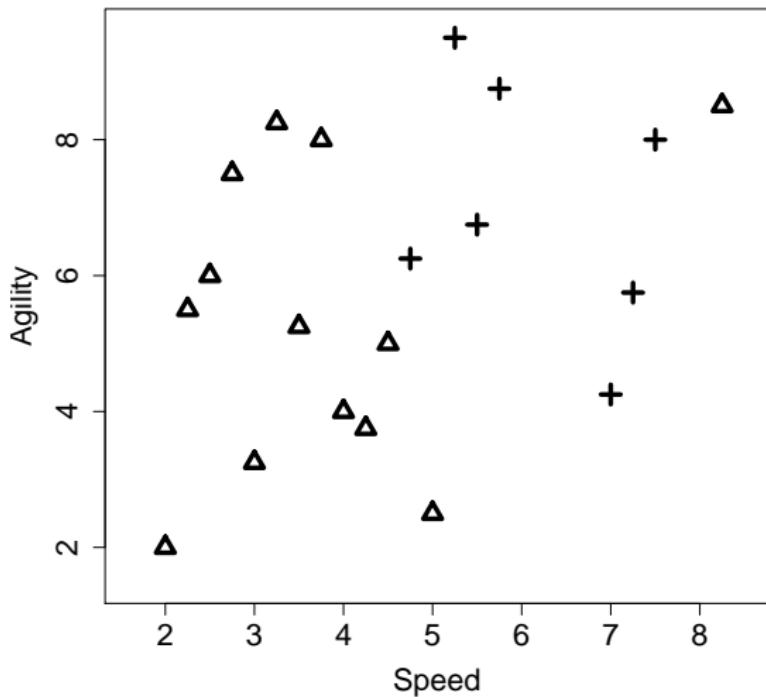


Figure: A feature space plot of the data in Table 2 [25]. The triangles represent '*Non-draft*' instances and the crosses represent the '*Draft*' instances.

Feature Space

- A **feature space** is an abstract n-dimensional space that is created by taking each of the descriptive features in an ABT to be the axes of a reference space and each instance in the dataset is mapped to a point in the feature space based on the values of its descriptive features.

Distance Metrics

- A **similarity metric** measures the similarity between two instances according to a feature space
- Mathematically, a **metric** must conform to the following four criteria:
 - ① **Non-negativity**: $\text{metric}(\mathbf{a}, \mathbf{b}) \geq 0$
 - ② **Identity**: $\text{metric}(\mathbf{a}, \mathbf{b}) = 0 \iff \mathbf{a} = \mathbf{b}$
 - ③ **Symmetry**: $\text{metric}(\mathbf{a}, \mathbf{b}) = \text{metric}(\mathbf{b}, \mathbf{a})$
 - ④ **Triangular Inequality**:
$$\text{metric}(\mathbf{a}, \mathbf{b}) \leq \text{metric}(\mathbf{a}, \mathbf{c}) + \text{metric}(\mathbf{b}, \mathbf{c})$$

where $\text{metric}(\mathbf{a}, \mathbf{b})$ is a function that returns the distance between two instances **a** and **b**.

Distance Metrics

- One of the best known metrics is **Euclidean distance** which computes the length of the straight line between two points. Euclidean distance between two instances **a** and **b** in a m -dimensional feature space is defined as:

$$\text{Euclidean}(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^m (\mathbf{a}[i] - \mathbf{b}[i])^2} \quad (1)$$

Example

The Euclidean distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75,AGILITY= 7.5) in Table 2 [25] is:

Example

The Euclidean distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75,AGILITY= 7.5) in Table 2 [25] is:

$$\begin{aligned} \text{Euclidean}(\langle 5.00, 2.50 \rangle, \langle 2.75, 7.50 \rangle) &= \sqrt{(5.00 - 2.75)^2 + (2.50 - 7.50)^2} \\ &= \sqrt{30.0625} = 5.4829 \end{aligned}$$

Distance Metrics

- Another, less well known, distance measure is the **Manhattan** distance or **taxi-cab distance**.
- The Manhattan distance between two instances **a** and **b** in a feature space with m dimensions is:¹

$$\text{Manhattan}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^m \text{abs}(\mathbf{a}[i] - \mathbf{b}[i]) \quad (2)$$

¹The *abs()* function surrounding the subtraction term indicates that we use the absolute value, i.e. non-negative value, when we are summing the differences; this makes sense because distances can't be negative.

Distance Metrics

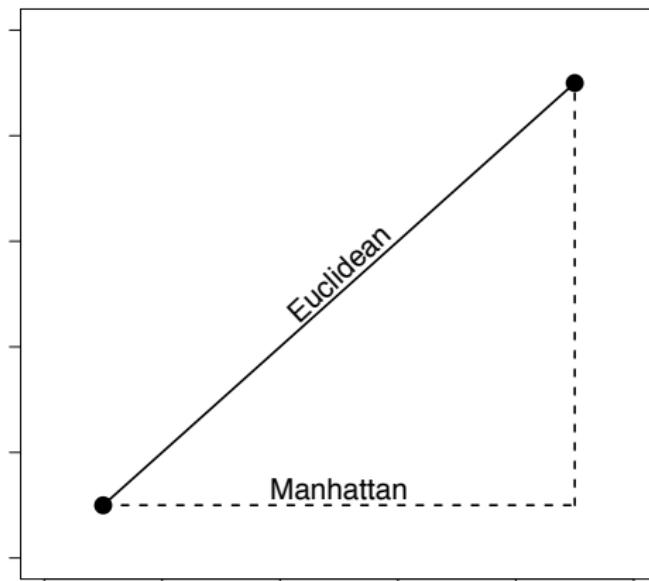


Figure: The Manhattan and Euclidean distances between two points.

Example

The Manhattan distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75,AGILITY= 7.5) in Table 2 [25] is:

Example

The Manhattan distance between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75,AGILITY= 7.5) in Table 2 [25] is:

$$\begin{aligned} \text{Manhattan}(\langle 5.00, 2.50 \rangle, \langle 2.75, 7.50 \rangle) &= \text{abs}(5.00 - 2.75) + \text{abs}(2.5 - 7.5) \\ &= 2.25 + 5 = 7.25 \end{aligned}$$

Distance Metrics

- The Euclidean and Manhattan distances are special cases of **Minkowski distance**
- The **Minkowski distance** between two instances **a** and **b** in a feature space with m descriptive features is:

$$\text{Minkowski}(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^m \text{abs}(\mathbf{a}[i] - \mathbf{b}[i])^p \right)^{\frac{1}{p}} \quad (3)$$

where different values of the parameter p result in different distance metrics

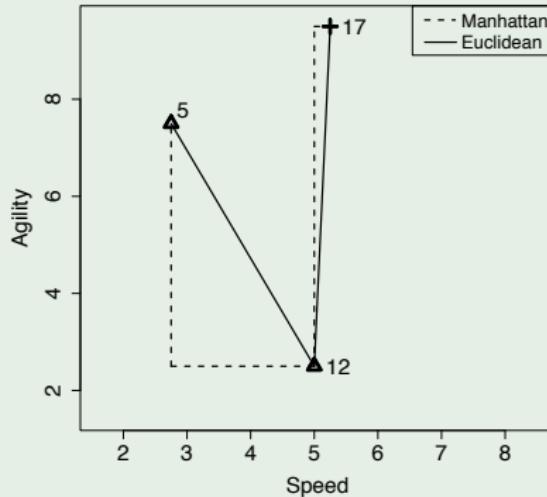
- The Minkowski distance with $p = 1$ is the Manhattan distance and with $p = 2$ is the Euclidean distance.

Distance Metrics

- The larger the value of p the more emphasis is placed on the features with large differences in values because these differences are raised to the power of p .

Example

Instance ID	Instance ID	Manhattan (Minkowski p=1)	Euclidean (Minkowski p=2)
12	5	7.25	5.4829
12	17	7.25	8.25



The Manhattan and Euclidean distances between instances d_{12} (SPEED= 5.00, AGILITY= 2.5) and d_5 (SPEED= 2.75, AGILITY= 7.5) and between instances d_{12} and d_{17} (SPEED= 5.25, AGILITY= 9.5).

Standard Approach: The Nearest Neighbor Algorithm

The Nearest Neighbour Algorithm

Require: set of training instances

Require: a query to be classified

- 1: Iterate across the instances in memory and find the instance that is shortest distance from the query position in the feature space.
- 2: Make a prediction for the query equal to the value of the target feature of the nearest neighbor.

A Worked Example

Table: The speed and agility ratings for 20 college athletes labelled with the decisions for whether they were drafted or not.

ID	Speed	Agility	Draft	ID	Speed	Agility	Draft
1	2.50	6.00	No	11	2.00	2.00	No
2	3.75	8.00	No	12	5.00	2.50	No
3	2.25	5.50	No	13	8.25	8.50	No
4	3.25	8.25	No	14	5.75	8.75	Yes
5	2.75	7.50	No	15	4.75	6.25	Yes
6	4.50	5.00	No	16	5.50	6.75	Yes
7	3.50	5.25	No	17	5.25	9.50	Yes
8	3.00	3.25	No	18	7.00	4.25	Yes
9	4.00	4.00	No	19	7.50	8.00	Yes
10	4.25	3.75	No	20	7.25	5.75	Yes

A Worked Example

Example

- Should we draft an athlete with the following profile:

SPEED= 6.75, AGILITY= 3

A Worked Example

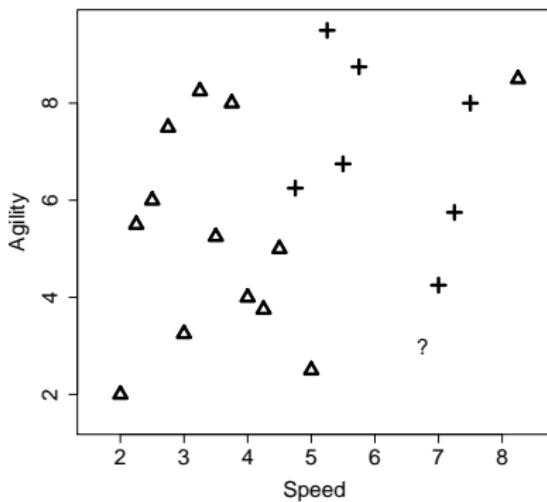


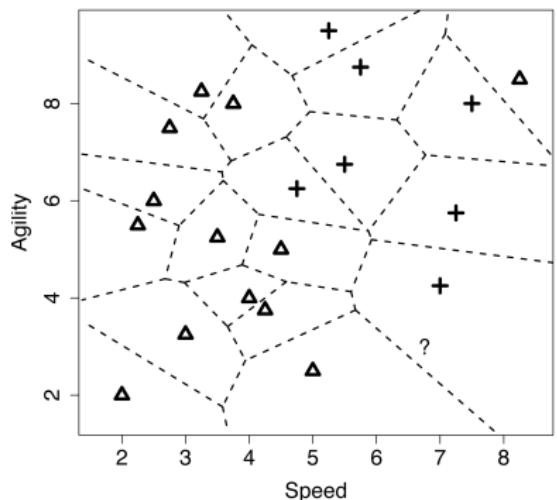
Figure: A feature space plot of the data in Table 2^[25] with the position in the feature space of the query represented by the ? marker. The triangles represent 'Non-draft' instances and the crosses represent the 'Draft' instances.

A Worked Example

Table: The distances (Dist.) between the query instance with SPEED = 6.75 and AGILITY = 3.00 and each instance in Table 2 [25].

ID	SPEED	AGILITY	DRAFT	Dist.	ID	SPEED	AGILITY	DRAFT	Dist.
18	7.00	4.25	yes	1.27	11	2.00	2.00	no	4.85
12	5.00	2.50	no	1.82	19	7.50	8.00	yes	5.06
10	4.25	3.75	no	2.61	3	2.25	5.50	no	5.15
20	7.25	5.75	yes	2.80	1	2.50	6.00	no	5.20
9	4.00	4.00	no	2.93	13	8.25	8.50	no	5.70
6	4.50	5.00	no	3.01	2	3.75	8.00	no	5.83
8	3.00	3.25	no	3.76	14	5.75	8.75	yes	5.84
15	4.75	6.25	yes	3.82	5	2.75	7.50	no	6.02
7	3.50	5.25	no	3.95	4	3.25	8.25	no	6.31
16	5.50	6.75	yes	3.95	17	5.25	9.50	yes	6.67

A Worked Example



(a) Voronoi tessellation

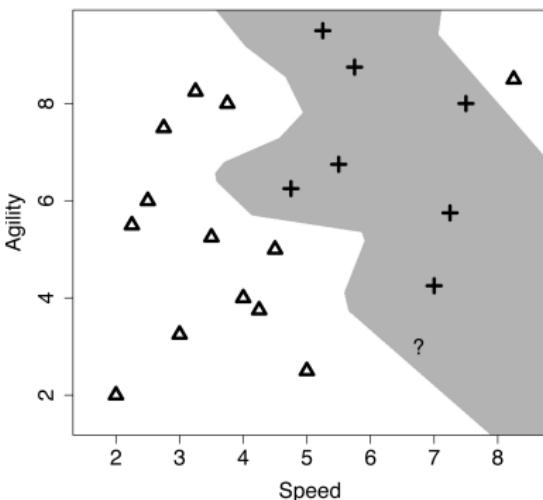
(b) Decision boundary ($k = 1$)

Figure: (a) The Voronoi tessellation of the feature space for the dataset in Table 2^[25] with the position of the query represented by the ? marker; (b) the decision boundary created by aggregating the neighboring Voronoi regions that belong to the same target level.

A Worked Example

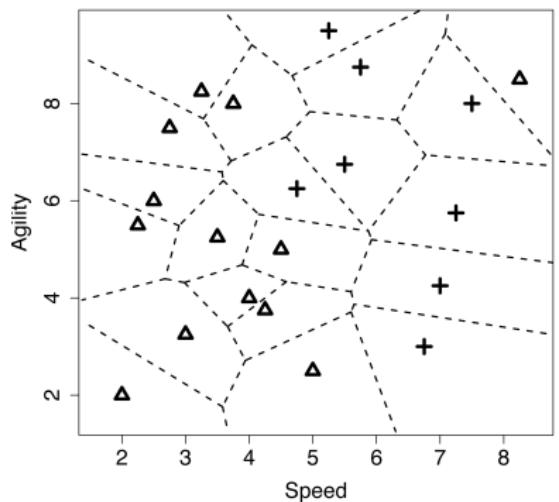
- One of the great things about nearest neighbour algorithms is that we can add in new data to update the model very easily.

A Worked Example

Table: The extended version of the college athletes dataset.

ID	SPEED	AGILITY	DRAFT	ID	SPEED	AGILITY	DRAFT
1	2.50	6.00	no	12	5.00	2.50	no
2	3.75	8.00	no	13	8.25	8.50	no
3	2.25	5.50	no	14	5.75	8.75	yes
4	3.25	8.25	no	15	4.75	6.25	yes
5	2.75	7.50	no	16	5.50	6.75	yes
6	4.50	5.00	no	17	5.25	9.50	yes
7	3.50	5.25	no	18	7.00	4.25	yes
8	3.00	3.25	no	19	7.50	8.00	yes
9	4.00	4.00	no	20	7.25	5.75	yes
10	4.25	3.75	no	21	6.75	3.00	yes
11	2.00	2.00	no				

A Worked Example



(a) Voronoi tessellation

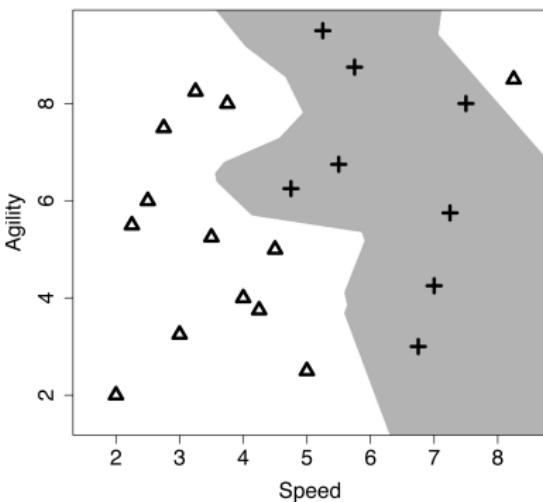
(b) Decision boundary ($k = 1$)

Figure: (a) The Voronoi tessellation of the feature space when the dataset has been updated to include the query instance; (b) the updated decision boundary reflecting the addition of the query instance in the training set.

Big Idea

Fundamentals
oooooooooooo

Standard Approach
oooooooo

Epilogue

Summary

Epilogue

- Returning to 1798 and HMS Calcutta, the next day you accompany your men on the expedition up the river and you encounter the strange animal the sailor had described to you.
- This time when you see the animal yourself you realize that it definitely isn't a duck!
- It turns out that you and your men are the first Europeans to encounter a platypus².

²The story recounted here of the discovery of the platypus is loosely based on real events.



Figure: A duck-billed platypus. The platypus image used in here was created by Jan Gillbank for the English for the Australian Curriculum website (<http://www.e4ac.edu.au>) and are used under the Create Commons Attribution 3.0 Unported licence (<http://creativecommons.org/licenses/by/3.0>). The image was sourced via Wikimedia Commons.

- This epilogue illustrates two important, and related, aspects of supervised machine learning:
 - ① supervised machine learning is based on the **stationarity assumption** which states that the data doesn't change - remains stationary - over time.
 - ② in the context of classification, supervised machine learning creates models that distinguish between the classes that are present in the dataset they are induced from. So, if a classification model is trained to distinguish between lions, frogs and ducks, the model will classify a query as being either a lion, a frog or a duck; even if the query is actually a platypus.

Big Idea

Fundamentals
oooooooooooo

Standard Approach
oooooooo

Epilogue

Summary

Summary

- Similarity-based prediction models attempt to mimic a very human way of reasoning by basing predictions for a target feature value on the most similar instances in memory—this makes them easy to interpret and understand.
- This advantage should not be underestimated as being able to understand how the model works gives people more confidence in the model and, hence, in the insight that it provides.

- The inductive bias underpinning similarity-based classification is that things that are similar (i.e., instances that have similar descriptive features) belong to the same class.
- The nearest neighbor algorithm creates an implicit global predictive model by aggregating local models, or neighborhoods.
- The definition of these neighborhoods is based on proximity within the feature space to the labelled training instances.
- Queries are classified using the label of the training instance defining the neighborhood in the feature space that contains the query.

1 Big Idea**2 Fundamentals**

- Feature Space
- Distance Metrics

3 Standard Approach: The Nearest Neighbor Algorithm

- A Worked Example

4 Epilogue**5 Summary**

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 5: Similarity-based Learning
Sections 5.4, 5.5

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

- 1 Handling Noisy Data**
- 2 Data Normalization**
- 3 Predicting Continuous Targets**
- 4 Other Measures of Similarity**
- 5 Feature Selection**
- 6 Efficient Memory Search**
- 7 Summary**

Handling Noisy Data

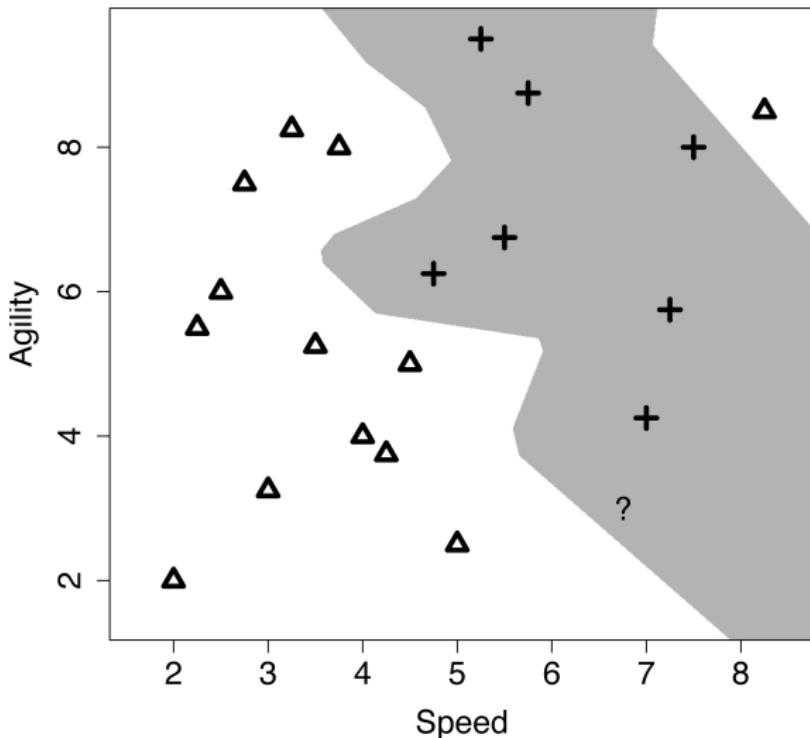


Figure: Is the instance at the top right of the diagram really *noise*?

- The **k nearest neighbors** model predicts the target level with the majority vote from the set of k nearest neighbors to the query \mathbf{q} :

$$\mathbb{M}_k(\mathbf{q}) = \operatorname{argmax}_{l \in levels(t)} \sum_{i=1}^k \delta(t_i, l) \quad (1)$$

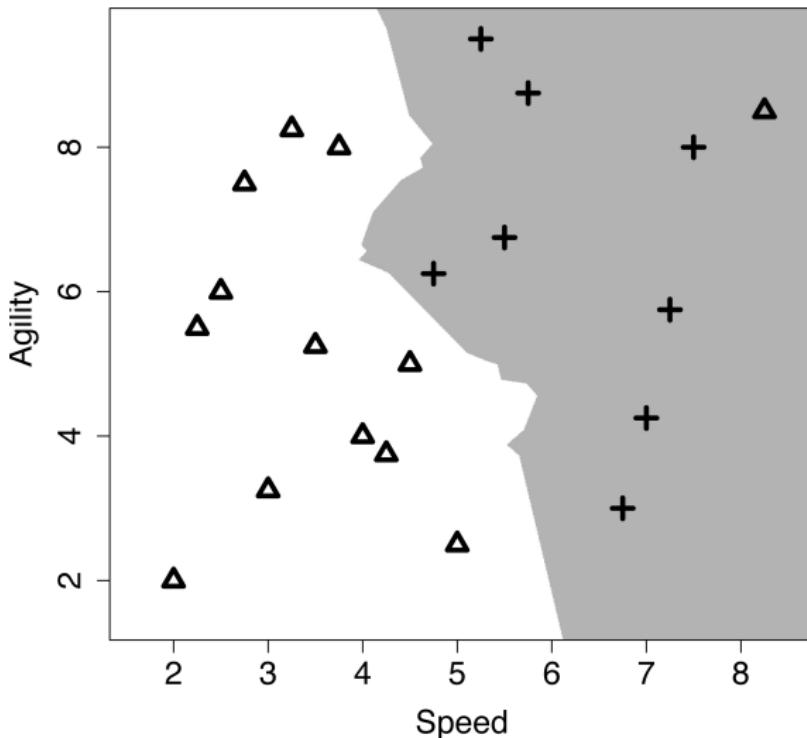


Figure: The decision boundary using majority classification of the nearest 3 neighbors.

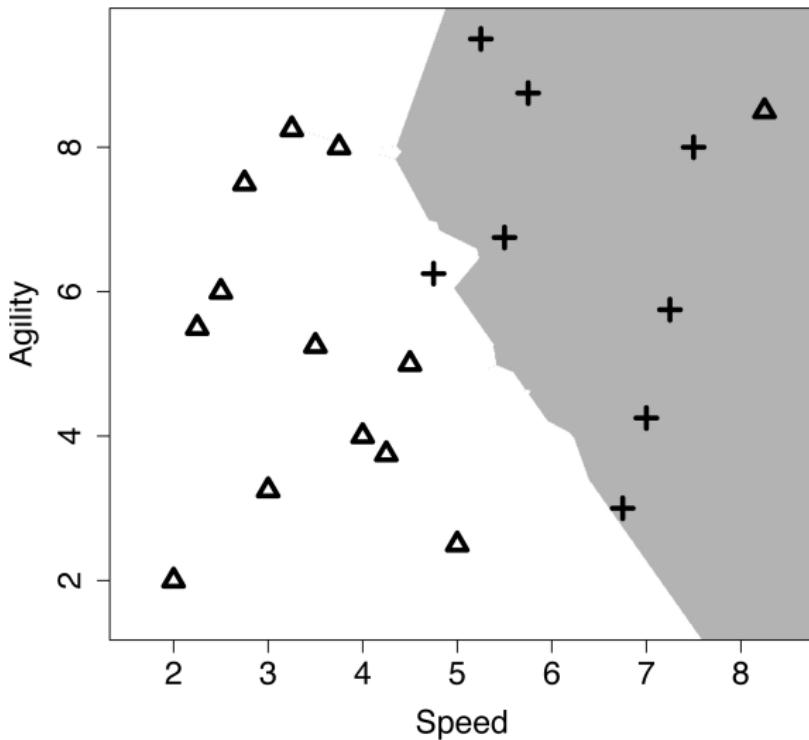


Figure: The decision boundary using majority classification of the nearest 5 neighbors.

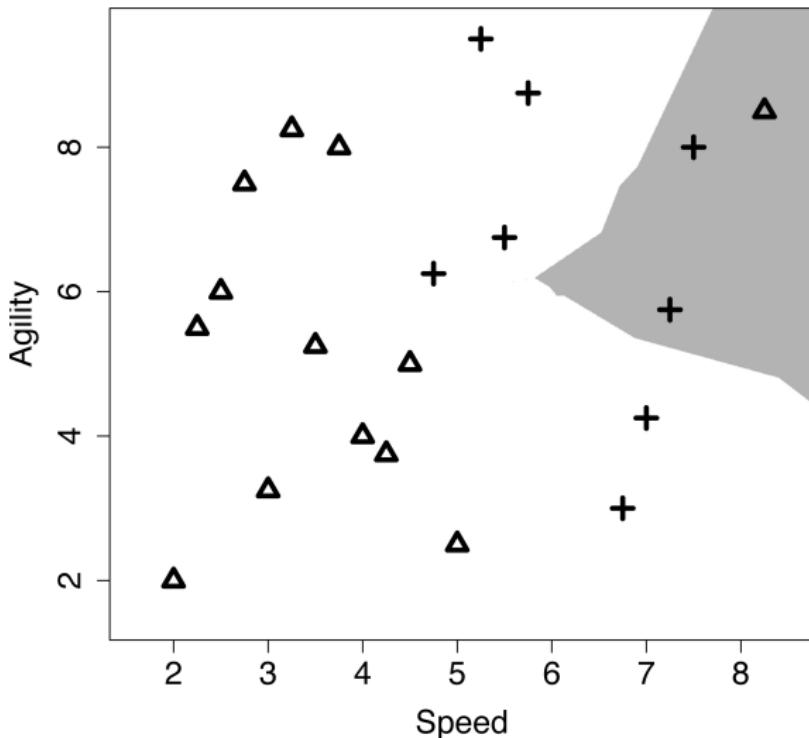


Figure: The decision boundary when k is set to 15.

- In a distance **weighted k nearest neighbor** algorithm the contribution of each neighbor to the classification decision is weighted by the reciprocal of the squared distance between the neighbor \mathbf{d} and the query \mathbf{q} :

$$\frac{1}{dist(\mathbf{q}, \mathbf{d})^2} \quad (2)$$

- The weighted k nearest neighbor model is defined as:

$$\mathbb{M}_k(\mathbf{q}) = \operatorname{argmax}_{l \in levels(t)} \sum_{i=1}^k \frac{1}{dist(\mathbf{q}, \mathbf{d}_i)^2} \times \delta(t_i, l) \quad (3)$$

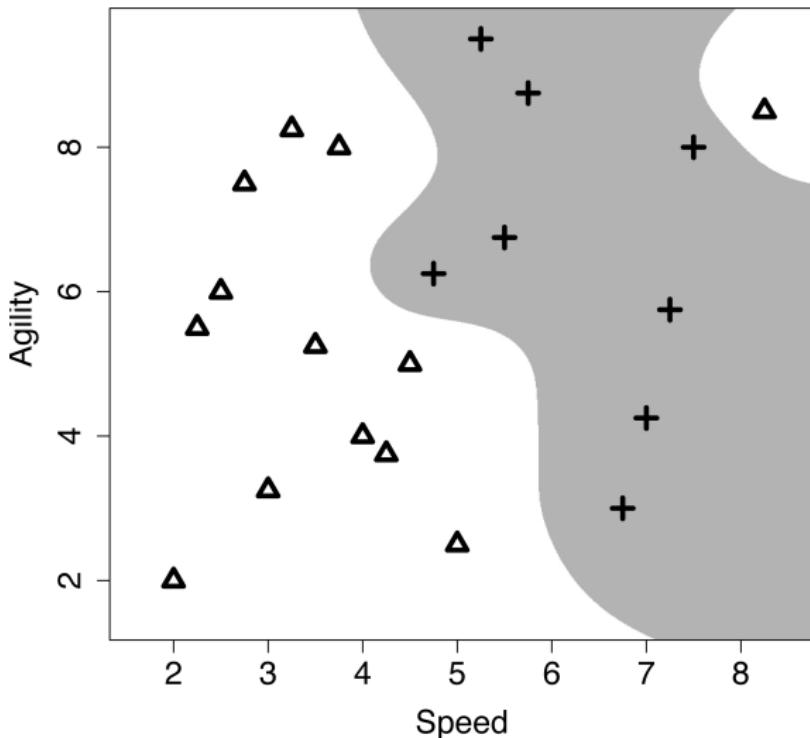


Figure: The weighted k nearest neighbor model decision boundary.

Data Normalization

Table: A dataset listing the salary and age information for customers and whether or not the purchased a pension plan .

ID	Salary	Age	Purchased
1	53700	41	No
2	65300	37	No
3	48900	45	Yes
4	64800	49	Yes
5	44200	30	No
6	55900	57	Yes
7	48600	26	No
8	72800	60	Yes
9	45300	34	No
10	73200	52	Yes

- The marketing department wants to decide whether or not they should contact a customer with the following profile:

$\langle \text{SALARY} = 56,000, \text{AGE} = 35 \rangle$

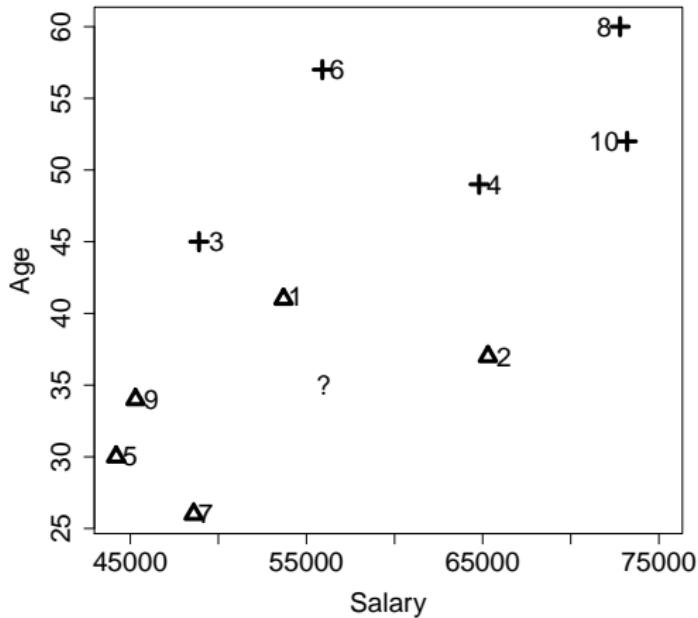


Figure: The salary and age feature space with the data in Table 1 [12] plotted. The instances are labelled their IDs, triangles represent the negative instances and crosses represent the positive instances. The location of the query $\langle \text{SALARY} = 56,000, \text{AGE} = 35 \rangle$ is indicated by the ?.

ID	Salary	Age	Purch.	Salary and Age		Salary Only		Age Only	
				Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	53700	41	No	2300.0078	2	2300	2	6	4
2	65300	37	No	9300.0002	6	9300	6	2	2
3	48900	45	Yes	7100.0070	3	7100	3	10	6
4	64800	49	Yes	8800.0111	5	8800	5	14	7
5	44200	30	No	11800.0011	8	11800	8	5	5
6	55900	57	Yes	102.3914	1	100	1	22	9
7	48600	26	No	7400.0055	4	7400	4	9	3
8	72800	60	Yes	16800.0186	9	16800	9	25	10
9	45300	34	No	10700.0000	7	10700	7	1	1
10	73200	52	Yes	17200.0084	10	17200	10	17	8

- This odd prediction is caused by features taking different ranges of values, this is equivalent to features having different variances.
- We can adjust for this using normalization; the equation for range normalization is:

$$a'_i = \frac{a_i - \min(a)}{\max(a) - \min(a)} \times (\text{high} - \text{low}) + \text{low} \quad (4)$$

ID	Normalized Dataset			Salary and Age		Salary Only		Age Only	
	Salary	Age	Purch.	Dist.	Neigh.	Dist.	Neigh.	Dist.	Neigh.
1	0.3276	0.4412	No	0.1935	1	0.0793	2	0.17647	4
2	0.7276	0.3235	No	0.3260	2	0.3207	6	0.05882	2
3	0.1621	0.5588	Yes	0.3827	5	0.2448	3	0.29412	6
4	0.7103	0.6765	Yes	0.5115	7	0.3034	5	0.41176	7
5	0.0000	0.1176	No	0.4327	6	0.4069	8	0.14706	3
6	0.4034	0.9118	Yes	0.6471	8	0.0034	1	0.64706	9
7	0.1517	0.0000	No	0.3677	3	0.2552	4	0.26471	5
8	0.9862	1.0000	Yes	0.9361	10	0.5793	9	0.73529	10
9	0.0379	0.2353	No	0.3701	4	0.3690	7	0.02941	1
10	1.0000	0.7647	Yes	0.7757	9	0.5931	10	0.50000	8

- Normalizing the data is an important thing to do for almost all machine learning algorithms, not just nearest neighbor!

Predicting Continuous Targets

- Return the average value in the neighborhood:

$$\mathbb{M}_k(\mathbf{q}) = \frac{1}{k} \sum_{i=1}^k t_i \quad (5)$$

Table: A dataset of whiskeys listing the age (in years) and the rating (between 1 and 5, with 5 being the best) and the bottle price of each whiskey.

ID	Age	Rating	Price	ID	Age	Rating	Price
1	0	2	30.00	11	19	5	500.00
2	12	3.5	40.00	12	6	4.5	200.00
3	10	4	55.00	13	8	3.5	65.00
4	21	4.5	550.00	14	22	4	120.00
5	12	3	35.00	15	6	2	12.00
6	15	3.5	45.00	16	8	4.5	250.00
7	16	4	70.00	17	10	2	18.00
8	18	3	85.00	18	30	4.5	450.00
9	18	3.5	78.00	19	1	1	10.00
10	16	3	75.00	20	4	3	30.00

Table: The whiskey dataset after the descriptive features have been normalized.

ID	Age	Rating	Price	ID	Age	Rating	Price
1	0.0000	0.25	30.00	11	0.6333	1.00	500.00
2	0.4000	0.63	40.00	12	0.2000	0.88	200.00
3	0.3333	0.75	55.00	13	0.2667	0.63	65.00
4	0.7000	0.88	550.00	14	0.7333	0.75	120.00
5	0.4000	0.50	35.00	15	0.2000	0.25	12.00
6	0.5000	0.63	45.00	16	0.2667	0.88	250.00
7	0.5333	0.75	70.00	17	0.3333	0.25	18.00
8	0.6000	0.50	85.00	18	1.0000	0.88	450.00
9	0.6000	0.63	78.00	19	0.0333	0.00	10.00
10	0.5333	0.50	75.00	20	0.1333	0.50	30.00

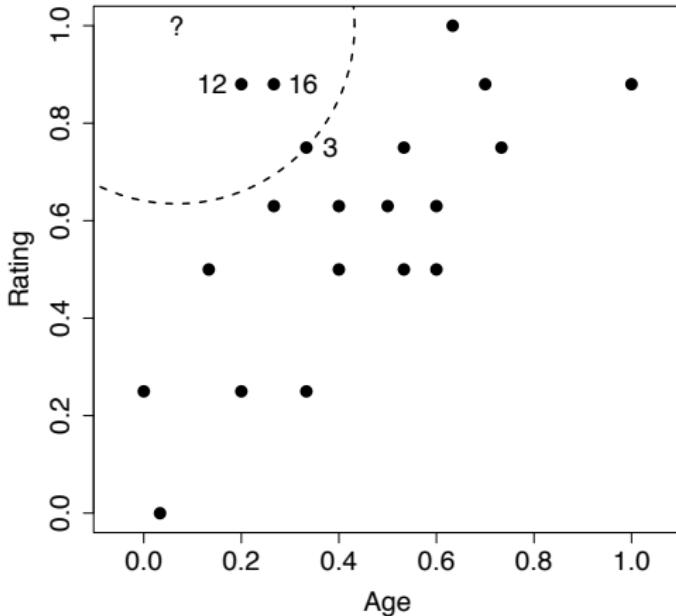


Figure: The AGE and RATING feature space for the whiskey dataset. The location of the query instance is indicated by the ? symbol. The circle plotted with a dashed line demarcates the border of the neighborhood around the query when $k = 3$. The three nearest neighbors to the query are labelled with their ID values.

- The model will return a price prediction that is the average price of the three neighbors:

$$\frac{200.00 + 250.00 + 55.00}{3} = 168.33$$

- In a **weighted k nearest neighbor** the model prediction equation is changed to:

$$\mathbb{M}_k(\mathbf{q}) = \frac{\sum_{i=1}^k \frac{1}{dist(\mathbf{q}, \mathbf{d}_i)^2} \times t_i}{\sum_{i=1}^k \frac{1}{dist(\mathbf{q}, \mathbf{d}_i)^2}} \quad (6)$$

Table: The calculations for the weighted k nearest neighbor prediction

ID	Price	Distance	Weight	Price × Weight
1	30.00	0.7530	1.7638	52.92
2	40.00	0.5017	3.9724	158.90
3	55.00	0.3655	7.4844	411.64
4	550.00	0.6456	2.3996	1319.78
5	35.00	0.6009	2.7692	96.92
6	45.00	0.5731	3.0450	137.03
7	70.00	0.5294	3.5679	249.75
8	85.00	0.7311	1.8711	159.04
9	78.00	0.6520	2.3526	183.50
10	75.00	0.6839	2.1378	160.33
11	500.00	0.5667	3.1142	1557.09
12	200.00	0.1828	29.9376	5987.53
13	65.00	0.4250	5.5363	359.86
14	120.00	0.7120	1.9726	236.71
15	12.00	0.7618	1.7233	20.68
16	250.00	0.2358	17.9775	4494.38
17	18.00	0.7960	1.5783	28.41
18	450.00	0.9417	1.1277	507.48
19	10.00	1.0006	0.9989	9.99
20	30.00	0.5044	3.9301	117.90
Totals:		99.2604	16249.85	

Other Measures of Similarity

Table: A binary dataset listing the behavior of two individuals on a website during a trial period and whether or not they subsequently signed-up for the website.

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No

Who is q more similar to d_1 or d_2 ?

$q = \langle \text{PROFILE}:1, \text{FAQ}:0, \text{HELP FORUM}:1, \text{NEWSLETTER}:0, \text{LIKED}:0, \rangle$

ID	Profile	FAQ	Help Forum	Newsletter	Liked	Signup
1	1	1	1	0	1	Yes
2	1	0	0	0	0	No

		q				q	
		Pres.	Abs.			Pres.	Abs.
d₁	Pres.	CP=2	PA=0	d₂	Pres.	CP=1	PA=1
	Abs.	AP=2	CA=1		Abs.	AP=0	CA=3

Table: The similarity between the current trial user, **q**, and the two users in the dataset, **d₁** and **d₂**, in terms of co-presence (CP), co-absence (CA), presence-absence (PA), and absence-presence (AP).

Russel-Rao

- The ratio between the number of co-presences and the total number of binary features considered.

$$sim_{RR}(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d})}{|\mathbf{q}|} \quad (7)$$

Russel-Rao

$$sim_{RR}(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d})}{|\mathbf{q}|} \quad (8)$$

`(PROFILE:1, FAQ:0, HELP FORUM:1, NEWSLETTER:0, LIKED:0,)`

		q				q			
		Pres.	Abs.			Pres.	Abs.		
d₁		Pres.	CP=2	PA=0	d₂		Pres.	CP=1	PA=1
		Abs.	AP=2	CA=1			Abs.	AP=0	CA=3

Russel-Rao

$$sim_{RR}(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d})}{|\mathbf{q}|} \quad (9)$$

Example

$$sim_{RR}(\mathbf{q}, \mathbf{d}_1) = \frac{2}{5} = 0.4$$

$$sim_{RR}(\mathbf{q}, \mathbf{d}_2) = \frac{1}{5} = 0.2$$

- The current trial user is judged to be more similar to instance \mathbf{d}_1 than \mathbf{d}_2 .

Sokal-Michener

- Sokal-Michener is defined as the ratio between the total number of co-presences and co-absences, and the total number of binary features considered.

$$sim_{SM}(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d}) + CA(\mathbf{q}, \mathbf{d})}{|\mathbf{q}|} \quad (10)$$

Sokal-Michener

$$sim_{SM}(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d}) + CA(\mathbf{q}, \mathbf{d})}{|\mathbf{q}|} \quad (11)$$

`(PROFILE:1, FAQ:0, HELP FORUM:1, NEWSLETTER:0, LIKED:0,)`

		q				q				
		Pres.	Abs.			Pres.	Abs.			
d₁		Pres.	CP=2 AP=2	PA=0	CA=1	d₂		Pres.	CP=1 AP=0	PA=1 CA=3

Sokal-Michener

$$sim_{SM}(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d}) + CA(\mathbf{q}, \mathbf{d})}{|\mathbf{q}|} \quad (12)$$

Example

$$sim_{SM}(\mathbf{q}, \mathbf{d}_1) = \frac{3}{5} = 0.6$$

$$sim_{SM}(\mathbf{q}, \mathbf{d}_2) = \frac{4}{5} = 0.8$$

- The current trial user is judged to be more similar to instance \mathbf{d}_2 than \mathbf{d}_1 .

Jaccard

- The Jaccard index ignore co-absences

$$sim_J(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d})}{CP(\mathbf{q}, \mathbf{d}) + PA(\mathbf{q}, \mathbf{d}) + AP(\mathbf{q}, \mathbf{d})} \quad (13)$$

Jaccard

$$sim_J(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d})}{CP(\mathbf{q}, \mathbf{d}) + PA(\mathbf{q}, \mathbf{d}) + AP(\mathbf{q}, \mathbf{d})} \quad (14)$$

`(PROFILE:1, FAQ:0, HELP FORUM:1, NEWSLETTER:0, LIKED:0,)`

		q				q	
		Pres.	Abs.			Pres.	Abs.
d₁		Pres.	CP=2	Pres.	CP=1		
Abs.		AP=2	CA=1	Abs.	PA=1		
				Abs.	AP=0		
					CA=3		

Jaccard

$$sim_J(\mathbf{q}, \mathbf{d}) = \frac{CP(\mathbf{q}, \mathbf{d})}{CP(\mathbf{q}, \mathbf{d}) + PA(\mathbf{q}, \mathbf{d}) + AP(\mathbf{q}, \mathbf{d})} \quad (15)$$

Example

$$sim_J(\mathbf{q}, \mathbf{d}_1) = \frac{2}{4} = 0.5$$

$$sim_J(\mathbf{q}, \mathbf{d}_2) = \frac{1}{2} = 0.5$$

- The current trial user is judged to be equally similar to instance \mathbf{d}_1 and \mathbf{d}_2 !

- **Cosine similarity** between two instances is the cosine of the inner angle between the two vectors that extend from the origin to each instance.

Cosine

$$\text{sim}_{\text{COSINE}}(\mathbf{a}, \mathbf{b}) = \frac{(\mathbf{a}[1] \times \mathbf{b}[1]) + \cdots + (\mathbf{a}[m] \times \mathbf{b}[m])}{\sqrt{\sum_{i=1}^m \mathbf{a}[i]^2} \times \sqrt{\sum_{i=1}^m \mathbf{b}[i]^2}}$$

- Calculate the cosine similarity between the following two instances:

$$\mathbf{d}_1 = \langle \text{SMS} = 97, \text{VOICE} = 21 \rangle$$

$$\mathbf{d}_2 = \langle \text{SMS} = 181, \text{VOICE} = 184 \rangle$$

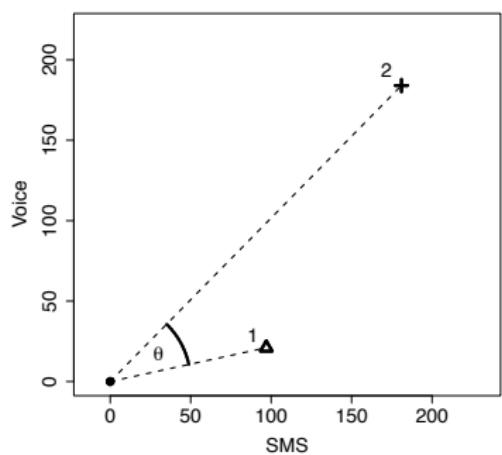
$$sim_{COSINE}(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^m (\mathbf{a}[i] \times \mathbf{b}[i])}{\sqrt{\sum_{i=1}^m \mathbf{a}[i]^2} \times \sqrt{\sum_{i=1}^m \mathbf{b}[i]^2}}$$

- Calculate the cosine similarity between the following two instances:

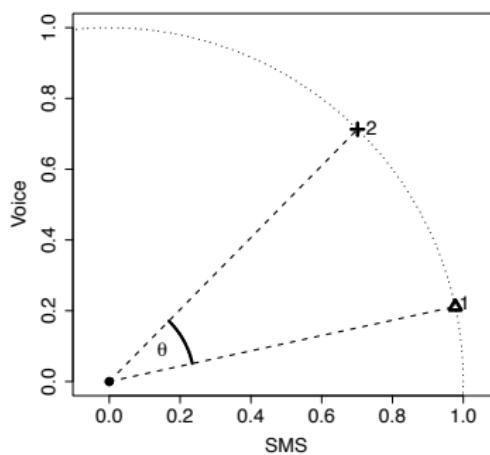
$$\mathbf{d}_1 = \langle \text{SMS} = 97, \text{VOICE} = 21 \rangle$$

$$\mathbf{d}_2 = \langle \text{SMS} = 181, \text{VOICE} = 184 \rangle$$

$$\begin{aligned} sim_{COSINE}(\mathbf{d}_1, \mathbf{d}_1) &= \frac{(97 \times 181) + (21 \times 184)}{\sqrt{97^2 + 21^2} \times \sqrt{181^2 + 184^2}} \\ &= 0.8362 \end{aligned}$$



(a)



(b)

Figure: (a) The θ represents the inner angle between the vector emanating from the origin to instance \mathbf{d}_1 (SMS = 97, VOICE = 21) and the vector emanating from the origin to instance \mathbf{d}_2 (SMS = 181, VOICE = 184); (b) shows \mathbf{d}_1 and \mathbf{d}_2 normalized to the unit circle.

- Calculate the cosine similarity between the following two instances:

$$\mathbf{d}_1 = \langle \text{SMS} = 97, \text{VOICE} = 21 \rangle$$

$$\mathbf{d}_3 = \langle \text{SMS} = 194, \text{VOICE} = 42 \rangle$$

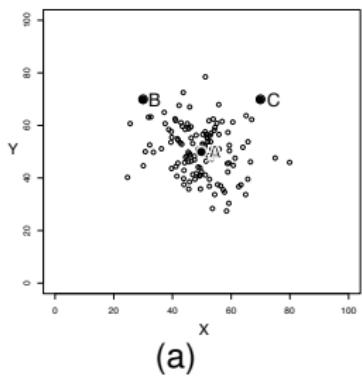
$$sim_{COSINE}(\mathbf{a}, \mathbf{b}) = \frac{\sum_{i=1}^m (\mathbf{a}[i] \times \mathbf{b}[i])}{\sqrt{\sum_{i=1}^m \mathbf{a}[i]^2} \times \sqrt{\sum_{i=1}^m \mathbf{b}[i]^2}}$$

- Calculate the cosine similarity between the following two instances:

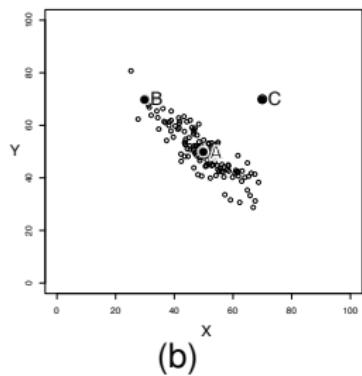
$$\mathbf{d}_1 = \langle \text{SMS} = 97, \text{VOICE} = 21 \rangle$$

$$\mathbf{d}_3 = \langle \text{SMS} = 194, \text{VOICE} = 42 \rangle$$

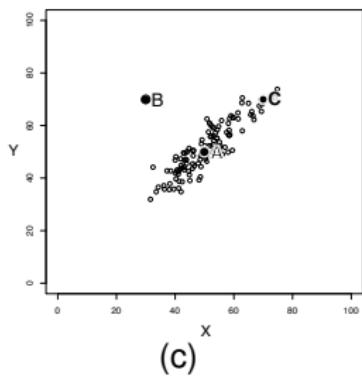
$$\begin{aligned} sim_{COSINE}(\mathbf{d}_1, \mathbf{d}_1) &= \frac{(97 \times 194) + (21 \times 42)}{\sqrt{97^2 + 21^2} \times \sqrt{194^2 + 42^2}} \\ &= 1 \end{aligned}$$



(a)



(b)



(c)

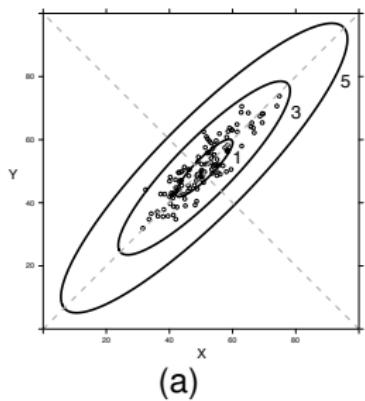
Figure: Scatter plots of three bivariate datasets with the same center point A and two queries B and C both equidistant from A. (a) A dataset uniformly spread around the center point. (b) A dataset with negative covariance. (c) A dataset with positive covariance.

- The mahalanobis distance uses covariance to scale distances so that distances along a direction where the dataset is spreadout a lot are scaled down and distances along directions where the dataset is tightly packed are scaled up.

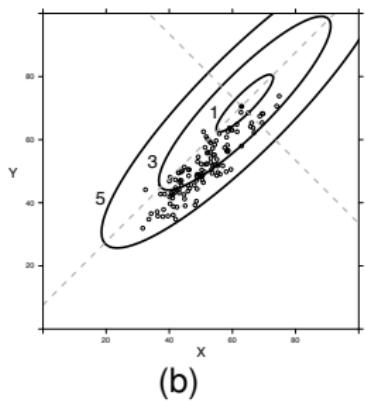
Mahalanobis(a, b) =

$$\sqrt{[\mathbf{a}[1] - \mathbf{b}[1], \dots, \mathbf{a}[m] - \mathbf{b}[m]] \times \sum^{-1} \times \begin{bmatrix} \mathbf{a}[1] - \mathbf{b}[1] \\ \dots \\ \mathbf{a}[m] - \mathbf{b}[m] \end{bmatrix}} \quad (16)$$

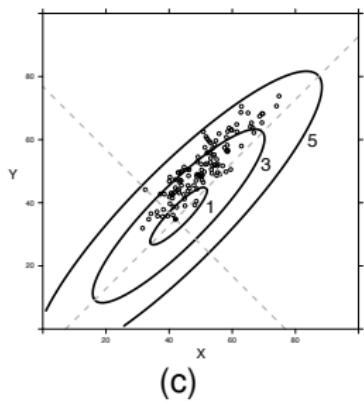
- Similar to Euclidean distance, the mahalanobis distance squares the differences of the features.
- But it also rescales the differences (using the inverse covariance matrix) so that all the features have unit variance and the effects of covariance is removed.



(a)



(b)



(c)

Figure: The coordinate systems defined by the mahalanobis metric using the co-variance matrix for the dataset in Figure 9(c) [46] using three different origins: (a) (50, 50), (b) (63, 71), (c) (42, 35). The ellipses in each figure plot the 1, 3, and 5 unit distances contours from each origin.

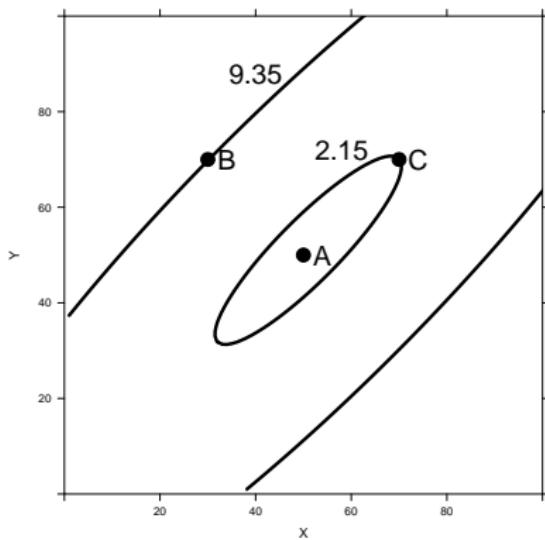


Figure: The effect of using a mahalanobis versus euclidean distance. Point A is the center of mass of the dataset in Figure 9(c) [46]. The ellipses plot the mahalanobis distance contours from A that B and C lie on. In euclidean terms B and C are equidistant from A, however using the mahalanobis metric C is much closer to A than B.

Feature Selection

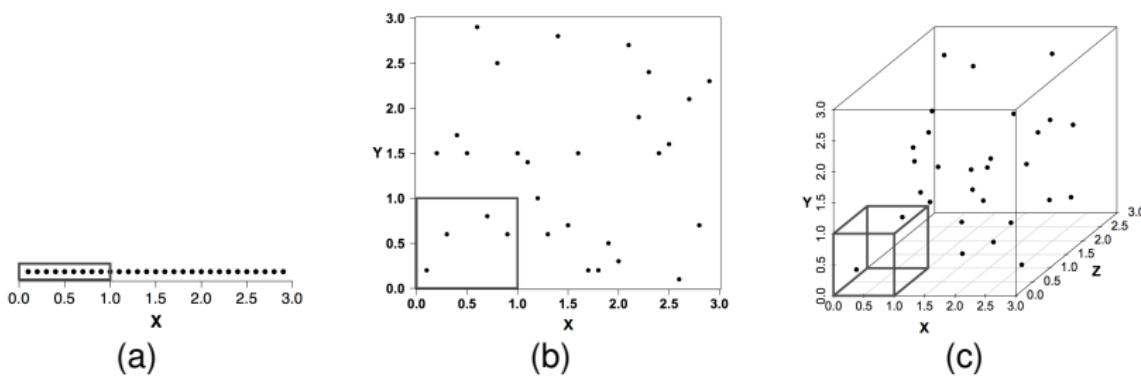


Figure: A set of scatter plots illustrating the curse of dimensionality. Across figures (a), (b) and (c) the density of the marked unit hypercubes decreases as the number of dimensions increases.

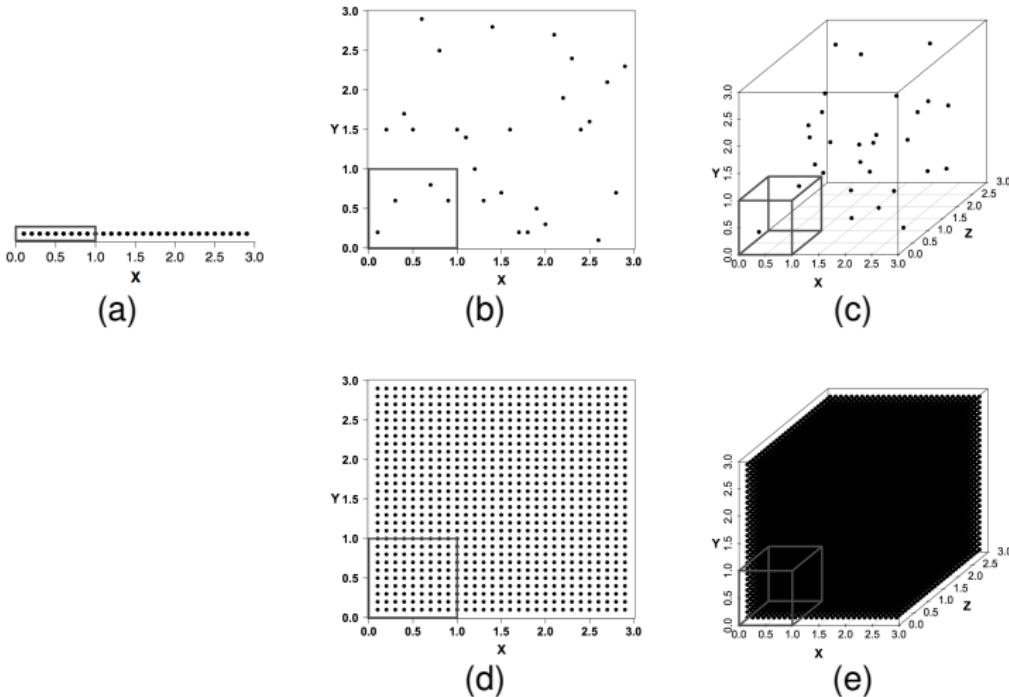


Figure: Figures (d) and (e) illustrate the cost we must incur if we wish to maintain the density of the instances in the feature space as the dimensionality of the feature space increases.

- During our discussion of feature selection approaches it will be useful to distinguish between different classes of descriptive features:
 - ➊ Predictive
 - ➋ Interacting
 - ➌ Redundant
 - ➍ Irrelevant

- When framed as a local search problem feature selection is defined in terms of an iterative process consisting of the following stages:
 - 1 Subset Generation
 - 2 Subset Selection
 - 3 Termination Condition
- The search can move through the search space in a number of ways:
 - Forward sequential selection
 - Backward sequential selection

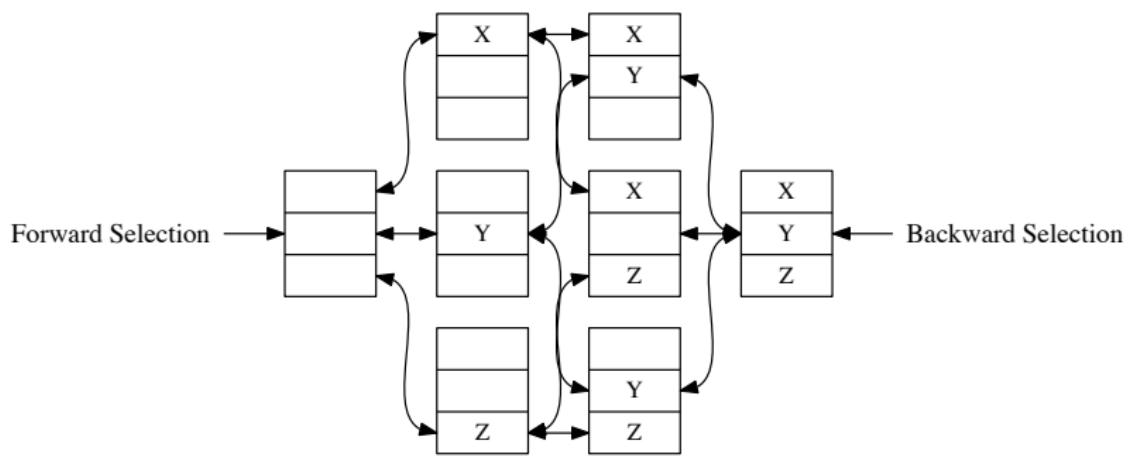
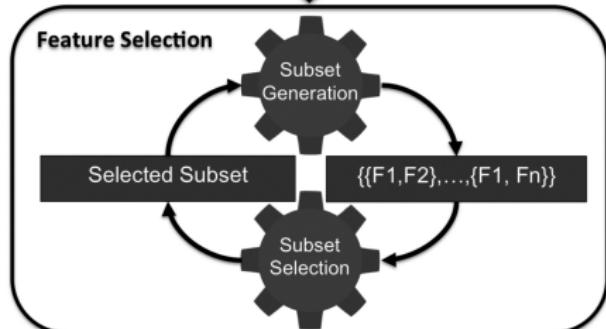


Figure: Feature Subset Space for a dataset with 3 features X, Y, Z .

ID	F1	F2	F3	...	F150	F151	...	F227	F228	...	F387	...	F_n	Target
1	-	-	-	...	-	-	...	-	-	...	-	...	-	-
2	-	-	-	...	-	-	...	-	-	...	-	...	-	-
...
K	-	-	-	...	-	-	...	-	-	...	-	...	-	-



ID	F2	F150	F227	F387	Target
1	-	-	-	-	-
2	-	-	-	-	-
...
K	-	-	-	-	-

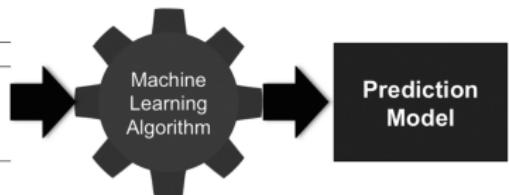


Figure: The process of model induction with feature selection.

Efficient Memory Search

- Assuming that the training set will remain relatively stable it is possible to speed up the prediction speed of a nearest neighbor model by investing in some one-off computation to create an index of the instances that enables efficient retrieval of the nearest neighbors.
- The **k-d tree**, which is short for k-dimensional tree, is one of the best known of these indexes.

Example

Let's build a k-d tree for the college athlete dataset

Table: The speed and agility ratings for 22 college athletes labelled with the decisions for whether they were drafted or not.

ID	Speed	Agility	Draft	ID	Speed	Agility	Draft
1	2.50	6.00	No	12	5.00	2.50	No
2	3.75	8.00	No	13	8.25	8.50	No
3	2.25	5.50	No	14	5.75	8.75	Yes
4	3.25	8.25	No	15	4.75	6.25	Yes
5	2.75	7.50	No	16	5.50	6.75	Yes
6	4.50	5.00	No	17	5.25	9.50	Yes
7	3.50	5.25	No	18	7.00	4.25	Yes
8	3.00	3.25	No	19	7.50	8.00	Yes
9	4.00	4.00	No	20	7.25	5.75	Yes
10	4.25	3.75	No	21	6.75	3.00	Yes
11	2.00	2.00	No				

First split on the SPEED feature

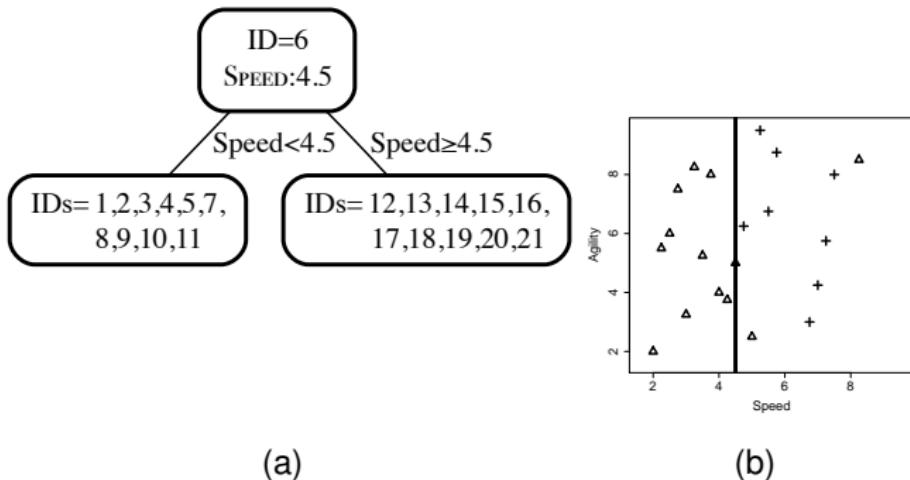


Figure: The k-d tree generated, for the dataset in Table 7 [60] after the initial split using the SPEED feature. (b) the partitioning of the feature space by the k-d tree in (a).

Next split on the AGILITY feature

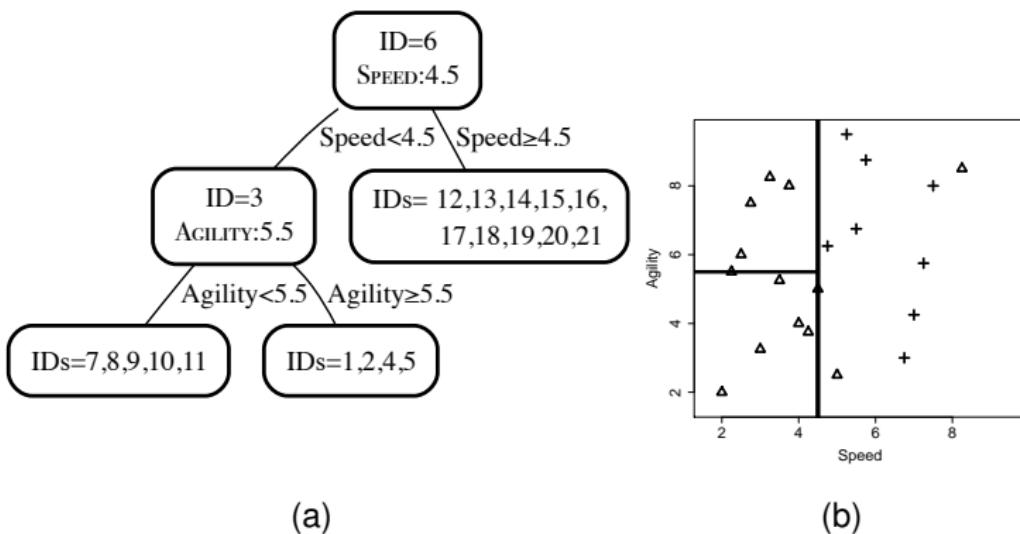
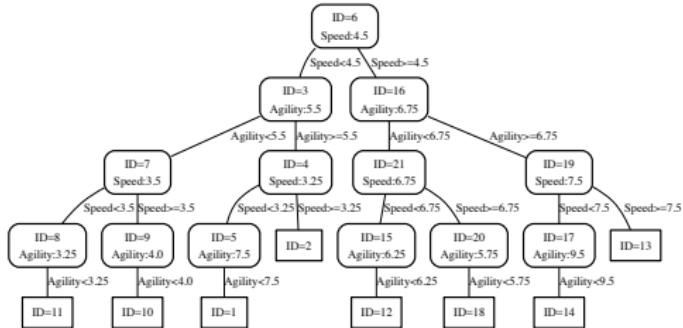
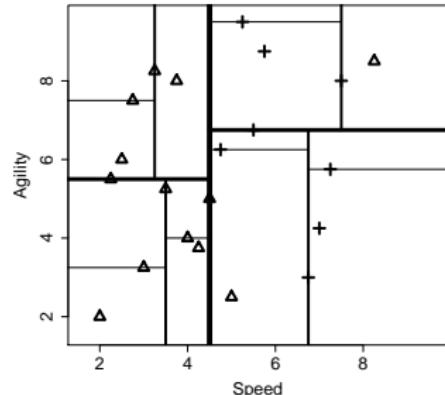


Figure: (a) the k-d tree after the dataset at the left child of the root has been split using the AGILITY feature with a threshold of 5.5. (b) the partitioning of the feature space by the k-d tree in (a)

After completing the tree building process



(a)



(b)

Figure: (a) The complete k-d tree generated, for the dataset in Table 7 [60]. (b) the partitioning of the feature space by the k-d tree in (a)

Finding neighbors for query **SPEED= 6, AGILITY= 3.5.**

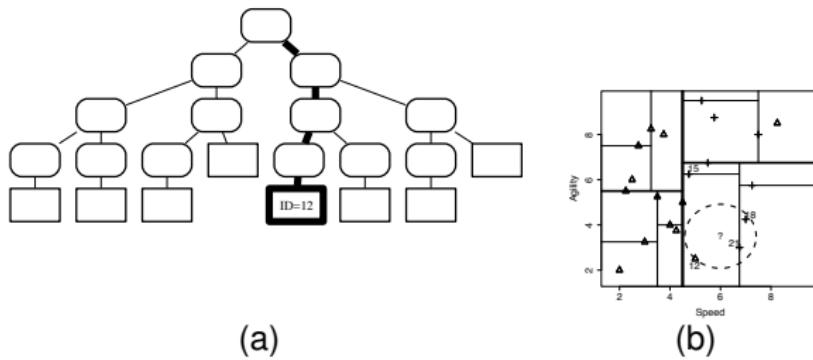
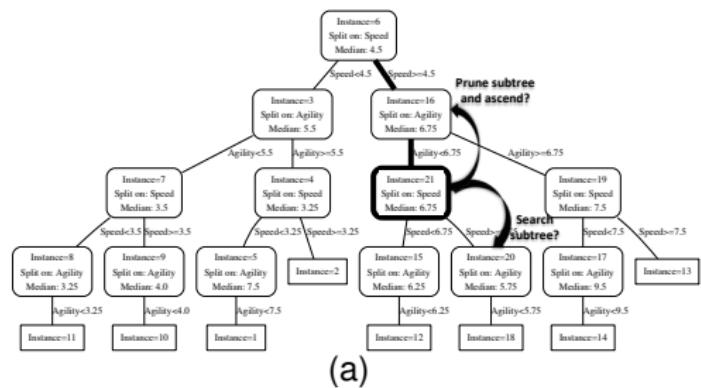
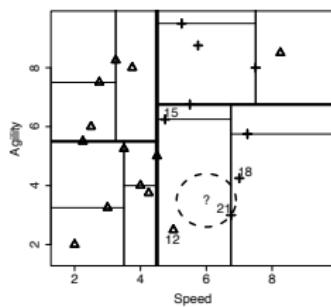


Figure: (a) the path taken from the root node to a leaf node when we search the tree. (b) the location of the query in the feature space is represented by the ? and the target hypersphere defining the region that must contain the true nearest neighbor.

Finding neighbors for query SPEED= 6, AGILITY= 3.5.



(a)



(b)

Figure: (a) the state of the retrieval process after instance d_{21} has been stored as *current-best*. (b) the dashed circle illustrates the extent of the target hypersphere after *current-best-distance* has been updated.

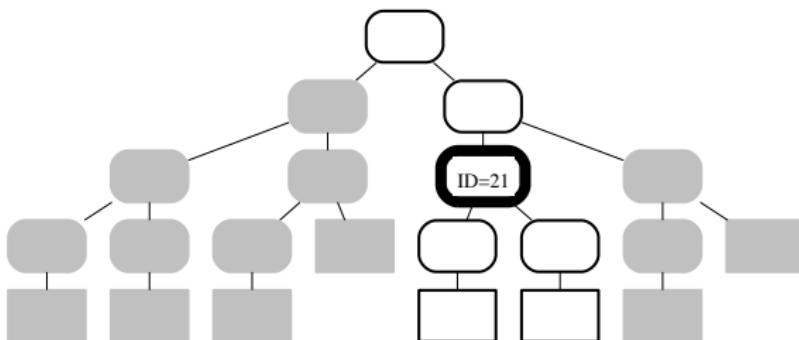


Figure: The greyed out branches indicate the portions of the k-d tree pruned from the search. The node with the bold outline represents the instance (d_{21}) that was returned as the nearest neighbor to the query.

Summary

- Nearest neighbor models are very sensitive to noise in the target feature the easiest way to solve this problem is to employ a ***k nearest neighbor***.
- **Normalization** techniques should almost always be applied when nearest neighbor models are used.
- It is easy to adapt a nearest neighbor model to **continuous targets**.
- There are many different measures of **similarity**.
- **Feature selection** is a particularly important process for nearest neighbor algorithms it alleviates the **curse of dimensionality**.
- As the number of instances becomes large, a nearest neighbor model will become slower—techniques such as the ***k-d tree*** can help with this issue.

- 1 Handling Noisy Data**
- 2 Data Normalization**
- 3 Predicting Continuous Targets**
- 4 Other Measures of Similarity**
- 5 Feature Selection**
- 6 Efficient Memory Search**
- 7 Summary**

Fundamentals of Machine Learning for Predictive Data Analytics

Appendix B Introduction to Probability for Machine Learning

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

1 Probability Basics

2 Probability Distributions and Summing Out

3 Some Useful Probability Rules

4 Summary

- Probability is the branch of mathematics that deals with measuring the likelihood (or uncertainty) around events.
- There are two ways of computing the likelihood of a future event:
 - 1 use the **relative frequency** of the event in the past,
 - 2 use a **subjective** estimate (ideally from an expert!)
- We will focus on using relative frequency.

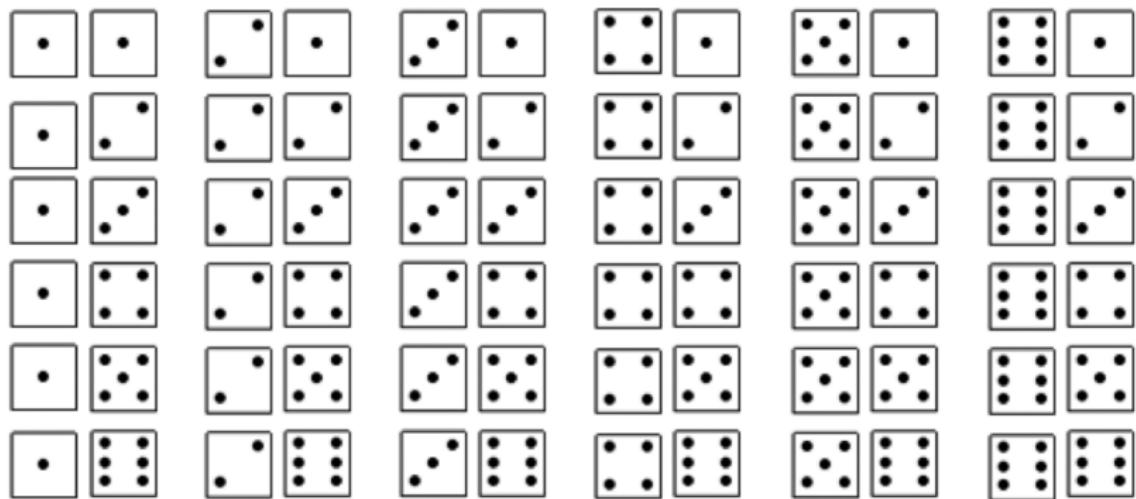


Figure: The **sample space** for the domain of two dice.

ID	Die1	Die2
1	3	4
2	1	5
3	6	5
4	3	3
5	1	1

Table: A dataset of instances from the sample space in Figure 1 [4].

- Throughout our discussions about probability we will be talking about **events** so it is very important that you understand this simple concept.

Events

- an **event** defines an assignment of values to the features in the domain; these assignments may define values for all the features in the domain (e.g. a full row in the dataset) or just to one or more features in the domain,

Example

- DIE1='3',
- DIE1='1', DIE2='5'.

Probability Functions: $P()$

- A feature can take one or more values from a domain and we can find out the likelihood of a feature taking any particular value using a **probability function** $P()$.
- A probability function is a function that takes an event (an assignment of values to features) as a parameter and returns the likelihood of that event.

Example

- $P(\text{DIE1} = '3')$ will return the likelihood of the event $\text{DIE1} = '3'$
- $P(\text{DIE1} = '3', \text{DIE2} = '4')$ will return the likelihood of the event where $\text{DIE1} = '3'$ and $\text{DIE2} = '4'$.

Properties of Probability Functions

$$0 \leq P(f = level) \leq 1$$

$$\sum_i P(f = level_i) = 1.0$$

- Probability functions are very easy to create when you have a dataset.
- The value returned by a probability function for an event is simply the **relative frequency** of that event in the dataset – in other words, how often the event happened divided by how often could it have happened.

Example

- The relative frequency of the event $\text{DIE1} = '3'$ is simply the count of all the rows in the dataset where the feature is assigned the relevant value divided by the number of rows in the dataset

Prior Probability (aka. Unconditional Probabilities)

- The probability of an event without any contextual information.
- The count of all the rows in the dataset where the feature(s) is assigned the relevant value(s) divided by the number of rows in the dataset.

Example

$$P(DIE1 = '3') = \frac{|\{\mathbf{d}_1, \mathbf{d}_4\}|}{|\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5\}|} = \frac{2}{5} = 0.4$$

Joint Probability

- The probability of two or more events happening together.
- The number of rows in the dataset where the set of assignments listed in the joint event holds divided by the total number of rows in the dataset.

Example

$$P(\text{DIE1} = '6', \text{DIE2} = '5') = \frac{|\{\mathbf{d}_3\}|}{|\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5\}|} = \frac{1}{5} = 0.2$$

Posterior Probabilities (aka. Conditional Probabilities)

- The probability of an event in a context where one or more events are known to have happened.
- The vertical bar symbol | can be read as *given that*.
- The number of rows in the dataset where both events are true divided by the number of rows in the dataset where just the given event is true.

Example

$$P(\text{DIE1} = '6' \mid \text{DIE2} = '5') = \frac{|\{\mathbf{d}_3\}|}{|\{\mathbf{d}_2, \mathbf{d}_3\}|} = \frac{1}{2} = 0.5$$

Table: A simple dataset for MENINGITIS with three common symptoms of the disease listed as descriptive features: HEADACHE, FEVER and VOMITING.

ID	Headache	Fever	Vomiting	Meningitis
11	True	True	False	False
37	False	True	False	False
42	True	False	True	False
49	True	False	True	False
54	False	True	False	True
57	True	False	True	False
73	True	False	True	False
75	True	False	True	True
89	False	True	False	False
92	True	False	True	True

Your Turn!

ID	Headach	Fever	Vomit	Meningitis
11	True	True	False	False
37	False	True	False	False
42	True	False	True	False
49	True	False	True	False
54	False	True	False	True
57	True	False	True	False
73	True	False	True	False
75	True	False	True	True
89	False	True	False	False
92	True	False	True	True

- $P(h) = ?$
- $P(m|h) = ?$
- $P(m, h) = ?$

Your Turn!

$$P(h) = \frac{|\{\mathbf{d}_{11}, \mathbf{d}_{42}, \mathbf{d}_{49}, \mathbf{d}_{57}, \mathbf{d}_{73}, \mathbf{d}_{75}, \mathbf{d}_{92}\}|}{|\{\mathbf{d}_{11}, \mathbf{d}_{37}, \mathbf{d}_{42}, \mathbf{d}_{49}, \mathbf{d}_{54}, \mathbf{d}_{57}, \mathbf{d}_{73}, \mathbf{d}_{75}, \mathbf{d}_{89}, \mathbf{d}_{92}\}|} = \frac{7}{10} = 0.7$$

$$P(m|h) = \frac{|\{\mathbf{d}_{75}, \mathbf{d}_{92}\}|}{|\{\mathbf{d}_{11}, \mathbf{d}_{42}, \mathbf{d}_{49}, \mathbf{d}_{57}, \mathbf{d}_{73}, \mathbf{d}_{75}, \mathbf{d}_{92}\}|} = \frac{2}{7} = 0.2857$$

$$P(m, h) = \frac{|\{\mathbf{d}_{75}, \mathbf{d}_{92}\}|}{|\{\mathbf{d}_{11}, \mathbf{d}_{37}, \mathbf{d}_{42}, \mathbf{d}_{49}, \mathbf{d}_{54}, \mathbf{d}_{57}, \mathbf{d}_{73}, \mathbf{d}_{75}, \mathbf{d}_{89}, \mathbf{d}_{92}\}|} = \frac{2}{10} = 0.2$$

Probability Distributions

- A probability distribution is a data structure that describes for all the values in the domain of a feature the probability of the feature taking that value.
- A probability distribution of a categorical feature is a vector that lists the probabilities associated with the values in the domain of the feature.
- We use bold notation $\mathbf{P}()$ to distinguish when we are talking about a probability distribution from a probability mass function $P()$.

Example

- Based on the following dataset the probability distribution for the binary feature, Meningitis, using the convention of the first element in the vector being the probability for 'True', would be:

ID	Headach	Fever	Vomit	Meningitis
11	True	True	False	False
37	False	True	False	False
42	True	False	True	False
49	True	False	True	False
54	False	True	False	True
57	True	False	True	False
73	True	False	True	False
75	True	False	True	True
89	False	True	False	False
92	True	False	True	True

$$\mathbf{P}(M) = < 0.3, 0.7 >$$

Joint Probability Distributions

- is a multi-dimensional matrix where each cell in the matrix lists the probability for one of the events in the sample space defined by the combination of feature values.

Example

- The joint probability distribution for the four binary features HEADING, FEVER, VOMITING, MENINGITIS in the Meningitis domain would be:

$$\mathbf{P}(H, F, V, M) = \begin{bmatrix} P(h, f, v, m), & P(\neg h, f, v, m) \\ P(h, f, v, \neg m), & P(\neg h, f, v, \neg m) \\ P(h, f, \neg v, m), & P(\neg h, f, \neg v, m) \\ P(h, f, \neg v, \neg m), & P(\neg h, f, \neg v, \neg m) \\ P(h, \neg f, v, m), & P(\neg h, \neg f, v, m) \\ P(h, \neg f, v, \neg m), & P(\neg h, \neg f, v, \neg m) \\ P(h, \neg f, \neg v, m), & P(\neg h, \neg f, \neg v, m) \\ P(h, \neg f, \neg v, \neg m), & P(\neg h, \neg f, \neg v, \neg m) \end{bmatrix}$$

- A **full joint probability distribution** is simply a joint probability distribution over all the features in a domain.

Summing out (aka Marginalisation)

- Given a full joint probability we can compute the probability of any event in the domain by summing over the cells in the joint probability where that event is true.

Example

- Imagine we want to compute the probability of $P(h)$ in the domain specified by the joint probability distribution $\mathbf{P}(H, F, V, M)$.
- Simply sum the values in the cells containing h , in other words the cells in the first column in the full joint probability.

$$\mathbf{P}(H, F, V, M) = \begin{bmatrix} P(h, f, v, m), & P(\neg h, f, v, m) \\ P(h, f, v, \neg m), & P(\neg h, f, v, \neg m) \\ P(h, f, \neg v, m), & P(\neg h, f, \neg v, m) \\ P(h, f, \neg v, \neg m), & P(\neg h, f, \neg v, \neg m) \\ P(h, \neg f, v, m), & P(\neg h, \neg f, v, m) \\ P(h, \neg f, v, \neg m), & P(\neg h, \neg f, v, \neg m) \\ P(h, \neg f, \neg v, m), & P(\neg h, \neg f, \neg v, m) \\ P(h, \neg f, \neg v, \neg m), & P(\neg h, \neg f, \neg v, \neg m) \end{bmatrix}$$

- We can also use summing out to compute joint probabilities from a joint probability distribution.

Example

- Imagine we wish to calculate the probability of h and f when we don't care what value V and M take (here, V and M are examples of a **hidden feature**; a feature whose value is not specified as part of the evidence and which is not a target feature).

Example

- To calculate $P(h, V = ?, M = ?, f)$ from $\mathbf{P}(H, V, F, M)$ by summing the values in all the cells where h and f are the case (in other words summing the top four cells in column one).

$$\mathbf{P}(H, F, V, M) = \begin{bmatrix} P(h, f, v, m), & P(\neg h, f, v, m) \\ P(h, f, v, \neg m), & P(\neg h, f, v, \neg m) \\ P(h, f, \neg v, m), & P(\neg h, f, \neg v, m) \\ P(h, f, \neg v, \neg m), & P(\neg h, f, \neg v, \neg m) \\ P(h, \neg f, v, m), & P(\neg h, \neg f, v, m) \\ P(h, \neg f, v, \neg m), & P(\neg h, \neg f, v, \neg m) \\ P(h, \neg f, \neg v, m), & P(\neg h, \neg f, \neg v, m) \\ P(h, \neg f, \neg v, \neg m), & P(\neg h, \neg f, \neg v, \neg m) \end{bmatrix}$$

Conditional Probability

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \quad (1)$$

- Use this rule to recalculate the probability of $P(m|h)$ (recall that $P(h) = 0.7$ and $P(m, h) = 0.2$)

Conditional Probability

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \quad (1)$$

- Use this rule to recalculate the probability of $P(m|h)$ (recall that $P(h) = 0.7$ and $P(m, h) = 0.2$)

Example

$$P(m|h) = \frac{P(m, h)}{P(h)} = \frac{0.2}{0.7} = 0.2857$$

Product Rule

$$P(X, Y) = P(X|Y) \times P(Y)$$

- Note: $P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X)$

Example

- Use the product rule to recalculate $P(m, h)$.

Product Rule

$$P(X, Y) = P(X|Y) \times P(Y)$$

- Note: $P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X)$

Example

- Use the product rule to recalculate $P(m, h)$.

$$P(m, h) = P(m|h) \times P(h) = 0.2857 \times 0.7 = 0.2$$

Chain Rule

- The Product Rule:

$$P(X, Y) = P(X|Y) \times P(Y)$$

- generalizes to the Chain Rule:

$$\begin{aligned} P(A, B, C, \dots, Z) &= P(Z) \times P(Y|Z) \times P(X|Y, Z) \times \dots \\ &\quad \times P(A|B, \dots, X, Y, Z) \end{aligned}$$

Theorem of Total Probability

$$P(X) = \sum_{i=1}^k P(X|Y_i)P(Y_i)$$

Example

- Use the Theorem of Total Probability to recalculate $P(h)$ by summing out M .

$$P(h) = (P(h|m) \times P(m)) + (P(h|\neg m) \times P(\neg m))$$

ID	Headach	Fever	Vomit	Meningitis
11	True	True	False	False
37	False	True	False	False
42	True	False	True	False
49	True	False	True	False
54	False	True	False	True
57	True	False	True	False
73	True	False	True	False
75	True	False	True	True
89	False	True	False	False
92	True	False	True	True

- $P(h|m) = ?$
- $P(m) = ?$
- $P(h|\neg m) = ?$
- $P(\neg m) = ?$

ID	Headach	Fever	Vomit	Meningitis
11	True	True	False	False
37	False	True	False	False
42	True	False	True	False
49	True	False	True	False
54	False	True	False	True
57	True	False	True	False
73	True	False	True	False
75	True	False	True	True
89	False	True	False	False
92	True	False	True	True

- $P(h|m) = 0.6666$
- $P(m) = 0.3$
- $P(h|\neg m) = 0.7143$
- $P(\neg m) = 0.7$

$$\begin{aligned} P(h) &= (P(h|m) \times P(m)) + (P(h|\neg m) \times P(\neg m)) \\ &=? \end{aligned}$$

$$\begin{aligned}P(h) &= (P(h|m) \times P(m)) + (P(h|\neg m) \times P(\neg m)) \\&= (0.6666 \times 0.3) + (0.7143 \times 0.7) = 0.7\end{aligned}$$

- We can if we wish sum out more than one feature.

Example

- For example, we could compute $P(h)$ by summing out all the other features in the dataset:

$$P(h) = \sum_{i \in \text{level}(M)} \sum_{j \in \text{level}(Fev)} \sum_{k \in \text{level}(V)} P(h|M_i, Fev_j, V_k) \times P(M_i, Fev_j, V_k)$$

Summary

1 Probability Basics

2 Probability Distributions and Summing Out

3 Some Useful Probability Rules

4 Summary

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 6: Probability-based Learning
Sections 6.1, 6.2, 6.3

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

1 Big Idea

2 Fundamentals

- Bayes' Theorem
- Bayesian Prediction
- Conditional Independence and Factorization

3 Standard Approach: The Naive Bayes' Classifier

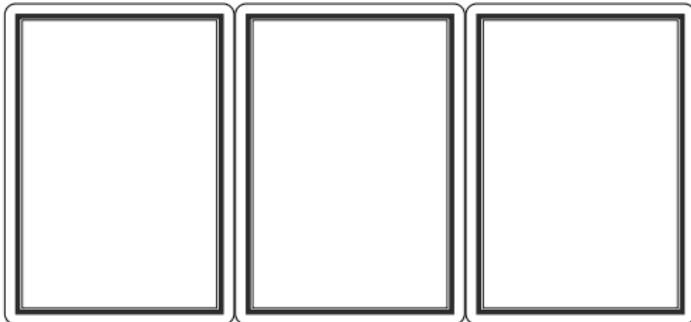
- A Worked Example

4 Summary

Big Idea

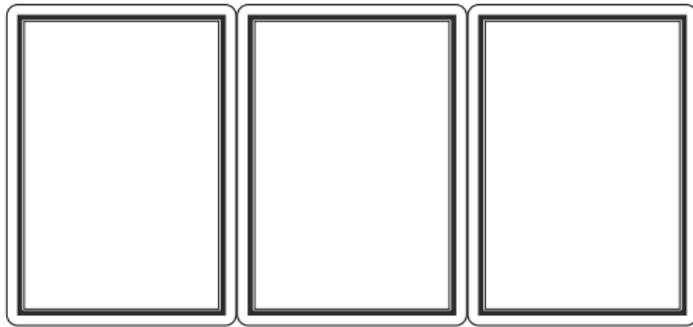


(a)

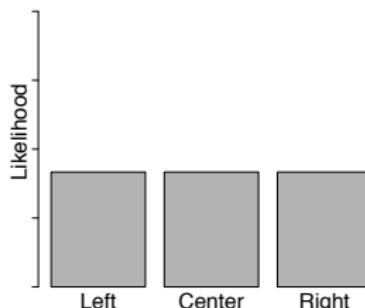


(b)

Figure: A game of *find the lady*

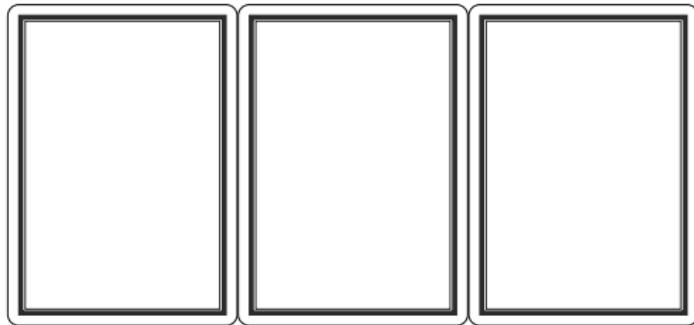


(a)

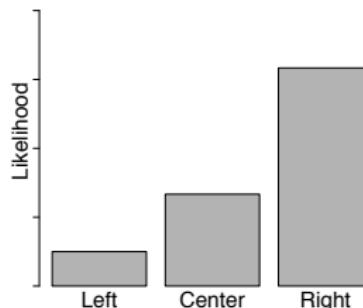


(b)

Figure: A game of *find the lady*: (a) the cards dealt face down on a table; and (b) the initial likelihoods of the queen ending up in each position.

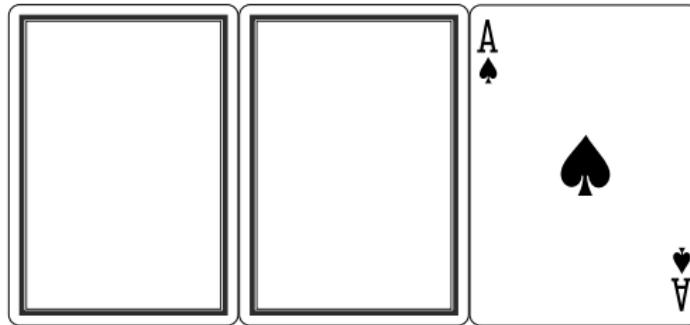


(a)

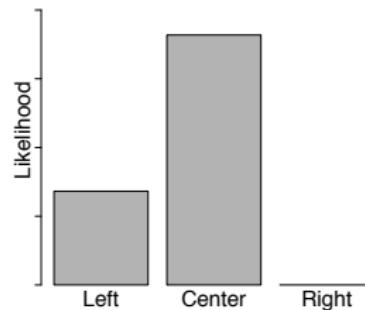


(b)

Figure: A game of *find the lady*: (a) the cards dealt face down on a table; and (b) a revised set of likelihoods for the position of the queen based on evidence collected.



(a)



(b)

Figure: A game of *find the lady*: (a) The set of cards after the wind blows over the one on the right; (b) the revised likelihoods for the position of the queen based on this new evidence.



Figure: A game of *find the lady*: The final positions of the cards in the game.



Big Idea

- We can use estimates of likelihoods to determine the most likely prediction that should be made.
- More importantly, we revise these predictions based on data we collect and whenever extra evidence becomes available.



Fundamentals

Table: A simple dataset for MENINGITIS diagnosis with descriptive features that describe the presence or absence of three common symptoms of the disease: HEADACHE, FEVER, and VOMITING.

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true



- A **probability function**, $P()$, returns the probability of a feature taking a specific value.
- A **joint probability** refers to the probability of an assignment of specific values to multiple different features.
- A **conditional probability** refers to the probability of one feature taking a specific value given that we already know the value of a different feature
- A **probability distribution** is a data structure that describes the probability of each possible value a feature can take. The sum of a probability distribution must equal 1.0.



- A **joint probability distribution** is a probability distribution over more than one feature assignment and is written as a multi-dimensional matrix in which each cell lists the probability of a particular combination of feature values being assigned.
- The sum of all the cells in a joint probability distribution must be 1.0.

$$\mathbf{P}(H, F, V, M) = \begin{bmatrix} P(h, f, v, m), & P(\neg h, f, v, m) \\ P(h, f, v, \neg m), & P(\neg h, f, v, \neg m) \\ P(h, f, \neg v, m), & P(\neg h, f, \neg v, m) \\ P(h, f, \neg v, \neg m), & P(\neg h, f, \neg v, \neg m) \\ P(h, \neg f, v, m), & P(\neg h, \neg f, v, m) \\ P(h, \neg f, v, \neg m), & P(\neg h, \neg f, v, \neg m) \\ P(h, \neg f, \neg v, m), & P(\neg h, \neg f, \neg v, m) \\ P(h, \neg f, \neg v, \neg m), & P(\neg h, \neg f, \neg v, \neg m) \end{bmatrix}$$



- Given a joint probability distribution, we can compute the probability of any event in the domain that it covers by summing over the cells in the distribution where that event is true.
- Calculating probabilities in this way is known as **summing out**.



Bayes' Theorem

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$



Example

After a yearly checkup, a doctor informs their patient that he has both bad news and good news. The bad news is that the patient has tested positive for a serious disease and that the test that the doctor has used is 99% accurate (i.e., the probability of testing positive when a patient has the disease is 0.99, as is the probability of testing negative when a patient does not have the disease). The good news, however, is that the disease is extremely rare, striking only 1 in 10,000 people.

- What is the actual probability that the patient has the disease?
- Why is the rarity of the disease good news given that the patient has tested positive for it?



Bayes' Theorem

$$P(d|t) = \frac{P(t|d)P(d)}{P(t)}$$

$$\begin{aligned} P(t) &= P(t|d)P(d) + P(t|\neg d)P(\neg d) \\ &= (0.99 \times 0.0001) + (0.01 \times 0.9999) = 0.0101 \end{aligned}$$

$$\begin{aligned} P(d|t) &= \frac{0.99 \times 0.0001}{0.0101} \\ &= 0.0098 \end{aligned}$$

Deriving Bayes theorem

$$P(Y|X)P(X) = P(X|Y)P(Y)$$

$$\frac{P(X|Y)P(Y)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$$

$$\frac{P(X|Y)P(Y)}{P(Y)} = \frac{P(Y|X)P(X)}{P(Y)}$$

$$\Rightarrow P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

Bayes' Theorem

- The divisor is the prior probability of the evidence
- This division functions as a normalization constant.

$$0 \leq P(X|Y) \leq 1$$

$$\sum_i P(X_i|Y) = 1.0$$



- We can calculate this divisor directly from the dataset.

$$P(Y) = \frac{|\{\text{rows where } Y \text{ is the case}\}|}{|\{\text{rows in the dataset}\}|}$$

- Or, we can use the **Theorem of Total Probability** to calculate this divisor.

$$P(Y) = \sum_i P(Y|X_i)P(X_i) \tag{1}$$



Generalized Bayes' Theorem

$$P(t = l | \mathbf{q}[1], \dots, \mathbf{q}[m]) = \frac{P(\mathbf{q}[1], \dots, \mathbf{q}[m] | t = l) P(t = l)}{P(\mathbf{q}[1], \dots, \mathbf{q}[m])}$$

Chain Rule

$$\begin{aligned} P(\mathbf{q}[1], \dots, \mathbf{q}[m]) &= \\ P(\mathbf{q}[1]) \times P(\mathbf{q}[2]|\mathbf{q}[1]) \times \\ \cdots \times P(\mathbf{q}[m]|\mathbf{q}[m-1], \dots, \mathbf{q}[2], \mathbf{q}[1]) \end{aligned}$$

- To apply the chain rule to a conditional probability we just add the conditioning term to each term in the expression:

$$\begin{aligned} P(\mathbf{q}[1], \dots, \mathbf{q}[m] | \mathbf{t} = \mathcal{I}) &= \\ P(\mathbf{q}[1] | \mathbf{t} = \mathcal{I}) \times P(\mathbf{q}[2] | \mathbf{q}[1], \mathbf{t} = \mathcal{I}) \times \dots \\ \cdots \times P(\mathbf{q}[m] | \mathbf{q}[m-1], \dots, \mathbf{q}[3], \mathbf{q}[2], \mathbf{q}[1], \mathbf{t} = \mathcal{I}) \end{aligned}$$

Bayesian Prediction

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

HEADACHE	FEVER	VOMITING	MENINGITIS
true	false	true	?

Bayesian Prediction

$$P(M|h, \neg f, v) = ?$$

- In the terms of Bayes' Theorem this problem can be stated as:

$$P(M|h, \neg f, v) = \frac{P(h, \neg f, v|M) \times P(M)}{P(h, \neg f, v)}$$

- There are two values in the domain of the MENINGITIS feature, '*true*' and '*false*', so we have to do this calculation twice.

Bayesian Prediction

- We will do the calculation for m first
- To carry out this calculation we need to know the following probabilities: $P(m)$, $P(h, \neg f, v)$ and $P(h, \neg f, v | m)$.

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

Bayesian Prediction

- We can calculate the required probabilities directly from the data. For example, we can calculate $P(m)$ and $P(h, \neg f, v)$ as follows:

$$P(m) = \frac{|\{\mathbf{d}_5, \mathbf{d}_8, \mathbf{d}_{10}\}|}{|\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_9, \mathbf{d}_{10}\}|} = \frac{3}{10} = 0.3$$

$$P(h, \neg f, v) = \frac{|\{\mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_6, \mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_{10}\}|}{|\{\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6, \mathbf{d}_7, \mathbf{d}_8, \mathbf{d}_9, \mathbf{d}_{10}\}|} = \frac{6}{10} = 0.6$$



- However, as an exercise we will use the chain rule calculate:

$$P(h, \neg f, v \mid m) = ?$$

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

- Using the chain rule calculate:

$$\begin{aligned}P(h, \neg f, v | m) &= P(h | m) \times P(\neg f | h, m) \times P(v | \neg f, h, m) \\&= \frac{|\{\mathbf{d}_8, \mathbf{d}_{10}\}|}{|\{\mathbf{d}_5, \mathbf{d}_8, \mathbf{d}_{10}\}|} \times \frac{|\{\mathbf{d}_8, \mathbf{d}_{10}\}|}{|\{\mathbf{d}_8, \mathbf{d}_{10}\}|} \times \frac{|\{\mathbf{d}_8, \mathbf{d}_{10}\}|}{|\{\mathbf{d}_8, \mathbf{d}_{10}\}|} \\&= \frac{2}{3} \times \frac{2}{2} \times \frac{2}{2} = 0.6666\end{aligned}$$

Bayesian Prediction

- So the calculation of $P(m|h, \neg f, v)$ is:

$$\begin{aligned} P(m|h, \neg f, v) &= \frac{\left(P(h|m) \times P(\neg f|h, m) \right.}{P(h, \neg f, v)} \\ &\quad \left. \times P(v|\neg f, h, m) \times P(m) \right) \\ &= \frac{0.6666 \times 0.3}{0.6} = 0.3333 \end{aligned}$$

- The corresponding calculation for $P(\neg m | h, \neg f, v)$ is:

$$\begin{aligned} P(\neg m | h, \neg f, v) &= \frac{P(h, \neg f, v | \neg m) \times P(\neg m)}{P(h, \neg f, v)} \\ &= \frac{\left(P(h | \neg m) \times P(\neg f | h, \neg m) \right.}{\left. \times P(v | \neg f, h, \neg m) \times P(\neg m) \right)} \\ &= \frac{0.7143 \times 0.8 \times 1.0 \times 0.7}{0.6} = 0.6667 \end{aligned}$$

$$P(m|h, \neg f, v) = 0.3333$$

$$P(\neg m|h, \neg f, v) = 0.6667$$

- These calculations tell us that it is twice as probable that the patient does not have meningitis than it is that they do even though the patient is suffering from a headache and is vomiting!

The Paradox of the False Positive

- The mistake of forgetting to factor in the prior gives rise to the **paradox of the false positive** which states that in order to make predictions about a rare event the model has to be as accurate as the prior of the event is rare or there is a significant chance of **false positives** predictions (i.e., predicting the event when it is not the case).



Bayesian Prediction

Bayesian MAP Prediction Model

$$\begin{aligned}\mathbb{M}_{MAP}(\mathbf{q}) &= \arg \max_{I \in levels(t)} P(t = I \mid \mathbf{q}[1], \dots, \mathbf{q}[m]) \\ &= \arg \max_{I \in levels(t)} \frac{P(\mathbf{q}[1], \dots, \mathbf{q}[m] \mid t = I) \times P(t = I)}{P(\mathbf{q}[1], \dots, \mathbf{q}[m])}\end{aligned}$$

Bayesian MAP Prediction Model (without normalization)

$$\mathbb{M}_{MAP}(\mathbf{q}) = \arg \max_{I \in levels(t)} P(\mathbf{q}[1], \dots, \mathbf{q}[m] \mid t = I) \times P(t = I)$$

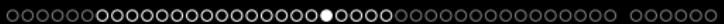
ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

HEADACHE	FEVER	VOMITING	MENINGITIS
true	true	false	?

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

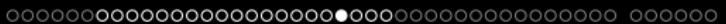
$$P(m \mid h, f, \neg v) = ?$$

$$P(\neg m \mid h, f, \neg v) = ?$$



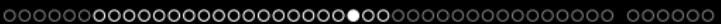
Bayesian Prediction

$$\begin{aligned} P(m \mid h, f, \neg v) &= \frac{\left(P(h|m) \times P(f \mid h, m) \right.}{P(h, f, \neg v)} \\ &\quad \left. \times P(\neg v \mid f, h, m) \times P(m) \right) \\ &= \frac{0.6666 \times 0 \times 0 \times 0.3}{0.1} = 0 \end{aligned}$$



Bayesian Prediction

$$\begin{aligned} P(\neg m \mid h, f, \neg v) &= \frac{\left(P(h \mid \neg m) \times P(f \mid h, \neg m) \right.}{P(h, f, \neg v)} \\ &\quad \left. \times P(\neg v \mid f, h, \neg m) \times P(\neg m) \right) \\ &= \frac{0.7143 \times 0.2 \times 1.0 \times 0.7}{0.1} = 1.0 \end{aligned}$$



Bayesian Prediction

$$P(m \mid h, f, \neg v) = 0$$

$$P(\neg m \mid h, f, \neg v) = 1.0$$

- There is something odd about these results!



Curse of Dimensionality

As the number of descriptive features grows the number of potential conditioning events grows. Consequently, an exponential increase is required in the size of the dataset as each new descriptive feature is added to ensure that for any conditional probability there are enough instances in the training dataset matching the conditions so that the resulting probability is reasonable.

- The probability of a patient who has a headache and a fever having meningitis should be greater than zero!
- Our dataset is not large enough → our model is **over-fitting** to the training data.
- The concepts of **conditional independence** and **factorization** can help us overcome this flaw of our current approach.



Conditional Independence and Factorization

- If knowledge of one event has no effect on the probability of another event, and *vice versa*, then the two events are **independent** of each other.
- If two events X and Y are independent then:

$$P(X|Y) = P(X)$$

$$P(X, Y) = P(X) \times P(Y)$$

- Recall, that when two events are dependent these rules are:

$$P(X|Y) = \frac{P(X, Y)}{P(Y)}$$

$$P(X, Y) = P(X|Y) \times P(Y) = P(Y|X) \times P(X)$$

Conditional Independence and Factorization

- Full independence between events is quite rare.
- A more common phenomenon is that two, or more, events may be independent if we know that a third event has happened.
- This is known as **conditional independence**.

Conditional Independence and Factorization

- For two events, X and Y , that are conditionally independent given knowledge of a third event, here Z , the definition of the probability of a joint event and conditional probability are:

$$P(X|Y, Z) = P(X|Z)$$

$$P(X, Y|Z) = P(X|Z) \times P(Y|Z)$$

$$P(X|Y) = \frac{P(X, Y)}{P(Y)}$$

$$P(X|Y) = P(X)$$

$$\begin{aligned} P(X, Y) &= P(X|Y) \times P(Y) \\ &= P(Y|X) \times P(X) \end{aligned}$$

$$P(X, Y) = P(X) \times P(Y)$$

X and Y are **independent**

X and Y are **dependent**



Conditional Independence and Factorization

- If the event $t = l$ causes the events $\mathbf{q}[1], \dots, \mathbf{q}[m]$ to happen then the events $\mathbf{q}[1], \dots, \mathbf{q}[m]$ are conditionally independent of each other given knowledge of $t = l$ and the chain rule definition can be simplified as follows:

$$\begin{aligned} P(\mathbf{q}[1], \dots, \mathbf{q}[m] \mid t = l) &= P(\mathbf{q}[1] \mid t = l) \times P(\mathbf{q}[2] \mid t = l) \times \cdots \times P(\mathbf{q}[m] \mid t = l) \\ &= \prod_{i=1}^m P(\mathbf{q}[i] \mid t = l) \end{aligned}$$

- Using this we can simplify the calculations in Bayes' Theorem, under the assumption of conditional independence between the descriptive features given the level t of the target feature:

$$P(t = I \mid \mathbf{q}[1], \dots, \mathbf{q}[m]) = \frac{\left(\prod_{i=1}^m P(\mathbf{q}[i] \mid t = I) \right) \times P(t = I)}{P(\mathbf{q}[1], \dots, \mathbf{q}[m])}$$



Conditional Independence and Factorization

Without conditional independence

$$P(X, Y, Z|W) = P(X|W) \times P(Y|X, W) \times P(Z|Y, X, W) \times P(W)$$

With conditional independence

$$P(X, Y, Z|W) = \underbrace{P(X|W)}_{\text{Factor1}} \times \underbrace{P(Y|W)}_{\text{Factor2}} \times \underbrace{P(Z|W)}_{\text{Factor3}} \times \underbrace{P(W)}_{\text{Factor4}}$$

- The joint probability distribution for the meningitis dataset.

$$\mathbf{P}(H, F, V, M) = \begin{bmatrix} P(h, f, v, m), & P(\neg h, f, v, m) \\ P(h, f, v, \neg m), & P(\neg h, f, v, \neg m) \\ P(h, f, \neg v, m), & P(\neg h, f, \neg v, m) \\ P(h, f, \neg v, \neg m), & P(\neg h, f, \neg v, \neg m) \\ P(h, \neg f, v, m), & P(\neg h, \neg f, v, m) \\ P(h, \neg f, v, \neg m), & P(\neg h, \neg f, v, \neg m) \\ P(h, \neg f, \neg v, m), & P(\neg h, \neg f, \neg v, m) \\ P(h, \neg f, \neg v, \neg m), & P(\neg h, \neg f, \neg v, \neg m) \end{bmatrix}$$

- Assuming the descriptive features are conditionally independent of each other given MENINGITIS we only need to store four factors:

Factor₁ : < $P(M)$ >

Factor₂ : < $P(h|m)$, $P(h|\neg m)$ >

Factor₃ : < $P(f|m)$, $P(f|\neg m)$ >

Factor₄ : < $P(v|m)$, $P(v|\neg m)$ >

$$P(H, F, V, M) = P(M) \times P(H|M) \times P(F|M) \times P(V|M)$$

ID	HEADACHE	FEVER	VOMITING	MENINGITIS
1	true	true	false	false
2	false	true	false	false
3	true	false	true	false
4	true	false	true	false
5	false	true	false	true
6	true	false	true	false
7	true	false	true	false
8	true	false	true	true
9	false	true	false	false
10	true	false	true	true

- Calculate the factors from the data.

$Factor_1 : < P(M) >$

$Factor_2 : < P(h|m), P(h|\neg m) >$

$Factor_3 : < P(f|m), P(f|\neg m) >$

$Factor_4 : < P(v|m), P(v|\neg m) >$



Conditional Independence and Factorization

Factor₁ : $\langle P(m) = 0.3 \rangle$

Factor₂ : $\langle P(h|m) = 0.6666, P(h|\neg m) = 0.7413 \rangle$

Factor₃ : $\langle P(f|m) = 0.3333, P(f|\neg m) = 0.4286 \rangle$

Factor₄ : $\langle P(v|m) = 0.6666, P(v|\neg m) = 0.5714 \rangle$



Conditional Independence and Factorization

Factor₁ : $\langle P(m) = 0.3 \rangle$

Factor₂ : $\langle P(h|m) = 0.6666, P(h|\neg m) = 0.7413 \rangle$

Factor₃ : $\langle P(f|m) = 0.3333, P(f|\neg m) = 0.4286 \rangle$

Factor₄ : $\langle P(v|m) = 0.6666, P(v|\neg m) = 0.5714 \rangle$

- Using the factors above calculate the probability of MENINGITIS= 'true' for the following query.

HEADACHE	FEVER	VOMITING	MENINGITIS
true	true	false	?



Conditional Independence and Factorization

$$P(m|h, f, \neg v) = \frac{P(h|m) \times P(f|m) \times P(\neg v|m) \times P(m)}{\sum_i P(h|M_i) \times P(f|M_i) \times P(\neg v|M_i) \times P(M_i)} = \\ \frac{0.6666 \times 0.3333 \times 0.3333 \times 0.3}{(0.6666 \times 0.3333 \times 0.3333 \times 0.3) + (0.7143 \times 0.4286 \times 0.4286 \times 0.7)} = 0.1948$$



Conditional Independence and Factorization

Factor₁ : $\langle P(m) = 0.3 \rangle$

Factor₂ : $\langle P(h|m) = 0.6666, P(h|\neg m) = 0.7413 \rangle$

Factor₃ : $\langle P(f|m) = 0.3333, P(f|\neg m) = 0.4286 \rangle$

Factor₄ : $\langle P(v|m) = 0.6666, P(v|\neg m) = 0.5714 \rangle$

- Using the factors above calculate the probability of MENINGITIS= 'false' for the same query.

HEADACHE	FEVER	VOMITING	MENINGITIS
true	true	false	?



Conditional Independence and Factorization

$$P(\neg m|h, f, \neg v) = \frac{P(h|\neg m) \times P(f|\neg m) \times P(\neg v|\neg m) \times P(\neg m)}{\sum_i P(h|M_i) \times P(f|M_i) \times P(\neg v|M_i) \times P(M_i)} = \\ \frac{0.7143 \times 0.4286 \times 0.4286 \times 0.7}{(0.6666 \times 0.3333 \times 0.3333 \times 0.3) + (0.7143 \times 0.4286 \times 0.4286 \times 0.7)} = 0.8052$$



Conditional Independence and Factorization

$$P(m|h, f, \neg v) = 0.1948$$

$$P(\neg m|h, f, \neg v) = 0.8052$$

- As before, the MAP prediction would be MENINGITIS = '*false*'
- The posterior probabilities are not as extreme!

Standard Approach: The Naive Bayes' Classifier



Naive Bayes' Classifier

$$\mathbb{M}(\mathbf{q}) = \arg \max_{l \in levels(t)} \left(\prod_{i=1}^m P(\mathbf{q}[i] \mid t = l) \right) \times P(t = l)$$



Naive Bayes' is simple to train!

- ① calculate the priors for each of the target levels
- ② calculate the conditional probabilities for each feature given each target level.

Table: A dataset from a loan application fraud detection domain.

ID	CREDIT HISTORY	GUARANTOR/CoAPPLICANT	ACCOMODATION	FRAUD
1	current	none	own	true
2	paid	none	own	false
3	paid	none	own	false
4	paid	guarantor	rent	true
5	arrears	none	own	false
6	arrears	none	own	true
7	current	none	own	false
8	arrears	none	own	false
9	current	none	rent	false
10	none	none	own	true
11	current	coapplicant	own	false
12	current	none	own	true
13	current	none	rent	true
14	paid	none	own	false
15	arrears	none	own	false
16	current	none	own	false
17	arrears	coapplicant	rent	false
18	arrears	none	free	false
19	arrears	none	own	false
20	paid	none	own	false

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = 'none' fr)$	=	0.1666	$P(CH = 'none' \neg fr)$	=	0
$P(CH = 'paid' fr)$	=	0.1666	$P(CH = 'paid' \neg fr)$	=	0.2857
$P(CH = 'current' fr)$	=	0.5	$P(CH = 'current' \neg fr)$	=	0.2857
$P(CH = 'arrears' fr)$	=	0.1666	$P(CH = 'arrears' \neg fr)$	=	0.4286
$P(GC = 'none' fr)$	=	0.8334	$P(GC = 'none' \neg fr)$	=	0.8571
$P(GC = 'guarantor' fr)$	=	0.1666	$P(GC = 'guarantor' \neg fr)$	=	0
$P(GC = 'coapplicant' fr)$	=	0	$P(GC = 'coapplicant' \neg fr)$	=	0.1429
$P(ACC = 'own' fr)$	=	0.6666	$P(ACC = 'own' \neg fr)$	=	0.7857
$P(ACC = 'rent' fr)$	=	0.3333	$P(ACC = 'rent' \neg fr)$	=	0.1429
$P(ACC = 'free' fr)$	=	0	$P(ACC = 'free' \neg fr)$	=	0.0714

Table: The probabilities needed by a Naive Bayes prediction model calculated from the dataset. Notation key: FR=FRAUDULENT, CH=CREDIT HISTORY, GC = GUARANTOR/CoAPPLICANT, ACC = ACCOMODATION, T='true', F='false'.

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = 'none' fr)$	=	0.1666	$P(CH = 'none' \neg fr)$	=	0
$P(CH = 'paid' fr)$	=	0.1666	$P(CH = 'paid' \neg fr)$	=	0.2857
$P(CH = 'current' fr)$	=	0.5	$P(CH = 'current' \neg fr)$	=	0.2857
$P(CH = 'arrears' fr)$	=	0.1666	$P(CH = 'arrears' \neg fr)$	=	0.4286
$P(GC = 'none' fr)$	=	0.8334	$P(GC = 'none' \neg fr)$	=	0.8571
$P(GC = 'guarantor' fr)$	=	0.1666	$P(GC = 'guarantor' \neg fr)$	=	0
$P(GC = 'coapplicant' fr)$	=	0	$P(GC = 'coapplicant' \neg fr)$	=	0.1429
$P(ACC = 'own' fr)$	=	0.6666	$P(ACC = 'own' \neg fr)$	=	0.7857
$P(ACC = 'rent' fr)$	=	0.3333	$P(ACC = 'rent' \neg fr)$	=	0.1429
$P(ACC = 'free' fr)$	=	0	$P(ACC = 'free' \neg fr)$	=	0.0714

CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMODATION	FRAUDULENT
paid	none	rent	?

A Worked Example

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = 'paid' fr)$	=	0.1666	$P(CH = 'paid' \neg fr)$	=	0.2857
$P(GC = 'none' fr)$	=	0.8334	$P(GC = 'none' \neg fr)$	=	0.8571
$P(ACC = 'rent' fr)$	=	0.3333	$P(ACC = 'rent' \neg fr)$	=	0.1429

$$\left(\prod_{k=1}^m P(\mathbf{q}[k] | fr) \right) \times P(fr) = 0.0139$$

$$\left(\prod_{k=1}^m P(\mathbf{q}[k] | \neg fr) \right) \times P(\neg fr) = 0.0245$$

CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMODATION	FRAUDULENT
paid	none	rent	?

A Worked Example

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = 'paid' fr)$	=	0.1666	$P(CH = 'paid' \neg fr)$	=	0.2857
$P(GC = 'none' fr)$	=	0.8334	$P(GC = 'none' \neg fr)$	=	0.8571
$P(ACC = 'rent' fr)$	=	0.3333	$P(ACC = 'rent' \neg fr)$	=	0.1429

$$\left(\prod_{k=1}^m P(\mathbf{q}[k] | fr) \right) \times P(fr) = 0.0139$$

$$\left(\prod_{k=1}^m P(\mathbf{q}[k] | \neg fr) \right) \times P(\neg fr) = 0.0245$$

CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMODATION	FRAUDULENT
paid	none	rent	'false'

The model is generalizing beyond the dataset!

ID	CREDIT HISTORY	GUARANTOR/ COAPPLICANT	ACCOMMODATION	FRAUD
1	current	none	own	true
2	paid	none	own	false
3	paid	none	own	false
4	paid	guarantor	rent	true
5	arrears	none	own	false
6	arrears	none	own	true
7	current	none	own	false
8	arrears	none	own	false
9	current	none	rent	false
10	none	none	own	true
11	current	coapplicant	own	false
12	current	none	own	true
13	current	none	rent	true
14	paid	none	own	false
15	arrears	none	own	false
16	current	none	own	false
17	arrears	coapplicant	rent	false
18	arrears	none	free	false
19	arrears	none	own	false
20	paid	none	own	false

CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMMODATION	FRAUDULENT
paid	none	rent	'false'



Summary

$$P(t|\mathbf{d}) = \frac{P(\mathbf{d}|t) \times P(t)}{P(\mathbf{d})} \quad (2)$$

- A Naive Bayes' classifier naively assumes that each of the descriptive features in a domain is conditionally independent of all of the other descriptive features, given the state of the target feature.
- This assumption, although often wrong, enables the Naive Bayes' model to maximally factorise the representation that it uses of the domain.
- Surprisingly, given the naivety and strength of the assumption it depends upon, a Naive Bayes' model often performs reasonably well.

1 Big Idea

2 Fundamentals

- Bayes' Theorem
- Bayesian Prediction
- Conditional Independence and Factorization

3 Standard Approach: The Naive Bayes' Classifier

- A Worked Example

4 Summary

Fundamentals of Machine Learning for Predictive Data Analytics

**Chapter 6: Probability-based Learning
Sections 6.4, 6.5**

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie brian.macnamee@ucd.ie aoife@theanalyticsstore.com

- 1 Smoothing
- 2 Continuous Features: Probability Density Functions
- 3 Continuous Features: Binning
- 4 Bayesian Networks
- 5 Summary

Smoothing

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = 'none' fr)$	=	0.1666	$P(CH = 'none' \neg fr)$	=	0
$P(CH = 'paid' fr)$	=	0.1666	$P(CH = 'paid' \neg fr)$	=	0.2857
$P(CH = 'current' fr)$	=	0.5	$P(CH = 'current' \neg fr)$	=	0.2857
$P(CH = 'arrears' fr)$	=	0.1666	$P(CH = 'arrears' \neg fr)$	=	0.4286
$P(GC = 'none' fr)$	=	0.8334	$P(GC = 'none' \neg fr)$	=	0.8571
$P(GC = 'guarantor' fr)$	=	0.1666	$P(GC = 'guarantor' \neg fr)$	=	0
$P(GC = 'coapplicant' fr)$	=	0	$P(GC = 'coapplicant' \neg fr)$	=	0.1429
$P(ACC = 'own' fr)$	=	0.6666	$P(ACC = 'own' \neg fr)$	=	0.7857
$P(ACC = 'rent' fr)$	=	0.3333	$P(ACC = 'rent' \neg fr)$	=	0.1429
$P(ACC = 'free' fr)$	=	0	$P(ACC = 'free' \neg fr)$	=	0.0714

CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMMODATION	FRAUDULENT
paid	guarantor	free	?

$P(fr) = 0.3$	$P(\neg fr) = 0.7$
$P(CH = paid fr) = 0.1666$	$P(CH = paid \neg fr) = 0.2857$
$P(GC = guarantor fr) = 0.1666$	$P(GC = guarantor \neg fr) = 0$
$P(ACC = free fr) = 0$	$P(ACC = free \neg fr) = 0.0714$

$$\left(\prod_{k=1}^m P(\mathbf{q}[k] | fr) \right) \times P(fr) = 0.0$$

$$\left(\prod_{k=1}^m P(\mathbf{q}[k] | \neg fr) \right) \times P(\neg fr) = 0.0$$

CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMMODATION	FRAUDULENT
paid	guarantor	free	?

- The standard way to avoid this issue is to use **smoothing**.
- Smoothing takes some of the probability from the events with lots of the probability share and gives it to the other probabilities in the set.

- There are several different ways to smooth probabilities, we will use **Laplace smoothing**.

Laplace Smoothing (conditional probabilities)

$$P(f = v|t) = \frac{\text{count}(f = v|t) + k}{\text{count}(f|t) + (k \times |\text{Domain}(f)|)}$$

Raw	$P(GC = \text{none} \neg fr)$	=	0.8571
Probabilities	$P(GC = \text{guarantor} \neg fr)$	=	0
	$P(GC = \text{coapplicant} \neg fr)$	=	0.1429
Smoothing	k	=	3
Parameters	$\text{count}(GC \neg fr)$	=	14
	$\text{count}(GC = \text{none} \neg fr)$	=	12
	$\text{count}(GC = \text{guarantor} \neg fr)$	=	0
	$\text{count}(GC = \text{coapplicant} \neg fr)$	=	2
	$ \text{Domain}(GC) $	=	3
Smoothed	$P(GC = \text{none} \neg fr) = \frac{12+3}{14+(3 \times 3)}$	=	0.6522
Probabilities	$P(GC = \text{guarantor} \neg fr) = \frac{0+3}{14+(3 \times 3)}$	=	0.1304
	$P(GC = \text{coapplicant} \neg fr) = \frac{2+3}{14+(3 \times 3)}$	=	0.2174

Table: Smoothing the posterior probabilities for the GUARANTOR/COAPPLICANT feature conditioned on FRAUDULENT being False.

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = none fr)$	=	0.2222	$P(CH = none \neg fr)$	=	0.1154
$P(CH = paid fr)$	=	0.2222	$P(CH = paid \neg fr)$	=	0.2692
$P(CH = current fr)$	=	0.3333	$P(CH = current \neg fr)$	=	0.2692
$P(CH = arrears fr)$	=	0.2222	$P(CH = arrears \neg fr)$	=	0.3462
$P(GC = none fr)$	=	0.5333	$P(GC = none \neg fr)$	=	0.6522
$P(GC = guarantor fr)$	=	0.2667	$P(GC = guarantor \neg fr)$	=	0.1304
$P(GC = coapplicant fr)$	=	0.2	$P(GC = coapplicant \neg fr)$	=	0.2174
$P(ACC = own fr)$	=	0.4667	$P(ACC = own \neg fr)$	=	0.6087
$P(ACC = rent fr)$	=	0.3333	$P(ACC = rent \neg fr)$	=	0.2174
$P(ACC = Free fr)$	=	0.2	$P(ACC = Free \neg fr)$	=	0.1739

Table: The Laplace smoothed, with $k = 3$, probabilities needed by a Naive Bayes prediction model calculated from the fraud detection dataset. Notation key: FR=FRAUDULENT, CH=CREDIT HISTORY, GC = GUARANTOR/CoAPPLICANT, ACC = ACCOMODATION, T='True', F='False'.

CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMMODATION	FRAUDULENT
paid	guarantor	free	?

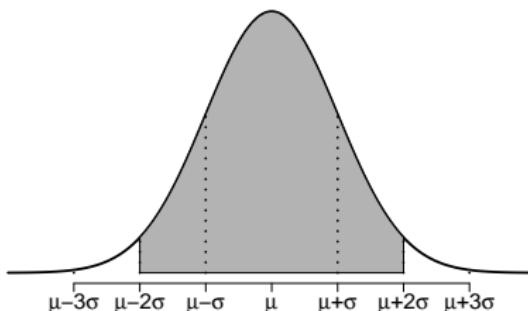
$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = paid fr)$	=	0.2222	$P(CH = paid \neg fr)$	=	0.2692
$P(GC = guarantor fr)$	=	0.2667	$P(GC = guarantor \neg fr)$	=	0.1304
$P(ACC = Free fr)$	=	0.2	$P(ACC = Free \neg fr)$	=	0.1739
$(\prod_{k=1}^m P(\mathbf{q}[m] fr)) \times P(fr) = 0.0036$					
$(\prod_{k=1}^m P(\mathbf{q}[m] \neg fr)) \times P(\neg fr) = 0.0043$					

Table: The relevant smoothed probabilities, from Table 2 [9], needed by the Naive Bayes prediction model in order to classify the query from the previous slide and the calculation of the scores for each candidate classification.

Continuous Features: Probability Density Functions

- A **probability density function** (PDF) represents the probability distribution of a continuous feature using a mathematical function, such as the normal distribution.

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



- A PDF defines a density curve and the shape of the curve is determined by:
 - the statistical distribution that is used to define the PDF
 - the values of the statistical distribution parameters

Table: Definitions of some standard probability distributions.

Normal

 $x \in \mathbb{R}$ $\mu \in \mathbb{R}$ $\sigma \in \mathbb{R}_{>0}$

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

Student- t $x \in \mathbb{R}$ $\phi \in \mathbb{R}$ $\rho \in \mathbb{R}_{>0}$ $\kappa \in \mathbb{R}_{>0}$

$$z = \frac{x - \phi}{\rho}$$

$$\tau(x, \phi, \rho, \kappa) = \frac{\Gamma(\frac{\kappa+1}{2})}{\Gamma(\frac{\kappa}{2}) \times \sqrt{\pi\kappa} \times \rho} \times \left(1 + \left(\frac{1}{\kappa} \times z^2\right)\right)^{-\frac{\kappa+1}{2}}$$

Exponential

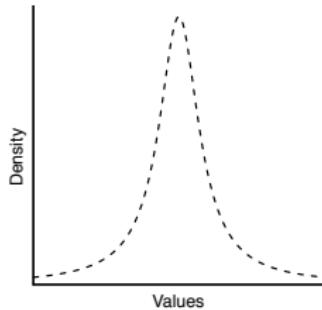
 $x \in \mathbb{R}$ $\lambda \in \mathbb{R}_{>0}$

$$E(x, \lambda) = \begin{cases} \lambda e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

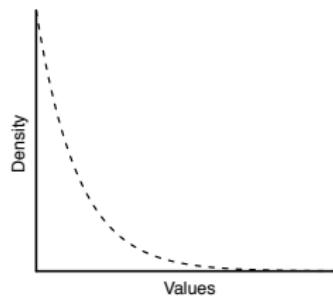
Mixture of n Gaussians $x \in \mathbb{R}$ $\{\mu_1, \dots, \mu_n | \mu_i \in \mathbb{R}\}$ $\{\sigma_1, \dots, \sigma_n | \sigma_i \in \mathbb{R}_{>0}\}$ $\{\omega_1, \dots, \omega_n | \omega_i \in \mathbb{R}_{>0}\}$

$$\sum_{i=1}^n \omega_i = 0$$

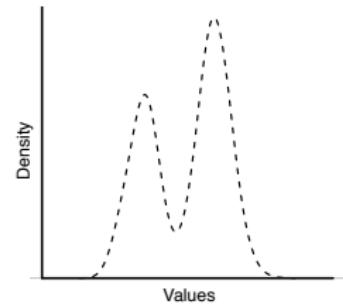
$$N(x, \mu_1, \sigma_1, \omega_1, \dots, \mu_n, \sigma_n, \omega_n) = \sum_{i=1}^n \frac{\omega_i}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$$



(a) Normal/Student-t



(b) Exponential



(c) Mixture of Gaussians

Figure: Plots of some well known probability distributions.

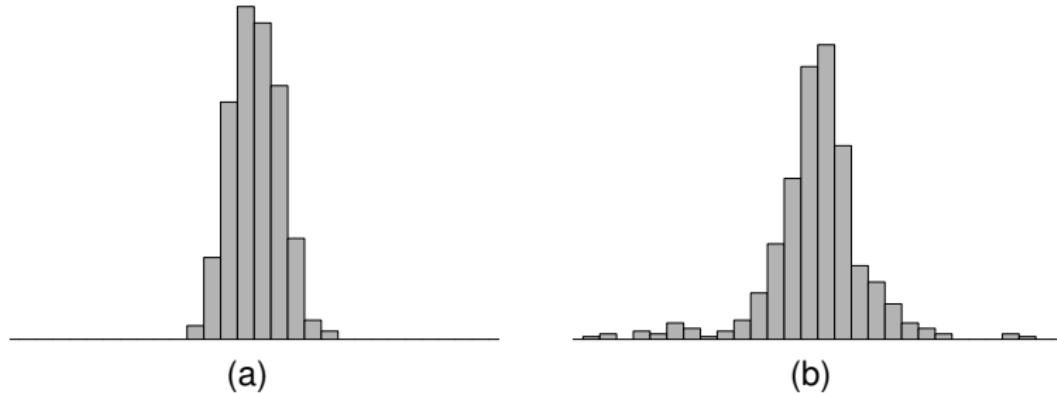


Figure: Histograms of two unimodal datasets: (a) the distribution has light tails; (b) the distribution has fat tails.

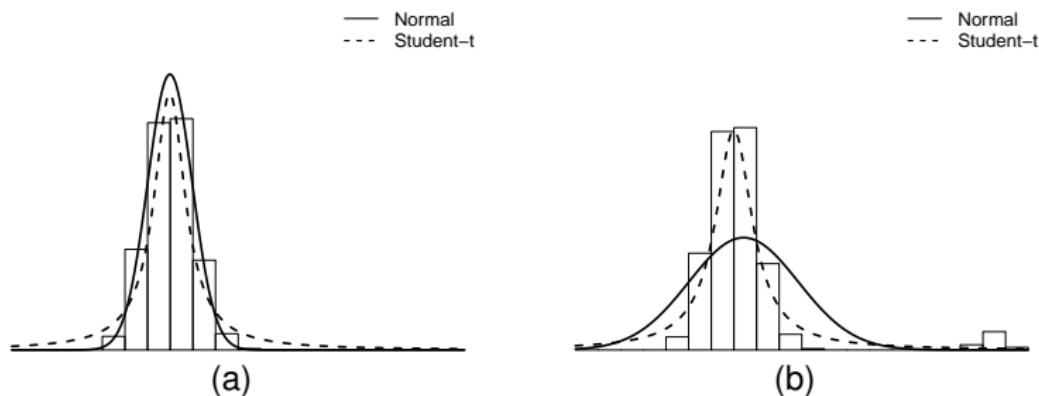


Figure: Illustration of the robustness of the student- t distribution to outliers: (a) a density histogram of a unimodal dataset overlaid with the density curves of a normal and a student- t distribution that have been fitted to the data; (b) a density histogram of the same dataset with outliers added, overlaid with the density curves of a normal and a student- t distribution that have been fitted to the data. The student- t distribution is less affected by the introduction of outliers. (This figure is inspired by Figure 2.16 in (Bishop, 2006).)

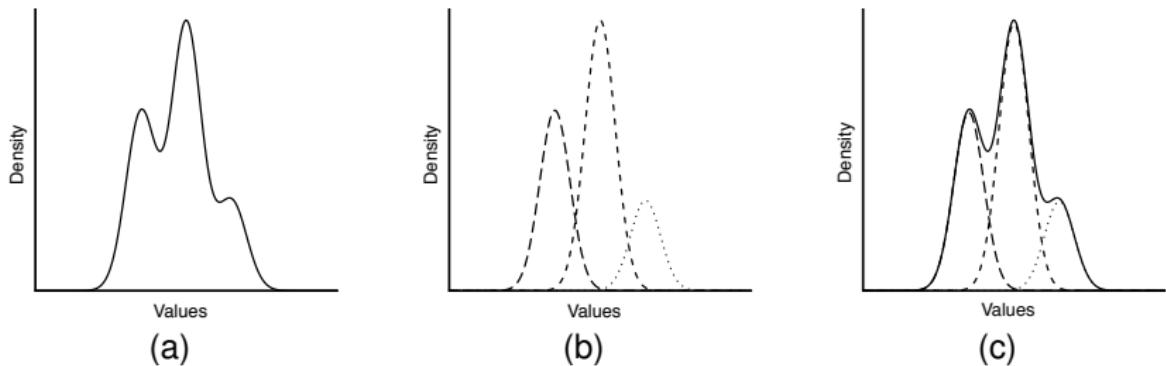
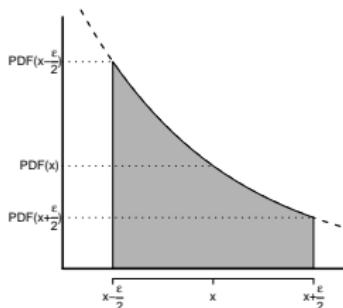
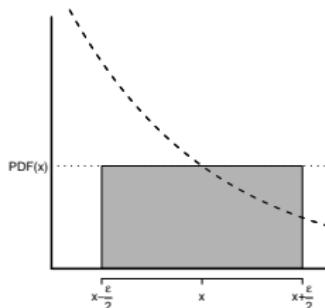


Figure: Illustration of how a mixture of Gaussians model is composed of a number of normal distributions. The curve plotted using a solid line is the mixture of Gaussians density curve, created using an appropriately weighted summation of the three normal curves, plotted using dashed and dotted lines.

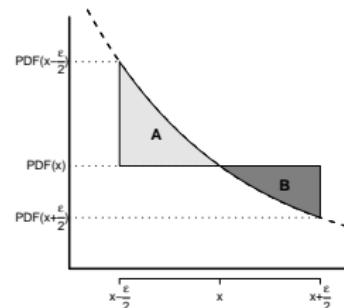
- A PDF is an abstraction over a density histogram and consequently PDF represents probabilities in terms of area under the curve.
- To use a PDF to calculate a probability we need to think in terms of the area under an interval of the PDF curve.
- We can calculate the area under a PDF by looking this up in a probability table or to use integration to calculate the area under the curve within the bounds of the interval.



(a)



(b)



(c)

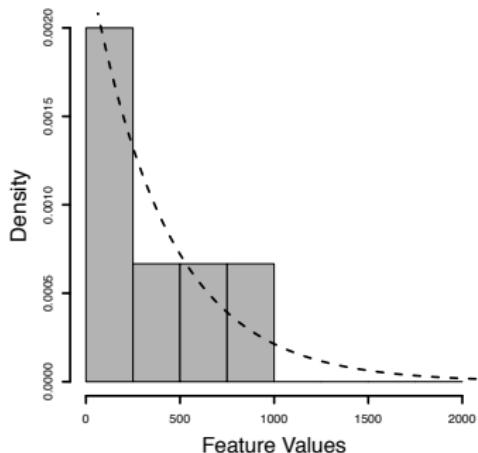
Figure: (a) The area under a density curve between the limits $x - \frac{\epsilon}{2}$ and $x + \frac{\epsilon}{2}$; (b) the approximation of this area computed by $PDF(x) \times \epsilon$; and (c) the error in the approximation is equal to the difference between area A, the area under the curve omitted from the approximation, and area B, the area above the curve erroneously included in the approximation. Both of these areas will get smaller as the width of the interval gets smaller, resulting in a smaller error in the approximation.

- There is no hard and fast rule for deciding on **interval size**
- instead, this decision is done on a case by case basis and is dependent on the precision required in answering a question.
- To illustrate how PDFs can be used in Naive Bayes models we will extend our loan application fraud detection query to have an ACCOUNT BALANCE feature

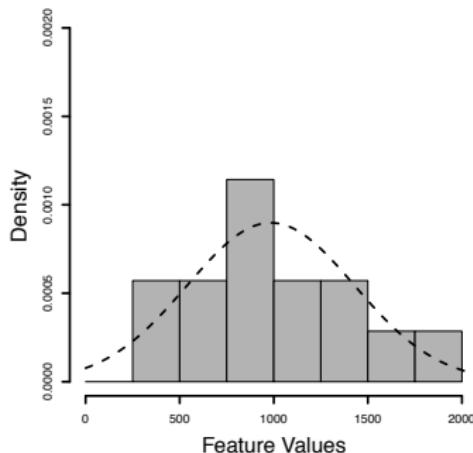
Table: The dataset from the loan application fraud detection domain with a new continuous descriptive features added: ACCOUNT BALANCE

ID	CREDIT HISTORY	GUARANTOR/CoAPPLICANT	ACCOMMODATION	ACCOUNT BALANCE	FRAUD
1	current	none	own	56.75	true
2	current	none	own	1,800.11	false
3	current	none	own	1,341.03	false
4	paid	guarantor	rent	749.50	true
5	arrears	none	own	1,150.00	false
6	arrears	none	own	928.30	true
7	current	none	own	250.90	false
8	arrears	none	own	806.15	false
9	current	none	rent	1,209.02	false
10	none	none	own	405.72	true
11	current	coapplicant	own	550.00	false
12	current	none	free	223.89	true
13	current	none	rent	103.23	true
14	paid	none	own	758.22	false
15	arrears	none	own	430.79	false
16	current	none	own	675.11	false
17	arrears	coapplicant	rent	1,657.20	false
18	arrears	none	free	1,405.18	false
19	arrears	none	own	760.51	false
20	current	none	own	985.41	false

- We need to define two PDFs for the new ACCOUNT BALANCE (AB) feature with each PDF conditioned on a different value in the domain or the target:
 - $P(AB = X|fr) = PDF_1(AB = X|fr)$
 - $P(AB = X|\neg fr) = PDF_2(AB = X|\neg fr)$
- Note that these two PDFs do not have to be defined using the same statistical distribution.



(a)



(b)

Figure: Histograms, using a bin size of 250 units, and density curves for the ACCOUNT BALANCE feature: (a) the fraudulent instances overlaid with a fitted exponential distribution; (b) the non-fraudulent instances overlaid with a fitted normal distribution.

- From the shape of these histograms it appears that
 - the distribution of values taken by the ACCOUNT BALANCE feature in the set of instances where the target feature FRAUDULENT= '*True*' follows an exponential distribution
 - the distributions of values taken by the ACCOUNT BALANCE feature in the set of instances where the target feature FRAUDULENT= '*False*' is similar to a normal distribution.
- Once we have selected the distributions the next step is to fit the distributions to the data.

- To fit the exponential distribution we simply compute the sample mean, \bar{x} , of the ACCOUNT BALANCE feature in the set of instances where FRAUDULENT= 'True' and set the λ parameter equal to one divided by \bar{x} .
- To fit the normal distribution to the set of instances where FRAUDULENT= 'False' we simply compute the sample mean and sample standard deviation, s , for the ACCOUNT BALANCE feature for this set of instances and set the parameters of the normal distribution to these values.

Table: Partitioning the dataset based on the value of the target feature and fitting the parameters of a statistical distribution to model the ACCOUNT BALANCE feature in each partition.

ACCOUNT			
ID	...	BALANCE	FRAUD
1		56.75	true
4		749.50	true
6		928.30	true
10	...	405.72	true
12		223.89	true
13		103.23	true
AB		411.22	
$\lambda =^1 / \overline{AB}$		0.0024	

ACCOUNT			
ID	...	BALANCE	FRAUD
2		1800.11	false
3		1341.03	false
5		1150.00	false
7		250.90	false
8		806.15	false
9		1209.02	false
11		550.00	false
14		758.22	false
15		430.79	false
16		675.11	false
17		1657.20	false
18		1405.18	false
19		760.51	false
20		985.41	false
AB		984.26	
$sd(AB)$		460.94	

Table: The Laplace smoothed (with $k = 3$) probabilities needed by a naive Bayes prediction model calculated from the dataset in Table 5 [23], extended to include the conditional probabilities for the new ACCOUNT BALANCE feature, which are defined in terms of PDFs.

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = none fr)$	=	0.2222	$P(CH = none \neg fr)$	=	0.1154
$P(CH = paid fr)$	=	0.2222	$P(CH = paid \neg fr)$	=	0.2692
$P(CH = current fr)$	=	0.3333	$P(CH = current \neg fr)$	=	0.2692
$P(CH = arrears fr)$	=	0.2222	$P(CH = arrears \neg fr)$	=	0.3462
$P(GC = none fr)$	=	0.5333	$P(GC = none \neg fr)$	=	0.6522
$P(GC = guarantor fr)$	=	0.2667	$P(GC = guarantor \neg fr)$	=	0.1304
$P(GC = coapplicant fr)$	=	0.2	$P(GC = coapplicant \neg fr)$	=	0.2174
$P(ACC = own fr)$	=	0.4667	$P(ACC = own \neg fr)$	=	0.6087
$P(ACC = rent fr)$	=	0.3333	$P(ACC = rent \neg fr)$	=	0.2174
$P(ACC = free fr)$	=	0.2	$P(ACC = free \neg fr)$	=	0.1739
$P(AB = x fr)$			$P(AB = x \neg fr)$		
≈		$E \left(\begin{array}{c} x, \\ \lambda = 0.0024 \end{array} \right)$	≈		$N \left(\begin{array}{c} x, \\ \mu = 984.26, \\ \sigma = 460.94 \end{array} \right)$

Table: A query loan application from the fraud detection domain.

Credit History	Guarantor/CoApplicant	Accommodation	Account Balance	Fraudulent
paid	guarantor	free	759.07	?

Table: The probabilities, from Table 7 [29], needed by the naive Bayes prediction model to make a prediction for the query $\langle CH = 'paid', GC = 'guarantor', ACC = 'free', AB = 759.07 \rangle$ and the calculation of the scores for each candidate prediction.

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = paid fr)$	=	0.2222	$P(CH = paid \neg fr)$	=	0.2692
$P(GC = guarantor fr)$	=	0.2667	$P(GC = guarantor \neg fr)$	=	0.1304
$P(ACC = free fr)$	=	0.2	$P(ACC = free \neg fr)$	=	0.1739
$P(AB = 759.07 fr)$			$P(AB = 759.07 \neg fr)$		
$\approx E \begin{pmatrix} 759.07, \\ \lambda = 0.0024 \end{pmatrix}$	=	0.00039	$\approx N \begin{pmatrix} 759.07, \\ \mu = 984.26, \\ \sigma = 460.94 \end{pmatrix}$	=	0.00077

$$\left(\prod_{k=1}^m P(\mathbf{q}[k]|fr) \right) \times P(fr) = 0.0000014$$

$$\left(\prod_{k=1}^m P(\mathbf{q}[k]|\neg fr) \right) \times P(\neg fr) = 0.0000033$$

Continuous Features: Binning

- In Section 3.6.2 we explained two of the best known binning techniques **equal-width** and **equal-frequency**.
- We can use these techniques to *bin* continuous features into categorical features
- In general we recommend **equal-frequency binning**.

Table: The dataset from a loan application fraud detection domain with a second continuous descriptive feature added: LOAN AMOUNT

ID	CREDIT HISTORY	GUARANTOR/COAPPLICANT	ACCOMMODATION	ACCOUNT BALANCE	LOAN AMOUNT	FRAUD
1	current	none	own	56.75	900	true
2	current	none	own	1 800.11	150 000	false
3	current	none	own	1 341.03	48 000	false
4	paid	guarantor	rent	749.50	10 000	true
5	arrears	none	own	1 150.00	32 000	false
6	arrears	none	own	928.30	250 000	true
7	current	none	own	250.90	25 000	false
8	arrears	none	own	806.15	18 500	false
9	current	none	rent	1 209.02	20 000	false
10	none	none	own	405.72	9 500	true
11	current	coapplicant	own	550.00	16 750	false
12	current	none	free	223.89	9 850	true
13	current	none	rent	103.23	95 500	true
14	paid	none	own	758.22	65 000	false
15	arrears	none	own	430.79	500	false
16	current	none	own	675.11	16 000	false
17	arrears	coapplicant	rent	1 657.20	15 450	false
18	arrears	none	free	1 405.18	50 000	false
19	arrears	none	own	760.51	500	false
20	current	none	own	985.41	35 000	false

Table: The LOAN AMOUNT continuous feature discretized into 4 equal-frequency bins.

BINNED LOAN AMOUNT				BINNED LOAN AMOUNT			
ID	LOAN AMOUNT	BINNED LOAN AMOUNT	FRAUD	ID	LOAN AMOUNT	BINNED LOAN AMOUNT	FRAUD
15	500	bin1	false	9	20,000	bin3	false
19	500	bin1	false	7	25,000	bin3	false
1	900	bin1	true	5	32,000	bin3	false
10	9,500	bin1	true	20	35,000	bin3	false
12	9,850	bin1	true	3	48,000	bin3	false
4	10,000	bin2	true	18	50,000	bin4	false
17	15,450	bin2	false	14	65,000	bin4	false
16	16,000	bin2	false	13	95,500	bin4	true
11	16,750	bin2	false	2	150,000	bin4	false
8	18,500	bin2	false	6	250,000	bin4	true

- Once we have discretized the data we need to record the raw continuous feature threshold between the bins so that we can use these for query feature values.

Table: The thresholds used to discretize the LOAN AMOUNT feature in queries.

Bin Thresholds		
	Bin1	$\leq 9,925$
$9,925 <$	Bin2	$\leq 19,250$
$19,225 <$	Bin3	$\leq 49,000$
$49,000 <$	Bin4	

Table: The Laplace smoothed (with $k = 3$) probabilities needed by a naive Bayes prediction model calculated from the fraud detection dataset. Notation key: FR = FRAUD, CH = CREDIT HISTORY, AB = ACCOUNT BALANCE, GC = GUARANTOR/CoAPPLICANT, ACC = ACCOMMODATION, BLA = BINNED LOAN AMOUNT.

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = \text{none} fr)$	=	0.2222	$P(CH = \text{none} \neg fr)$	=	0.1154
$P(CH = \text{paid} fr)$	=	0.2222	$P(CH = \text{paid} \neg fr)$	=	0.2692
$P(CH = \text{current} fr)$	=	0.3333	$P(CH = \text{current} \neg fr)$	=	0.2692
$P(CH = \text{arrears} fr)$	=	0.2222	$P(CH = \text{arrears} \neg fr)$	=	0.3462
$P(GC = \text{none} fr)$	=	0.5333	$P(GC = \text{none} \neg fr)$	=	0.6522
$P(GC = \text{guarantor} fr)$	=	0.2667	$P(GC = \text{guarantor} \neg fr)$	=	0.1304
$P(GC = \text{coapplicant} fr)$	=	0.2	$P(GC = \text{coapplicant} \neg fr)$	=	0.2174
$P(ACC = \text{own} fr)$	=	0.4667	$P(ACC = \text{own} \neg fr)$	=	0.6087
$P(ACC = \text{rent} fr)$	=	0.3333	$P(ACC = \text{rent} \neg fr)$	=	0.2174
$P(ACC = \text{free} fr)$	=	0.2	$P(ACC = \text{free} \neg fr)$	=	0.1739
$P(AB = x fr)$			$P(AB = x \neg fr)$		
$\approx E\left(\frac{x}{\lambda}, \lambda = 0.0024\right)$			$\approx N\left(\frac{x}{\mu}, \mu = 984.26, \sigma = 460.94\right)$		
$P(BLA = \text{bin1} fr)$	=	0.3333	$P(BLA = \text{bin1} \neg fr)$	=	0.1923
$P(BLA = \text{bin2} fr)$	=	0.2222	$P(BLA = \text{bin2} \neg fr)$	=	0.2692
$P(BLA = \text{bin3} fr)$	=	0.1667	$P(BLA = \text{bin3} \neg fr)$	=	0.3077
$P(BLA = \text{bin4} fr)$	=	0.2778	$P(BLA = \text{bin4} \neg fr)$	=	0.2308

Table: A query loan application from the fraud detection domain.

Credit History	Guarantor/ CoApplicant	Accomodation	Account Balance	Loan Amount	Fraudulent
paid	guarantor	free	759.07	8,000	?

Table: The relevant smoothed probabilities, from Table 13 [37], needed by the naive Bayes model to make a prediction for the query $\langle CH = 'paid', GC = 'guarantor', ACC = 'free', AB = 759.07, LA = 8\,000 \rangle$ and the calculation of the scores for each candidate prediction.

$P(fr)$	=	0.3	$P(\neg fr)$	=	0.7
$P(CH = paid fr)$	=	0.2222	$P(CH = paid \neg fr)$	=	0.2692
$P(GC = guarantor fr)$	=	0.2667	$P(GC = guarantor \neg fr)$	=	0.1304
$P(ACC = free fr)$	=	0.2	$P(ACC = free \neg fr)$	=	0.1739
$P(AB = 759.07 fr)$			$P(AB = 759.07 \neg fr)$		
$\approx E \begin{pmatrix} 759.07, \\ \lambda = 0.0024 \end{pmatrix}$	=	0.00039	$\approx N \begin{pmatrix} 759.07, \\ \mu = 984.26, \\ \sigma = 460.94 \end{pmatrix}$	=	0.00077
$P(BLA = bin1 fr)$	=	0.3333	$P(BLA = bin1 \neg fr)$	=	0.1923
$(\prod_{k=1}^m P(\mathbf{q}[k] fr)) \times P(fr) = 0.000000462$					
$(\prod_{k=1}^n P(\mathbf{q}[k] \neg fr)) \times P(\neg fr) = 0.000000633$					

Bayesian Networks

- **Bayesian networks** use a graph-based representation to encode the structural relationships—such as direct influence and conditional independence—between subsets of features in a domain.
- Consequently, a Bayesian network representation is generally more compact than a full joint distribution, yet is not forced to assert global conditional independence between all descriptive features.

A Bayesian Network is a directed acyclical graph that is composed of three basic elements:

- nodes
- edges
- conditional probability tables (CPT)

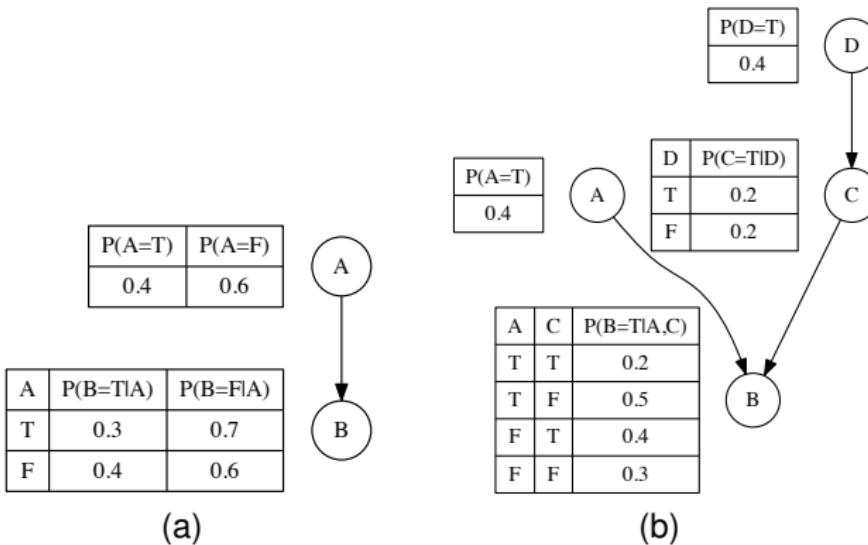


Figure: (a) A Bayesian network for a domain consisting of two binary features. The structure of the network states that the value of feature A directly influences the value of feature B. (b) A Bayesian network consisting of 4 binary features with a path containing 3 generations of nodes: D, C, and B.

- In probability terms the directed edge from A to B in Figure (a) on the previous slide states that:

$$P(A, B) = P(B|A) \times P(A) \quad (1)$$

- For example, the probability of the event a and $\neg b$ is

$$P(a, \neg b) = P(\neg b|a) \times P(a) = 0.7 \times 0.4 = 0.28$$

- Equation (1)^[44] can be generalized to the statement that for any network with N nodes, the probability of an event x_1, \dots, x_n , can be computed using the following formula:

$$P(x_1, \dots, x_n) = \prod_{i=1}^n P(x_i | Parents(x_i)) \quad (2)$$

- For example, using the more complex Bayesian network in figure (b) above, we can calculate the probability of the joint event $P(a, \neg b, \neg c, d)$ as follows:

$$\begin{aligned}P(a, \neg b, \neg c, d) &= P(\neg b|a, \neg c) \times P(\neg c|d) \times P(a) \times P(d) \\&= 0.5 \times 0.8 \times 0.4 \times 0.4 = 0.064\end{aligned}$$

- We can use Bayes' Theorem to invert the dependencies between nodes in a network.
- Returning to the simpler network in figure (a) above we can calculate $P(a|\neg b)$ as follows:

$$\begin{aligned} P(a|\neg b) &= \frac{P(\neg b|a) \times P(a)}{P(\neg b)} = \frac{P(\neg b|a) \times P(a)}{\sum_i P(\neg b|A_i)} \\ &= \frac{P(\neg b|a) \times P(a)}{(P(\neg b|a) \times P(a)) + (P(\neg b|\neg a) \times P(\neg a))} \\ &= \frac{0.7 \times 0.4}{(0.7 \times 0.4) + (0.6 \times 0.6)} = 0.4375 \end{aligned}$$

- For conditional independence we need to take into account not only the parents of a node by also the state of its children and their parents.
- The set of nodes in a graph that make a node independent of the rest of the graph are known as the **Markov blanket** of a node.

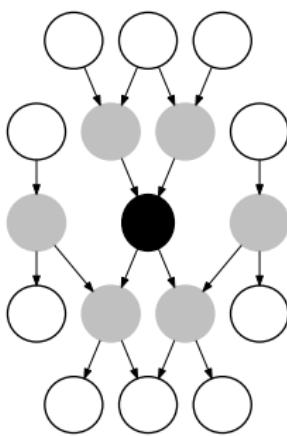


Figure: A depiction of the Markov blanket of a node. The gray nodes define the Markov blanket of the black node. The black node is conditionally independent of the white nodes given the state of the gray nodes.

- The conditional independence of a node x_i in a graph with n nodes is defined as:

$$P(x_i | x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = \\ P(x_i | \text{Parents}(x_i)) \prod_{j \in \text{Children}(x_i)} P(x_j | \text{Parents}(x_j)) \quad (3)$$

- Applying the equation of the preceding slide to the network in figure (b) above we can calculate the probability of $P(c|\neg a, b, d)$ as

$$\begin{aligned}P(c|\neg a, b, d) &= P(c|d) \times P(b|c, \neg a) \\&= 0.2 \times 0.4 = 0.08\end{aligned}$$

- A naive Bayes classifier is a Bayesian network with a specific topological structure.

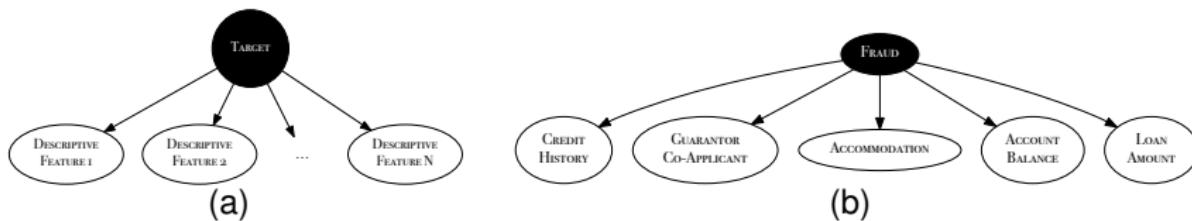


Figure: (a) A Bayesian network representation of the conditional independence asserted by a naive Bayes model between the descriptive features given knowledge of the target feature; (b) a Bayesian network representation of the conditional independence assumption for the naive Bayes model in the fraud example.

- When we computed a conditional probability for a target feature using a naive Bayes model, we used the following calculation

$$P(t|\mathbf{d}[1], \dots, \mathbf{d}[n]) = P(t) \prod_{j \in \text{Children}(t)} P(\mathbf{d}[j]|t)$$

- This equation is equivalent to Equation (3)^[50] from earlier.

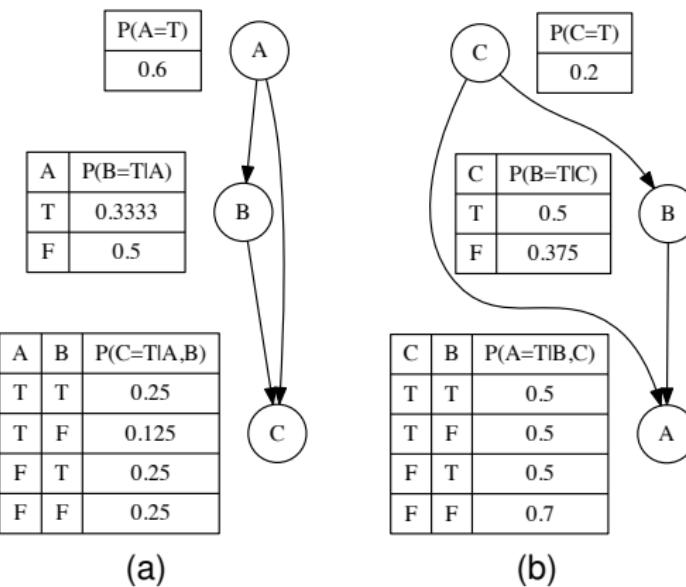
- Computing a conditional probability for a node becomes more complex if the value of one or more of the parent nodes is unknown.

- For example, in the context of the network in figure (b) above, to compute $P(b|a, d)$ where the status of node C is unknown we would do the following calculations:
 - Compute the distribution for C given D : $P(c | d) = 0.2$, $P(\neg c | d) = 0.8$
 - Compute $P(b \perp a, C)$ by summing out C :

$$P(b | a, C) = \sum_i P(b | a, C_i)$$

$$\begin{aligned}
 P(b | a, C) &= \sum_i P(b | a, C_i) = \sum_i \frac{P(b, a, C_i)}{P(a, C_i)} \\
 &= \frac{(P(b | a, c) \times P(a) \times P(c)) + (P(b | a, \neg c) \times P(a) \times P(\neg c))}{(P(a) \times P(c)) + (P(a) \times P(\neg c))} \\
 &= \frac{(0.2 \times 0.4 \times 0.2) + (0.5 \times 0.4 \times 0.8)}{(0.4 \times 0.2) + (0.4 \times 0.8)} = 0.44
 \end{aligned}$$

- This example illustrates the power of Bayesian networks.
 - When complete knowledge of the state of all the nodes in the network is not available, we clamp the values of nodes that we do have knowledge of and sum out the unknown nodes.



(a)

Figure: Two different Bayesian networks, each defining the same full joint probability distribution.

- We can illustrate that these two networks encode the same joint probability distribution by using each network to compute $P(\neg a, b, c)$
- Using network (a) we get:

$$\begin{aligned}P(\neg a, b, c) &= P(c|\neg a, b) \times P(b|\neg a) \times P(\neg a) \\&= 0.25 \times 0.5 \times 0.4 = 0.05\end{aligned}$$

- Using network (b) we get:

$$\begin{aligned}P(\neg a, b, c) &= P(\neg a|c, b) \times P(b|c) \times P(c) \\&= 0.5 \times 0.5 \times 0.2 = 0.05\end{aligned}$$

- The simplest was to construct a Bayesian network is to use a hybrid approach where:
 - ➊ the topology of the network is given to the learning algorithm,
 - ➋ and the learning task involves inducing the CPT from the data.

Table: (a) Some socio-economic data for a set of countries; (b) a binned version of the data listed in (a).

COUNTRY ID	GINI COEF	SCHOOL YEARS	LIFE EXP	CPI	GINI COEF	SCHOOL YEARS	LIFE EXP	CPI
Afghanistan	27.82	0.40	59.61	1.52	low	low	low	low
Argentina	44.49	10.10	75.77	3.00	high	low	low	low
Australia	35.19	11.50	82.09	8.84	low	high	high	high
Brazil	54.69	7.20	73.12	3.77	high	low	low	low
Canada	32.56	14.20	80.99	8.67	low	high	high	high
China	42.06	6.40	74.87	3.64	high	low	low	low
Egypt	30.77	5.30	70.48	2.86	low	low	low	low
Germany	28.31	12.00	80.24	8.05	low	high	high	high
Haiti	59.21	3.40	45.00	1.80	high	low	low	low
Ireland	34.28	11.50	80.15	7.54	low	high	high	high
Israel	39.2	12.50	81.30	5.81	low	high	high	high
New Zealand	36.17	12.30	80.67	9.46	low	high	high	high
Nigeria	48.83	4.10	51.30	2.45	high	low	low	low
Russia	40.11	12.90	67.62	2.45	high	high	low	low
Singapore	42.48	6.10	81.788	9.17	high	low	high	high
South Africa	63.14	8.50	54.547	4.08	high	low	low	low
Sweden	25.00	12.80	81.43	9.30	low	high	high	high
U.K.	35.97	13.00	80.09	7.78	low	high	high	high
U.S.A	40.81	13.70	78.51	7.14	high	high	high	high
Zimbabwe	50.10	6.7	53.684	2.23	high	low	low	low

(a)

(b)

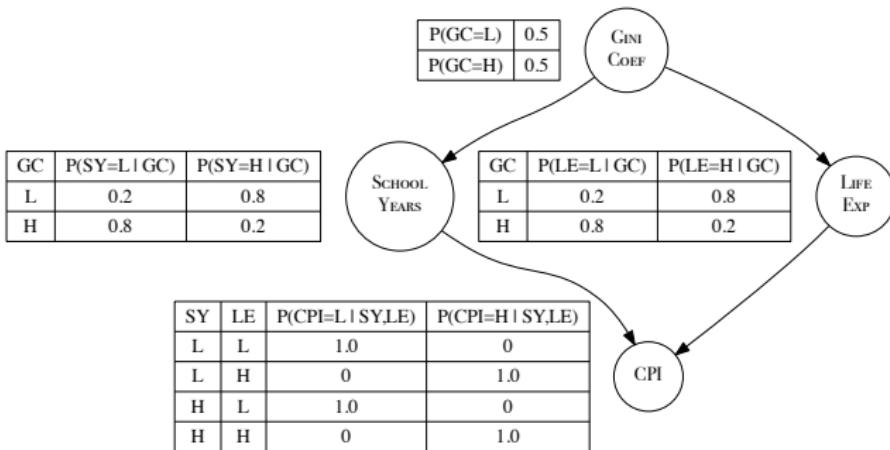


Figure: A Bayesian network that encodes the causal relationships between the features in the corruption domain. The CPT entries have been calculated using the data from Table 16 [61](b).

$$\mathbb{M}(\mathbf{q}) = \arg \max_{l \in levels(t)} BayesianNetwork(t = l, \mathbf{q}) \quad (4)$$

Example

- We wish to predict the CPI for a country with the follow profile:

GINI COEF = '*high*', SCHOOL YEARS = '*high*'

$$\begin{aligned} P(CPI = H | SY = H, GC = H) &= \frac{P(CPI = H, SY = H, GC = H)}{P(SY = H, GC = H)} \\ &= \frac{\sum_{i \in H, L} P(CPI = H, SY = H, GC = H, LE = i)}{P(SY = H, GC = H)} \end{aligned}$$

$$\begin{aligned} & \sum_{i \in \{H,L\}} P(CPI = H, SY = H, GC = H, LE = i) \\ &= \sum_{i \in \{H,L\}} P(CPI = H | SY = H, LE = i) \times P(SY = H | GC = H) \\ & \quad \times P(LE = i | GC = H) \times P(GC = H) \\ &= (P(CPI = H | SY = H, LE = H) \times P(SY = H | GC = H) \\ & \quad \times P(LE = H | GC = H) \times P(GC = H)) \\ & \quad + (P(CPI = H | SY = H, LE = L) \times P(SY = H | GC = H) \\ & \quad \times P(LE = L | GC = H) \times P(GC = H)) \\ &= (1.0 \times 0.2 \times 0.2 \times 0.5) + (0 \times 0.2 \times 0.8 \times 0.5) = 0.02 \end{aligned}$$

$$\begin{aligned} P(SY = H, GC = H) &= P(SY = H|GC = H) \times P(GC = H) \\ &= 0.2 \times 0.5 = 0.1 \end{aligned}$$

$$P(CPI = H | SY = H, GC = H) = \frac{0.02}{0.1} = 0.2$$

- Because of the calculation complexity that can arise when using Bayesian networks to do exact inference a popular approach is to approximate the required probability distribution using **Markov Chain Monte Carlo** algorithms.
- Gibbs sampling** is one of the best known MCMC algorithms.
 - 1 Clamp the values of the evidence variables and randomly assign the values of the non-evidence variables.
 - 2 Generate samples by changing the value of one of the non-evidence variables using the distribution for the node conditioned on the state of the rest of the network.

Table: Examples of the samples generated using Gibbs sampling.

Sample Number	Gibbs Iteration	Feature Updated	GINI COEF	SCHOOL YEARS	LIFE EXP	CPI
1	37	CPI	high	high	high	low
2	44	LIFE EXP	high	high	high	low
3	51	CPI	high	high	high	low
4	58	LIFE EXP	high	high	low	high
5	65	CPI	high	high	high	low
6	72	LIFE EXP	high	high	high	low
7	79	CPI	high	high	low	high
8	86	LIFE EXP	high	high	low	low
9	93	CPI	high	high	high	low
10	100	LIFE EXP	high	high	high	low
11	107	CPI	high	high	low	high
12	114	LIFE EXP	high	high	high	low
13	121	CPI	high	high	high	low
14	128	LIFE EXP	high	high	high	low
15	135	CPI	high	high	high	low
16	142	LIFE EXP	high	high	low	low

$$\mathbb{M}(\mathbf{q}) = \arg \max_{l \in levels(t)} Gibbs(t = l, \mathbf{q}) \quad (5)$$

Summary

- Naive Bayes models can suffer from zero probabilities of relatively rare events. **Smoothing** is an easy way to combat this.
- Two ways to handle continuous features in probability-based models are: **Probability density functions** and **Binning**
- Using probability density functions requires that we match the observed data to an existing distribution.
- Although binning results in information loss it is a simple and effective way to handle continuous features in probability-based models.
- Bayesian network representation is generally more compact than a full joint distribution, yet is not forced to assert global conditional independence between all descriptive features.

- 1 Smoothing**
- 2 Continuous Features: Probability Density Functions**
- 3 Continuous Features: Binning**
- 4 Bayesian Networks**
- 5 Summary**



Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 7: Error-based Learning

Sections 7.1, 7.2, 7.3

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie

brian.macnamee@ucd.ie

aoife@theanalyticsstore.com

1 Big Idea

2 Fundamentals

- Simple Linear Regression
- Measuring Error
- Error Surfaces

3 Standard Approach: Multivariate Linear Regression with Gradient Descent

- Multivariate Linear Regression
- Gradient Descent
- Choosing Learning Rates & Initial Weights
- A Worked Example

4 Summary

Big Idea



- A **paramaterised** prediction model is initialised with a set of random parameters and an error function is used to judge how well this initial model performs when making predictions for instances in a training dataset.
- Based on the value of the error function the parameters are iteratively adjusted to create a more and more accurate model.

Fundamentals

Table: The **office rentals dataset**: a dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING	RENTAL PRICE
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

Table: The **office rentals dataset**: a dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.

RENTAL		
ID	SIZE	PRICE
1	500	320
2	550	380
3	620	400
4	630	390
5	665	385
6	700	410
7	770	480
8	880	600
9	920	570
10	1,000	620

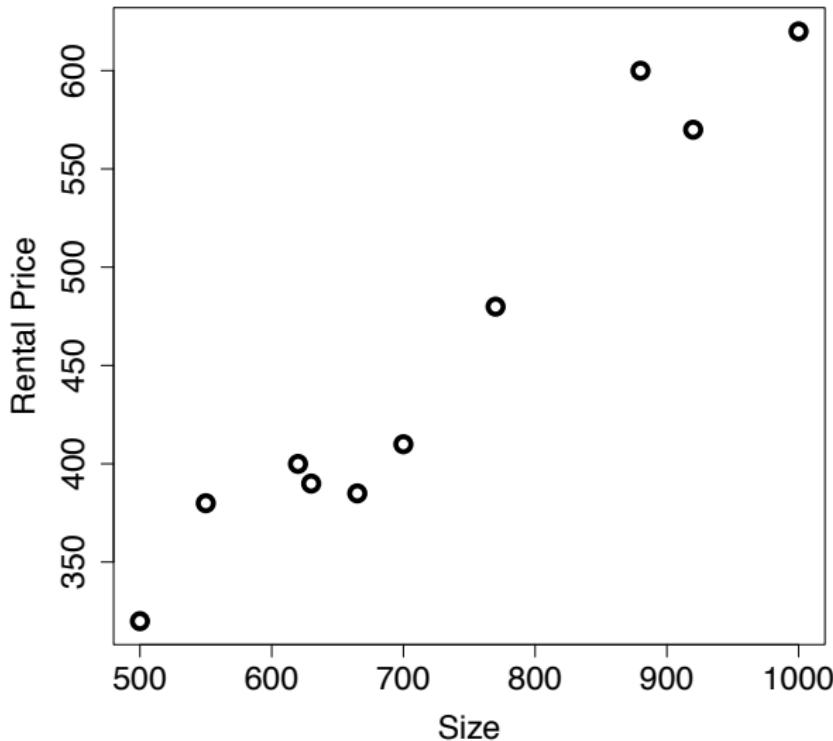


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset.

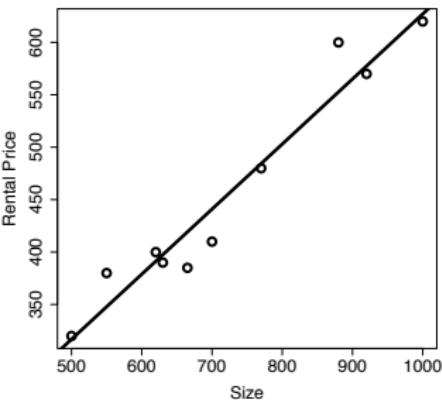
- From the scatter plot it appears that there is a linear relationship between the SIZE and RENTAL PRICE.
 - The equation of a line can be written as:

$$y = mx + b \quad (1)$$

Simple Linear Regression

- The scatter plot below shows the same scatter plot as shown in Figure 1^[8] with a simple linear model added to capture the relationship between office sizes and office rental prices.
- This model is:

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$



Simple Linear Regression

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$

- Using this model determine the expected rental price of the 730 square foot office:

Simple Linear Regression

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times \text{SIZE}$$

- Using this model determine the expected rental price of the 730 square foot office:

$$\text{RENTAL PRICE} = 6.47 + 0.62 \times 730$$

$$= 459.07$$

Simple Linear Regression

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] \quad (2)$$

Measuring Error

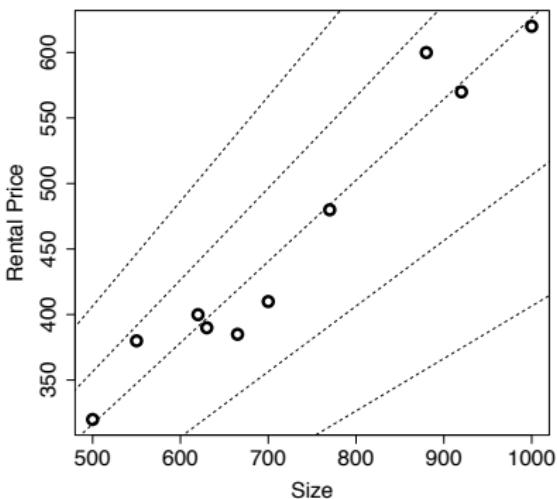


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset. A collection of possible simple linear regression models capturing the relationship between these two features are also shown. For all models $w[0]$ is set to 6.47. From top to bottom the models use 0.4, 0.5, 0.62, 0.7 and 0.8 respectively for $w[1]$.

Measuring Error

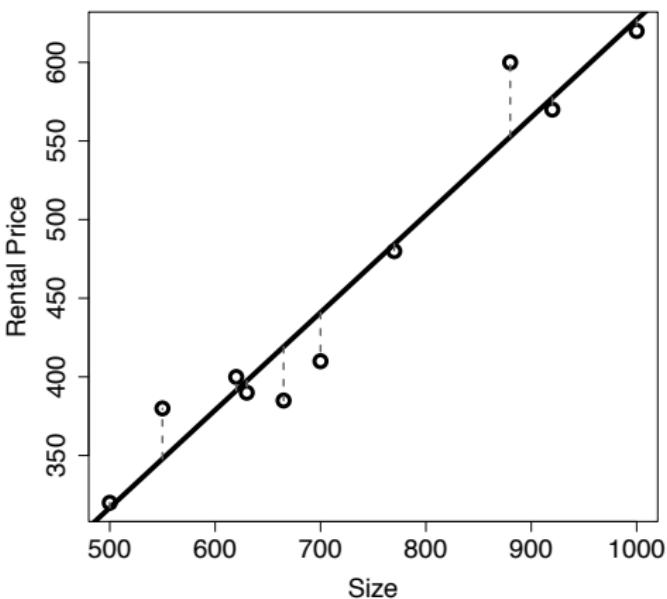


Figure: A scatter plot of the SIZE and RENTAL PRICE features from the office rentals dataset showing a candidate prediction model (with $w[0] = 6.47$ and $w[1] = 0.62$) and the resulting errors.

Measuring Error

$$L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \frac{1}{2} \sum_{i=1}^n (t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i[1]))^2 \quad (3)$$

$$= \frac{1}{2} \sum_{i=1}^n (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 \quad (4)$$

Table: Calculating the sum of squared errors for the candidate model (with $w[0] = 6.47$ and $w[1] = 0.62$) making predictions for the office rentals dataset.

ID	RENTAL PRICE	Model Prediction	Error Error	Squared Error
1	320	316.79	3.21	10.32
2	380	347.82	32.18	1,035.62
3	400	391.26	8.74	76.32
4	390	397.47	-7.47	55.80
5	385	419.19	-34.19	1,169.13
6	410	440.91	-30.91	955.73
7	480	484.36	-4.36	19.01
8	600	552.63	47.37	2,243.90
9	570	577.46	-7.46	55.59
10	620	627.11	-7.11	50.51
Sum				5,671.64
Sum of squared errors (Sum/2)				2,835.82

Error Surfaces

- For every possible combination of weights, $w[0]$ and $w[1]$, there is a corresponding sum of squared errors value that can be joined together to make a surface.

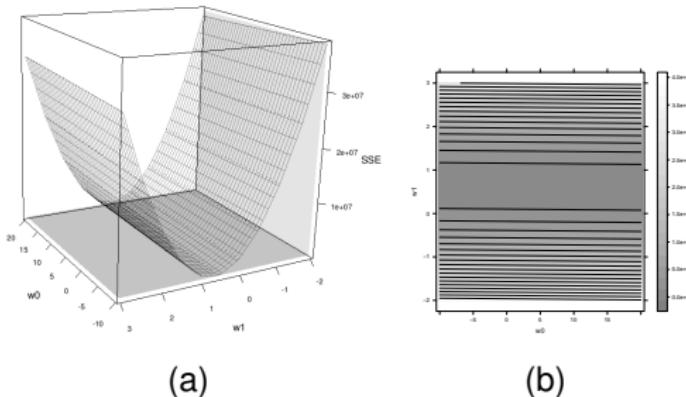


Figure: (a) A 3D surface plot and (b) a contour plot of the error surface generated by plotting the sum of squared errors value for the office rentals training set for each possible combination of values for $w[0]$ (from the range $[-10, 20]$) and $w[1]$ (from the range $[-2, 3]$).

- The x - y plane is known as a **weight space** and the surface is known as an **error surface**.
- The model that best fits the training data is the model corresponding to the lowest point on the error surface.

- Using Equation (4)^[16] we can formally define this point on the error surface as the point at which:

$$\frac{\partial}{\partial \mathbf{w}[0]} \frac{1}{2} \sum_{i=1}^n (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0 \quad (5)$$

and

$$\frac{\partial}{\partial \mathbf{w}[1]} \frac{1}{2} \sum_{i=1}^n (t_i - (\mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}_i[1]))^2 = 0 \quad (6)$$

- There are a number of different ways to find this point.
- We will describe a **guided search** approach known as the **gradient descent** algorithm.



Standard Approach: Multivariate Linear Regression with Gradient Descent

Multivariate Linear Regression

Table: A dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-center offices.

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING	RENTAL PRICE
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

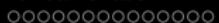


Multivariate Linear Regression

- We can define a multivariate linear regression model as:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \mathbf{w}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \cdots + \mathbf{w}[m] \times \mathbf{d}[m] \quad (7)$$

$$= \mathbf{w}[0] + \sum_{j=1}^m \mathbf{w}[j] \times \mathbf{d}[j] \quad (8)$$



Multivariate Linear Regression

- We can make Equation (8)^[23] look a little neater by inventing a dummy descriptive feature, $\mathbf{d}[0]$, that is always equal to 1:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \mathbf{w}[0] \times \mathbf{d}[0] + \mathbf{w}[1] \times \mathbf{d}[1] + \dots + \mathbf{w}[m] \times \mathbf{d}[m] \quad (10)$$

$$= \sum_{j=0}^m \mathbf{w}[j] \times \mathbf{d}[j] \quad (10)$$

$$= \mathbf{w} \cdot \mathbf{d} \quad (11)$$



Multivariate Linear Regression

- The sum of squared errors loss function, L_2 , definition that we gave in Equation (4)^[16] changes only very slightly to reflect the new regression equation:

$$L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \frac{1}{2} \sum_{i=1}^n (t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i))^2 \quad (12)$$

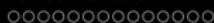
$$= \frac{1}{2} \sum_{i=1}^n (t_i - (\mathbf{w} \cdot \mathbf{d}_i))^2 \quad (13)$$



Multivariate Linear Regression

- This multivariate model allows us to include all but one of the descriptive features in Table 3 [17] in a regression model to predict office rental prices.
- The resulting multivariate regression model equation is:

$$\begin{aligned}\text{RENTAL PRICE} = & \mathbf{w}[0] + \mathbf{w}[1] \times \text{SIZE} + \mathbf{w}[2] \times \text{FLOOR} \\ & + \mathbf{w}[3] \times \text{BROADBAND RATE}\end{aligned}$$



Multivariate Linear Regression

- We will see in the next section how the best-fit set of weights for this equation are found, but for now we will set:
 - $w[0] = -0.1513$,
 - $w[1] = 0.6270$,
 - $w[2] = -0.1781$,
 - $w[3] = 0.0714$.
- This means that the model is rewritten as:

$$\begin{aligned}\text{RENTAL PRICE} = & -0.1513 + 0.6270 \times \text{SIZE} \\ & - 0.1781 \times \text{FLOOR} \\ & + 0.0714 \times \text{BROADBAND RATE}\end{aligned}$$



Multivariate Linear Regression

- Using this model:

$$\begin{aligned} \text{RENTAL PRICE} = & -0.1513 + 0.6270 \times \text{SIZE} \\ & - 0.1781 \times \text{FLOOR} \\ & + 0.0714 \times \text{BROADBAND RATE} \end{aligned}$$

- we can, for example, predict the expected rental price of a 690 square foot office on the 11th floor of a building with a broadband rate of 50 Mb per second as:

$$\text{RENTAL PRICE} = ?$$



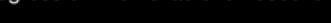
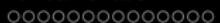
Multivariate Linear Regression

- Using this model:

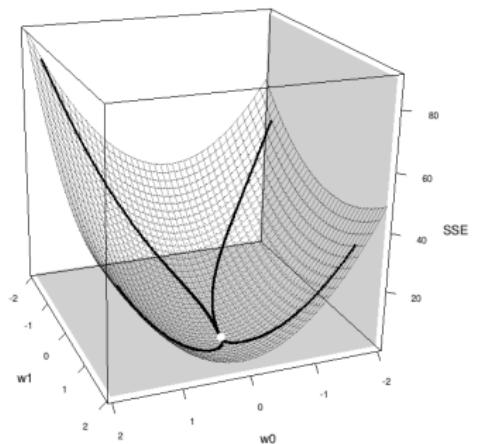
$$\begin{aligned}\text{RENTAL PRICE} = & -0.1513 + 0.6270 \times \text{SIZE} \\ & - 0.1781 \times \text{FLOOR} \\ & + 0.0714 \times \text{BROADBAND RATE}\end{aligned}$$

- we can, for example, predict the expected rental price of a 690 square foot office on the 11th floor of a building with a broadband rate of 50 Mb per second as:

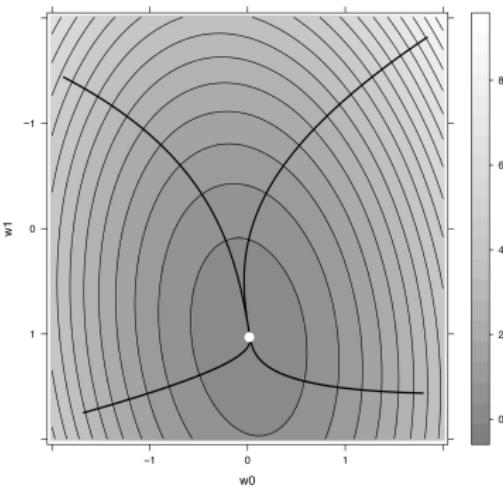
$$\begin{aligned}\text{RENTAL PRICE} &= -0.1513 + 0.6270 \times 690 \\ &\quad - 0.1781 \times 11 + 0.0714 \times 50 \\ &= 434.0896\end{aligned}$$



Gradient Descent



(a)



(b)

Figure: (a) A 3D surface plot and (b) a contour plot of the same error surface. The lines indicate the path that the gradient decent algorithm would take across this error surface from different starting positions to the global minimum - marked as the white dot in the centre.

- The journey across the error surface that is taken by the gradient descent algorithm when training the simple version of the office rentals example - involving just SIZE and RENTAL PRICE.

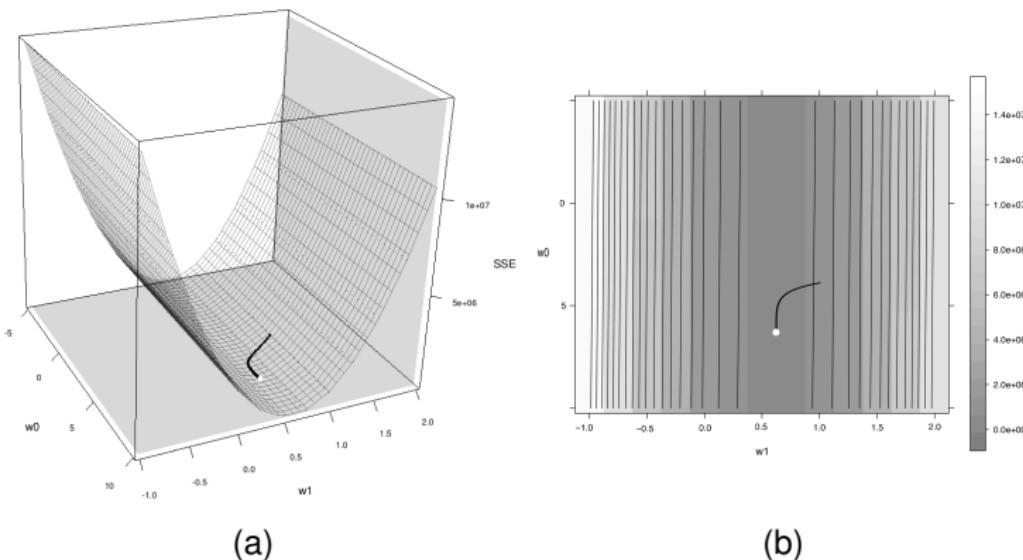


Figure: (a) A 3D surface plot and (b) a contour plot of the error surface for the office rentals dataset showing the path that the gradient descent algorithm takes towards the best fit model.

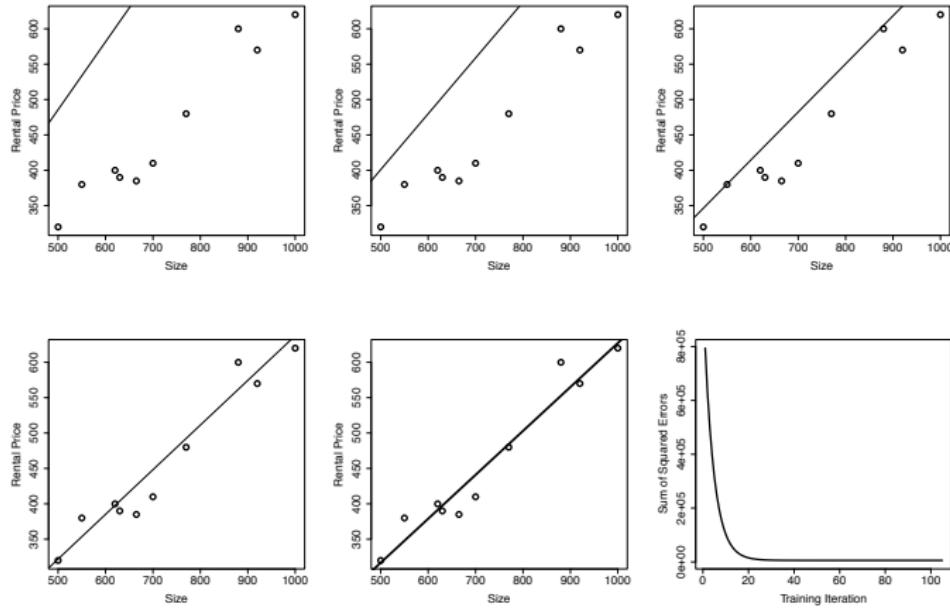


Figure: A selection of the simple linear regression models developed during the gradient descent process for the office rentals dataset. The final panel shows the sum of squared error values generated during the gradient descent process.

Require: set of training instances \mathcal{D}

Require: a learning rate α that controls how quickly the algorithm converges

Require: a function, **errorDelta**, that determines the direction in which to adjust a given weight, $\mathbf{w}[j]$, so as to move down the slope of an error surface determined by the dataset, \mathcal{D}

Require: a convergence criterion that indicates that the algorithm has completed

1: $\mathbf{w} \leftarrow$ random starting point in the weight space

2: **repeat**

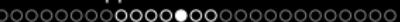
3: **for** each $\mathbf{w}[j]$ in \mathbf{w} **do**

4: $\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \text{errorDelta}(\mathcal{D}, \mathbf{w}[j])$

5: **end for**

6: **until** convergence occurs

- The gradient descent algorithm for training multivariate linear regression models.

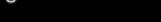


Gradient Descent

- The most important part to the gradient descent algorithm is Line Rule 4 on which the weights are updated.

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \text{errorDelta}(\mathcal{D}, \mathbf{w}[j])$$

- Each weight is considered independently and for each one a small adjustment is made by adding a small **delta** value to the current weight, $\mathbf{w}[j]$.
- This adjustment should ensure that the change in the weight leads to a move *downwards* on the error surface.



Gradient Descent

- Imagine for a moment that our training dataset, \mathcal{D} contains **just one training** example: (\mathbf{d}, t)
- The gradient of the error surface is given as the partial derivative of L_2 with respect to each weight, $\mathbf{w}[j]$:

$$\frac{\partial}{\partial \mathbf{w}[j]} L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \frac{\partial}{\partial \mathbf{w}[j]} \left(\frac{1}{2} (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d}))^2 \right) \quad (14)$$

$$= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times \frac{\partial}{\partial \mathbf{w}[j]} (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \quad (15)$$

$$= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times \frac{\partial}{\partial \mathbf{w}[j]} (t - (\mathbf{w} \cdot \mathbf{d})) \quad (16)$$

$$= (t - \mathbb{M}_{\mathbf{w}}(\mathbf{d})) \times -\mathbf{d}[j] \quad (17)$$

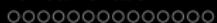
Gradient Descent

- Adjusting the calculation to take into account multiple training instances:

$$\frac{\partial}{\partial \mathbf{w}[j]} L_2(\mathbb{M}_{\mathbf{w}}, \mathcal{D}) = \sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

- We use this equation to define the **errorDelta** in our gradient descent algorithm.

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{\text{errorDelta}(\mathcal{D}, \mathbf{w}[j])}$$



Choosing Learning Rates & Initial Weights

- The learning rate, α , determines the size of the adjustment made to each weight at each step in the process.
- Unfortunately, choosing learning rates is not a well defined science.
- Most practitioners use rules of thumb and trial and error.

Choosing Learning Rates & Initial Weights

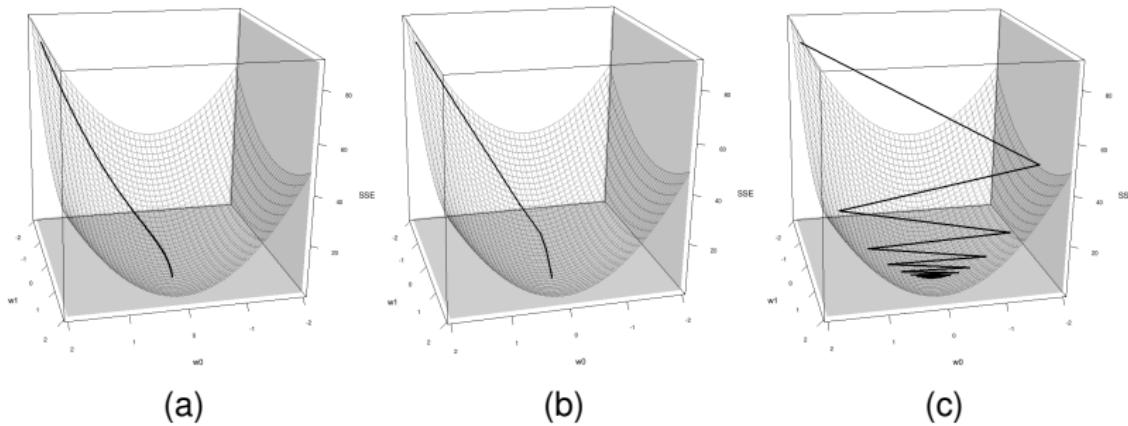


Figure: Plots of the journeys made across the error surface for the simple office rentals prediction problem for different learning rates: (a) a very small learning rate (0.002), (b) a medium learning rate (0.08) and (c) a very large learning rate (0.18).



Choosing Learning Rates & Initial Weights

- A typical range for learning rates is $[0.00001, 10]$
- Based on empirical evidence, choosing random initial weights uniformly from the range $[-0.2, 0.2]$ tends to work well.



A Worked Example

- We are now in a position to build a linear regression model that uses all of the continuous descriptive features in the office rentals dataset.
- The general structure of the model is:

$$\begin{aligned}\text{RENTAL PRICE} = & \mathbf{w}[0] + \mathbf{w}[1] \times \text{SIZE} + \mathbf{w}[2] \times \text{FLOOR} \\ & + \mathbf{w}[3] \times \text{BROADBAND RATE}\end{aligned}$$

A Worked Example

Table: The **office rentals dataset**: a dataset that includes office rental prices and a number of descriptive features for 10 Dublin city-centre offices.

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING	RENTAL PRICE
1	500	4	8	C	320
2	550	7	50	A	380
3	620	9	7	A	400
4	630	5	24	B	390
5	665	8	100	C	385
6	700	4	8	B	410
7	770	10	7	B	480
8	880	12	50	A	600
9	920	14	8	C	570
10	1,000	9	24	B	620

A Worked Example

- For this example let's assume that:
 - $\alpha = 0.00000002$

Initial Weights

w[0]:	-0.146	w[1]:	0.185	w[2]:	-0.044	w[3]:	0.119
-------	--------	-------	-------	-------	--------	-------	-------

A Worked Example

Iteration 1

ID	RENTAL PRICE			Squared Error	$\text{errorDelta}(\mathcal{D}, \mathbf{w}[i])$			
		Pred.	Error		$\mathbf{w}[0]$	$\mathbf{w}[1]$	$\mathbf{w}[2]$	$\mathbf{w}[3]$
1	320	93.26	226.74	51411.08	226.74	113370.05	906.96	1813.92
2	380	107.41	272.59	74307.70	272.59	149926.92	1908.16	13629.72
3	400	115.15	284.85	81138.96	284.85	176606.39	2563.64	1993.94
4	390	119.21	270.79	73327.67	270.79	170598.22	1353.95	6498.98
5	385	134.64	250.36	62682.22	250.36	166492.17	2002.91	25036.42
6	410	130.31	279.69	78226.32	279.69	195782.78	1118.76	2237.52
7	480	142.89	337.11	113639.88	337.11	259570.96	3371.05	2359.74
8	600	168.32	431.68	186348.45	431.68	379879.24	5180.17	21584.05
9	570	170.63	399.37	159499.37	399.37	367423.83	5591.23	3194.99
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
				Sum	1067571.59	3185.61	2412073.90	27888.65
				Sum of squared errors (Sum/2)	533785.80			88727.43



A Worked Example

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{errorDelta(\mathcal{D}, \mathbf{w}[j])}$$

Initial Weights

$\mathbf{w}[0]: -0.146 \quad \mathbf{w}[1]: 0.185 \quad \mathbf{w}[2]: -0.044 \quad \mathbf{w}[3]: 0.119$

Example

$$\mathbf{w}[1] \leftarrow 0.185 + 0.00000002 \times 2,412,074 = 0.23324148$$

New Weights (Iteration 1)

$\mathbf{w}[0]: -0.146 \quad \mathbf{w}[1]: 0.233 \quad \mathbf{w}[2]: -0.043 \quad \mathbf{w}[3]: 0.121$

A Worked Example

Iteration 2

ID	PRICE	RENTAL		Squared Error	errorDelta($\mathcal{D}, \mathbf{w}[i]$)			
		Pred.	Error		$w[0]$	$w[1]$	$w[2]$	$w[3]$
1	320	117.40	202.60	41047.92	202.60	101301.44	810.41	1620.82
2	380	134.03	245.97	60500.69	245.97	135282.89	1721.78	12298.44
3	400	145.08	254.92	64985.12	254.92	158051.51	2294.30	1784.45
4	390	149.65	240.35	57769.68	240.35	151422.55	1201.77	5768.48
5	385	166.90	218.10	47568.31	218.10	145037.57	1744.81	21810.16
6	410	164.10	245.90	60468.86	245.90	172132.91	983.62	1967.23
7	480	180.06	299.94	89964.69	299.94	230954.68	2999.41	2099.59
8	600	210.87	389.13	151424.47	389.13	342437.01	4669.60	19456.65
9	570	215.03	354.97	126003.34	354.97	326571.94	4969.57	2839.76
10	620	187.58	432.42	186989.95	432.42	432423.35	3891.81	10378.16
		Sum		886723.04	2884.32	2195615.84	25287.08	80023.74
Sum of squared errors (Sum/2)		443361.52						

A Worked Example

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{errorDelta(\mathcal{D}, \mathbf{w}[j])}$$

Initial Weights (Iteration 2)

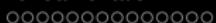
$\mathbf{w}[0]:$	-0.146	$\mathbf{w}[1]:$	0.233	$\mathbf{w}[2]:$	-0.043	$\mathbf{w}[3]:$	0.121
------------------	--------	------------------	-------	------------------	--------	------------------	-------

Exercise

$$\mathbf{w}[1] \leftarrow ?, \alpha = 0.00000002$$

New Weights (Iteration 2)

$\mathbf{w}[0]:$?	$\mathbf{w}[1]:$?	$\mathbf{w}[2]:$?	$\mathbf{w}[3]:$?
------------------	---	------------------	---	------------------	---	------------------	---



A Worked Example

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \underbrace{\sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times d_i[j])}_{errorDelta(\mathcal{D}, \mathbf{w}[j])}$$

Initial Weights (Iteration 2)

$\mathbf{w}[0]:$	-0.146	$\mathbf{w}[1]:$	0.233	$\mathbf{w}[2]:$	-0.043	$\mathbf{w}[3]:$	0.121
------------------	--------	------------------	-------	------------------	--------	------------------	-------

Exercise

$$\mathbf{w}[1] \leftarrow -0.233 + 0.00000002 \times 2195615.84 = 0.27691232$$

New Weights (Iteration 2)

$\mathbf{w}[0]:$	-0.145	$\mathbf{w}[1]:$	0.277	$\mathbf{w}[2]:$	-0.043	$\mathbf{w}[3]:$	0.123
------------------	--------	------------------	-------	------------------	--------	------------------	-------



A Worked Example

- The algorithm then keeps iteratively applying the weight update rule until it converges on a stable set of weights beyond which little improvement in model accuracy is possible.
- After 100 iterations the final values for the weights are:
 - $w[0] = -0.1513$,
 - $w[1] = 0.6270$,
 - $w[2] = -0.1781$
 - $w[3] = 0.0714$
- which results in a sum of squared errors value of 2,913.5

1 Big Idea

2 Fundamentals

- Simple Linear Regression
- Measuring Error
- Error Surfaces

3 Standard Approach: Multivariate Linear Regression with Gradient Descent

- Multivariate Linear Regression
- Gradient Descent
- Choosing Learning Rates & Initial Weights
- A Worked Example

4 Summary

Fundamentals of Machine Learning for Predictive Data Analytics

Chapter 7: Error-based Learning Sections 7.4, 7.5

John Kelleher and Brian Mac Namee and Aoife D'Arcy

john.d.kelleher@dit.ie

brian.macnamee@ucd.ie

aoife@theanalyticsstore.com

- 1 Interpreting Multivariable Linear Regression Models
- 2 Setting the Learning Rate Using Weight Decay
- 3 Handling Categorical Descriptive Features
- 4 Handling Categorical Target Features: Logistic Regression
- 5 Modeling Non-linear Relationships
- 6 Multinomial Logistic Regression
- 7 Support Vector Machines

Interpreting Multivariable Linear Regression Models

- The weights used by linear regression models indicate the effect of each descriptive feature on the predictions returned by the model.
- Both the **sign** and the **magnitude** of the weight provide information on how the descriptive feature effects the predictions of the model.

Table: Weights and standard errors for each feature in the office rentals model.

Descriptive Feature	Weight	Standard Error	t-statistic	p-value
SIZE	0.6270	0.0545	11.504	<0.0001
FLOOR	-0.1781	2.7042	-0.066	0.949
BROADBAND RATE	0.071396	0.2969	0.240	0.816

- It is tempting to infer the relative importance of the different descriptive features in the model from the magnitude of the weights
- However, direct comparison of the weights tells us little about their relative importance.
- A better way to determine the importance of each descriptive feature in the model is to perform a **statistical significance test**.

- The statistical significance test we use to analyze the importance of a descriptive feature $\mathbf{d}[j]$ in a linear regression model is the ***t-test***.
- The null hypothesis for this test is that the feature does not have a significant impact on the model. The test statistic we calculate is called the *t*-statistic.

- The standard error for the overall model is calculated as

$$se = \sqrt{\frac{\sum_{i=1}^n (t_i - M_{\mathbf{w}}(\mathbf{d}_i))^2}{n - 2}} \quad (1)$$

- A standard error calculation is then done for a descriptive feature as follows:

$$se(\mathbf{d}[j]) = \frac{se}{\sqrt{\sum_{i=1}^n (\mathbf{d}_i[j] - \bar{\mathbf{d}}[j])^2}} \quad (2)$$

- The t -statistic for this test is calculated as follows:

$$t = \frac{\mathbf{w}[j]}{se(\mathbf{d}[j])} \quad (3)$$

- Using a standard t -statistic look-up table, we can then determine the p -value associated with this test (this is a two tailed t -test with degrees of freedom set to the number of instances in the training set minus 2).
- If the p -value is less than the required significance level, typically 0.05, we reject the null hypothesis and say that the descriptive feature has a significant impact on the model; otherwise we say that it does not.

Table: Weights and standard errors for each feature in the office rentals model.

Descriptive Feature	Weight	Standard Error	t -statistic	p -value
SIZE	0.6270	0.0545	11.504	<0.0001
FLOOR	-0.1781	2.7042	-0.066	0.949
BROADBAND RATE	0.071396	0.2969	0.240	0.816

Setting the Learning Rate Using Weight Decay

- **Learning rate decay** allows the learning rate to start at a large value and then decay over time according to a predefined schedule.
- A good approach is to use the following decay schedule:

$$\alpha_\tau = \alpha_0 \frac{c}{c + \tau} \quad (4)$$

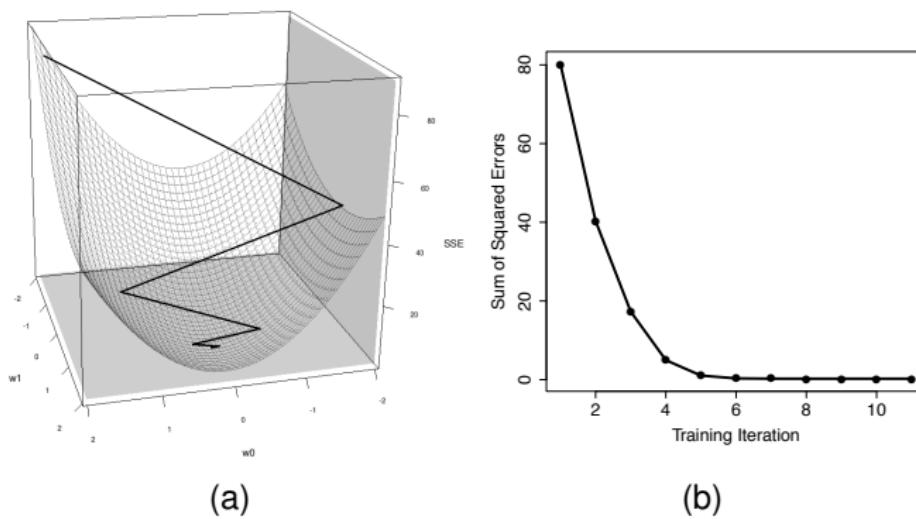


Figure: (a) The journey across the error surface for the office rentals prediction problem when learning rate decay is used ($\alpha_0 = 0.18$, $c = 10$); (b) a plot of the changing sum of squared error values during this journey.

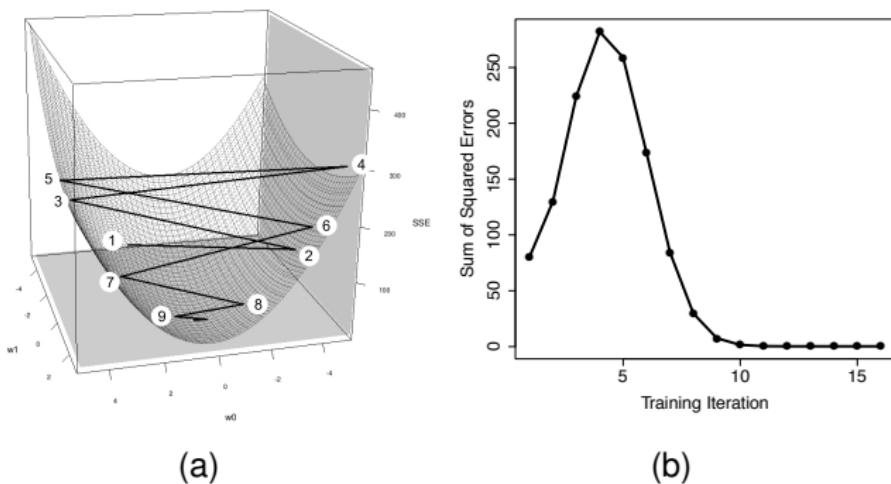


Figure: (a) The journey across the error surface for the office rentals prediction problem when learning rate decay is used ($\alpha_0 = 0.25$, $c = 100$); (b) a plot of the changing sum of squared error values during this journey.

Handling Categorical Descriptive Features

- The basic structure of the multivariable linear regression model allows for only continuous descriptive features, so we need a way to handle categorical descriptive features.
- The most common approach to handling categorical features uses a transformation that converts a single categorical descriptive feature into a number of continuous descriptive feature values that can encode the levels of the categorical feature.
- For example, the ENERGY RATING descriptive feature would be converted into three new continuous descriptive features, as it has 3 distinct levels: 'A', 'B', or 'C'.

Table: The office rentals dataset adjusted to handle the categorical ENERGY RATING descriptive feature in linear regression models.

ID	SIZE	FLOOR	BROADBAND RATE	ENERGY RATING A	ENERGY RATING B	ENERGY RATING C	RENTAL PRICE
1	500	4	8	0	0	1	320
2	550	7	50	1	0	0	380
3	620	9	7	1	0	0	400
4	630	5	24	0	1	0	390
5	665	8	100	0	0	1	385
6	700	4	8	0	1	0	410
7	770	10	7	0	1	0	480
8	880	12	50	1	0	0	600
9	920	14	8	0	0	1	570
10	1 000	9	24	0	1	0	620

- Returning to our example, the regression equation for this RENTAL PRICE model would change to

$$\begin{aligned}\text{RENTAL PRICE} = & \mathbf{w}[0] + \mathbf{w}[1] \times \text{SIZE} + \mathbf{w}[2] \times \text{FLOOR} \\ & + \mathbf{w}[3] \times \text{BROADBAND RATE} \\ & + \mathbf{w}[4] \times \text{ENERGY RATING A} \\ & + \mathbf{w}[5] \times \text{ENERGY RATING B} \\ & + \mathbf{w}[6] \times \text{ENERGY RATING C}\end{aligned}$$

where the newly added categorical features allow the original ENERGY RATING feature to be included.

Handling Categorical Target Features: Logistic Regression

Table: A dataset listing features for a number of generators.

ID	RPM	VIBRATION	STATUS	ID	RPM	VIBRATION	STATUS
1	568	585	good	29	562	309	faulty
2	586	565	good	30	578	346	faulty
3	609	536	good	31	593	357	faulty
4	616	492	good	32	626	341	faulty
5	632	465	good	33	635	252	faulty
6	652	528	good	34	658	235	faulty
7	655	496	good	35	663	299	faulty
8	660	471	good	36	677	223	faulty
9	688	408	good	37	685	303	faulty
10	696	399	good	38	698	197	faulty
11	708	387	good	39	699	311	faulty
12	701	434	good	40	712	257	faulty
13	715	506	good	41	722	193	faulty
14	732	485	good	42	735	259	faulty
15	731	395	good	43	738	314	faulty
16	749	398	good	44	753	113	faulty
17	759	512	good	45	767	286	faulty
18	773	431	good	46	771	264	faulty
19	782	456	good	47	780	137	faulty
20	797	476	good	48	784	131	faulty
21	794	421	good	49	798	132	faulty
22	824	452	good	50	820	152	faulty
23	835	441	good	51	834	157	faulty
24	862	372	good	52	858	163	faulty
25	879	340	good	53	888	91	faulty
26	892	370	good	54	891	156	faulty
27	913	373	good	55	911	79	faulty
28	933	330	good	56	939	99	faulty

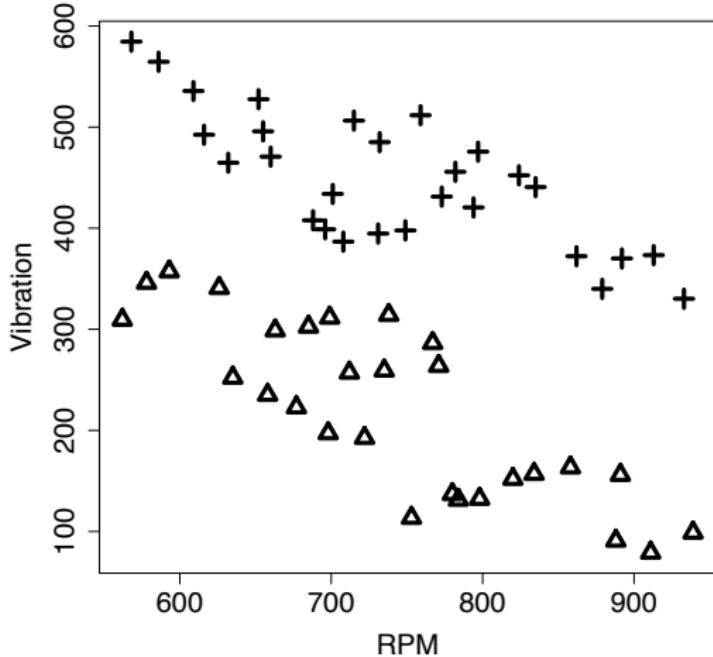


Figure: A scatter plot of the RPM and VIBRATION descriptive features from the generators dataset shown in Table 4 [18] where 'good' generators are shown as crosses and 'faulty' generators are shown as triangles.

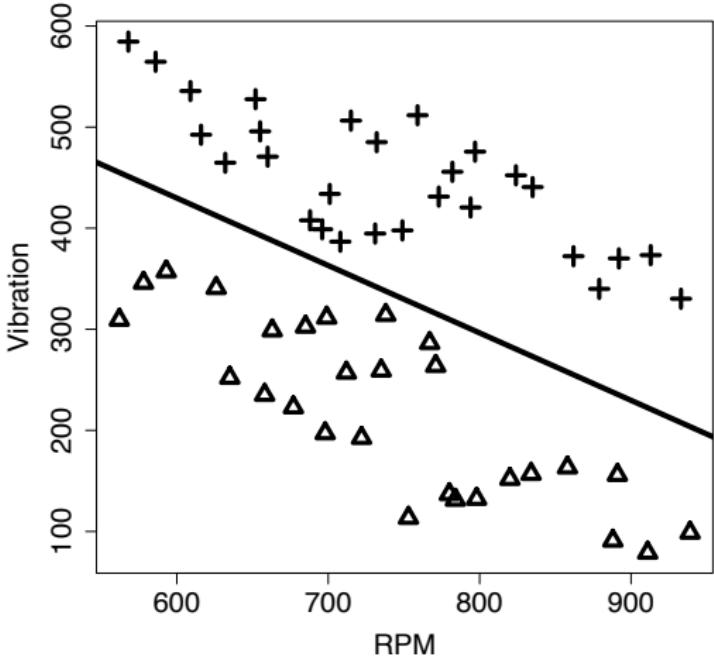


Figure: A scatter plot of the RPM and VIBRATION descriptive features from the generators dataset shown in Table 4 [18]. A decision boundary separating '*good*' generators (crosses) from '*faulty*' generators (triangles) is also shown.

- As the decision boundary is a **linear separator** it can be defined using the equation of the line as:

$$\text{VIBRATION} = 830 - 0.667 \times \text{RPM} \quad (5)$$

or

$$830 - 0.667 \times \text{RPM} - \text{VIBRATION} = 0 \quad (6)$$

- Applying Equation (6)^[21] to the instance $\text{RPM} = 810$, $\text{VIBRATION} = 495$, which is above the decision boundary, gives the following result:

$$830 - 0.667 \times 810 - 495 = -205.27$$

- By contrast, if we apply Equation (6)^[21] to the instance $\text{RPM} = 650$ and $\text{VIBRATION} = 240$, which is below the decision boundary, we get

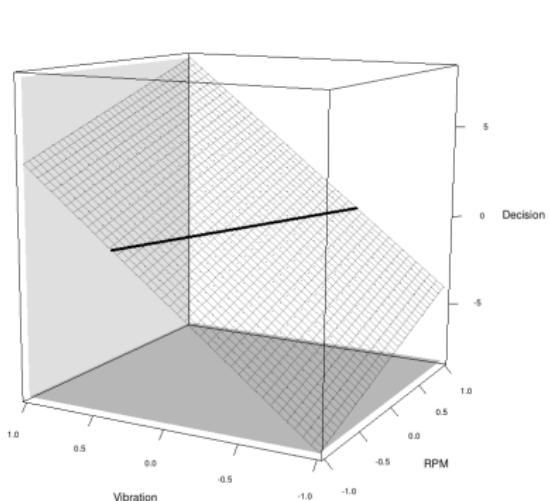
$$830 - 0.667 \times 650 - 240 = 156.45$$

- All the data points above the decision boundary will result in a negative value when plugged into the decision boundary equation, while all data points below the decision boundary will result in a positive value.

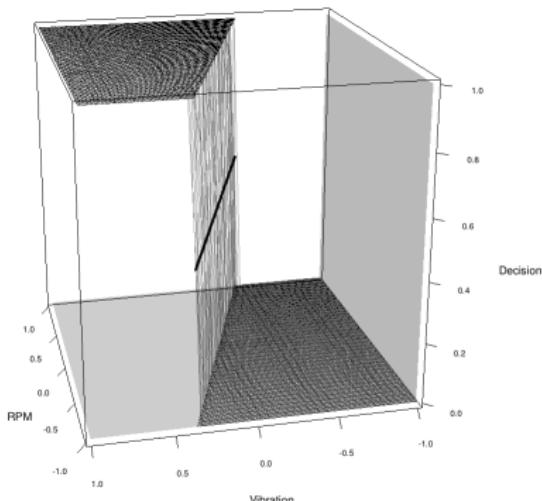
- Reverting to our previous notation we have:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{d} \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

- The surface defined by this rule is known as a **decision surface**.



(a)



(b)

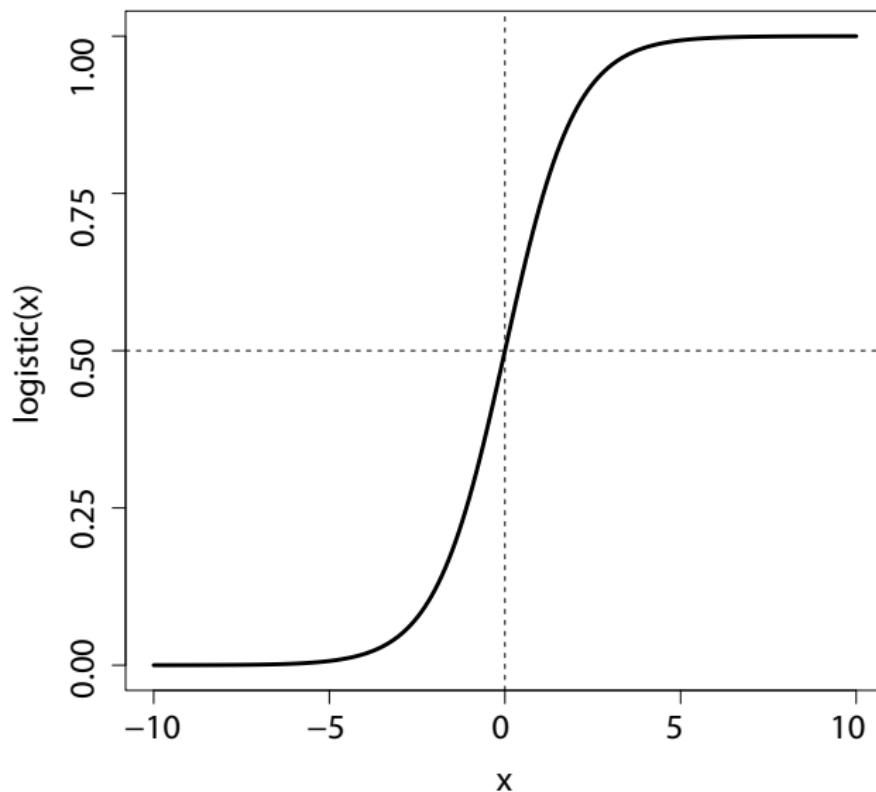
Figure: (a) A surface showing the value of Equation (6)^[21] for all values of RPM and VIBRATION. The decision boundary given in Equation (6)^[21] is highlighted. (b) The same surface linearly thresholded at zero to operate as a predictor.

- The hard decision boundary given in Equation (7)^[24] is **discontinuous** so is not differentiable and so we can't calculate the gradient of the error surface.
- Furthermore, the model always makes completely confident predictions of 0 or 1, whereas a little more subtlety is desirable.
- We address these issues by using a more sophisticated threshold function that is continuous, and therefore differentiable, and that allows for the subtlety desired: the **logistic function**

logistic function

$$\text{Logistic}(x) = \frac{1}{1 + e^{-x}} \quad (8)$$

where x is a numeric value and e is **Euler's number** and is approximately equal to 2.7183.



- To build a logistic regression model, we simply pass the output of the basic linear regression model through the logistic function

$$\begin{aligned} M_{\mathbf{w}}(\mathbf{d}) &= \text{Logistic}(\mathbf{w} \cdot \mathbf{d}) \\ &= \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{d}}} \end{aligned} \tag{9}$$

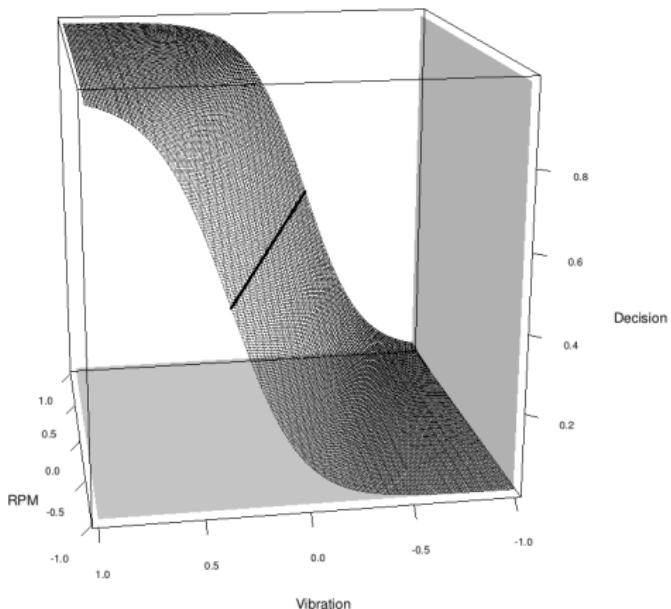
A note on training logistic regression models:

- Before we train a logistic regression model we map the binary target feature levels to 0 or 1.
- The error of the model on each instance is then the difference between the target feature (0 or 1) and the value of the prediction [0, 1].

Example

$$\mathbb{M}_w(\langle \text{RPM}, \text{VIBRATION} \rangle)$$

$$= \frac{1}{1 + e^{(-0.4077 + 4.1697 \times \text{RPM} + 6.0460 \times \text{VIBRATION})}}$$



- The decision surface for the example logistic regression model.

$$P(t = \text{'faulty'} | \mathbf{d}) = \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

$$P(t = \text{'good'} | \mathbf{d}) = 1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d})$$

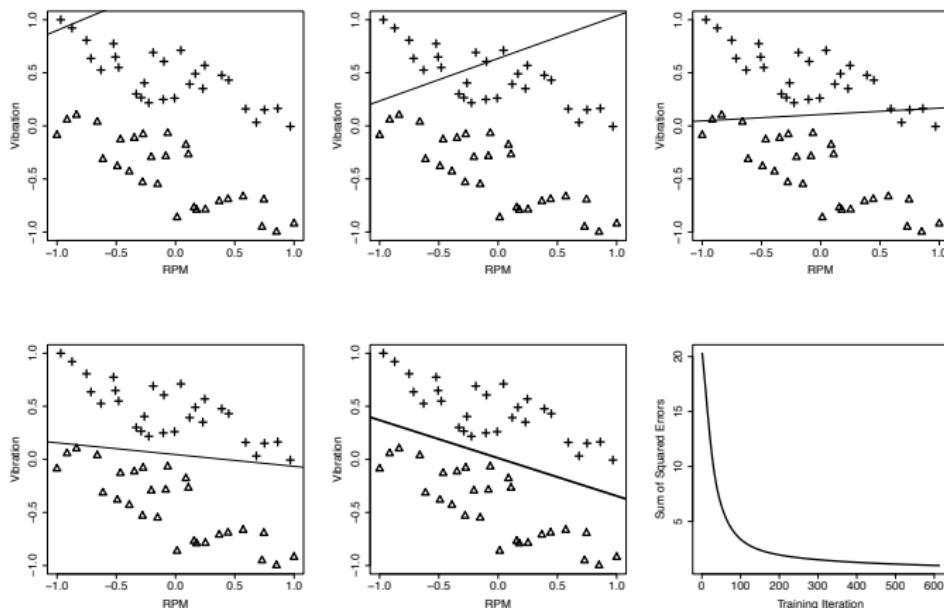


Figure: A selection of the logistic regression models developed during the gradient descent process for the machinery dataset from Table 4 [18]. The bottom-right panel shows the sum of squared error values generated during the gradient descent process.

- To repurpose the gradient descent algorithm for training logistic regression models the only change that needs to be made is i in the weight update rule.
- See pg. 360 in book for details of how to derive the new weight update rule.
- The new weight update rule is:

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^n ((t - M_{\mathbf{w}}(\mathbf{d}_i)) \times M_{\mathbf{w}}(\mathbf{d}_i) \times (1 - M_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

ID	RPM	VIBRATION	STATUS	ID	RPM	VIBRATION	STATUS
1	498	604	faulty	35	501	463	good
2	517	594	faulty	36	526	443	good
3	541	574	faulty	37	536	412	good
4	555	587	faulty	38	564	394	good
5	572	537	faulty	39	584	398	good
6	600	553	faulty	40	602	398	good
7	621	482	faulty	41	610	428	good
8	632	539	faulty	42	638	389	good
9	656	476	faulty	43	652	394	good
10	653	554	faulty	44	659	336	good
11	679	516	faulty	45	662	364	good
12	688	524	faulty	46	672	308	good
13	684	450	faulty	47	691	248	good
14	699	512	faulty	48	694	401	good
15	703	505	faulty	49	718	313	good
16	717	377	faulty	50	720	410	good
17	740	377	faulty	51	723	389	good
18	749	501	faulty	52	744	227	good
19	756	492	faulty	53	741	397	good
20	752	381	faulty	54	770	200	good
21	762	508	faulty	55	764	370	good
22	781	474	faulty	56	790	248	good
23	781	480	faulty	57	786	344	good
24	804	460	faulty	58	792	290	good
25	828	346	faulty	59	818	268	good
26	830	366	faulty	60	845	232	good
27	864	344	faulty	61	867	195	good
28	882	403	faulty	62	878	168	good
29	891	338	faulty	63	895	218	good
30	921	362	faulty	64	916	221	good
31	941	301	faulty	65	950	156	good
32	965	336	faulty	66	956	174	good
33	976	297	faulty	67	973	134	good
34	994	287	faulty	68	1002	121	good

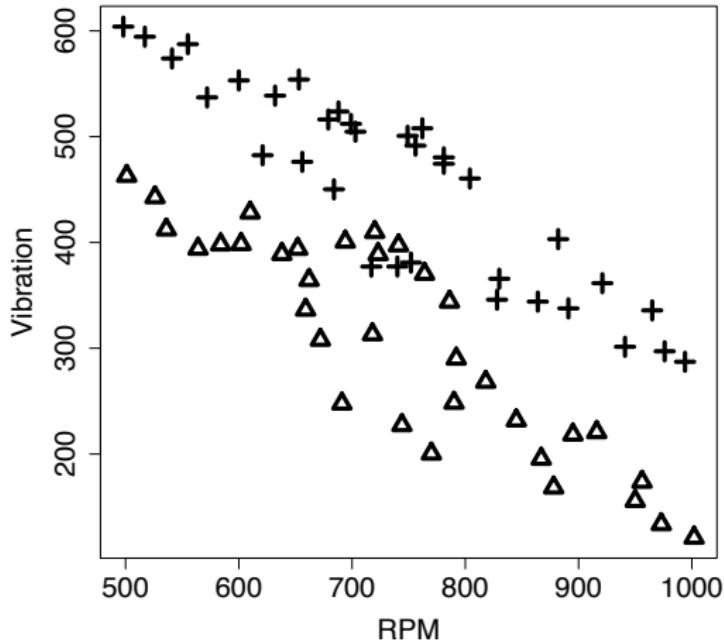


Figure: A scatter plot of the extended generators dataset given in Table 35^[35], which results in instances with the different target levels overlapping with each other. 'good' generators are shown as crosses, and 'faulty' generators are shown as triangles.

- For logistic regression models we recommend that descriptive feature values always be normalized.
- In this example, before the training process begins, both descriptive features are normalized to the range $[-1, 1]$.

- For this example let's assume that:
 - $\alpha = 0.02$

Initial Weights

$w[0]: -2.9465 \quad w[1]: -1.0147 \quad w[2]: -2.1610$

Iteration 1

ID	TARGET LEVEL	Squared Error			$\text{errorDelta}(\mathcal{D}, \mathbf{w}[i])$		
		Pred.	Error	Error	$\mathbf{w}[0]$	$\mathbf{w}[1]$	$\mathbf{w}[2]$
1	1	0.5570	0.4430	0.1963	0.1093	-0.1093	0.1093
2	1	0.5168	0.4832	0.2335	0.1207	-0.1116	0.1159
3	1	0.4469	0.5531	0.3059	0.1367	-0.1134	0.1197
4	1	0.4629	0.5371	0.2885	0.1335	-0.1033	0.1244
...							
65	0	0.0037	-0.0037	0.0000	0.0000	0.0000	0.0000
66	0	0.0042	-0.0042	0.0000	0.0000	0.0000	0.0000
67	0	0.0028	-0.0028	0.0000	0.0000	0.0000	0.0000
68	0	0.0022	-0.0022	0.0000	0.0000	0.0000	0.0000
Sum				24.4738	2.7031	-0.7015	1.6493
Sum of squared errors (Sum/2)				12.2369			

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times (1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

New Weights (after Iteration 1)

$\mathbf{w}[0]:$	-2.8924	$\mathbf{w}[1]:$	-1.0287	$\mathbf{w}[2]:$	-2.1940
------------------	---------	------------------	---------	------------------	---------

Iteration 2

ID	TARGET LEVEL	Squared Error			$\text{errorDelta}(\mathcal{D}, \mathbf{w}[i])$		
		Pred.	Error	Error	$\mathbf{w}[0]$	$\mathbf{w}[1]$	$\mathbf{w}[2]$
1	1	0.5817	0.4183	0.1749	0.1018	-0.1018	0.1018
2	1	0.5414	0.4586	0.2103	0.1139	-0.1053	0.1094
3	1	0.4704	0.5296	0.2805	0.1319	-0.1094	0.1155
4	1	0.4867	0.5133	0.2635	0.1282	-0.0992	0.1194
...							
65	0	0.0037	-0.0037	0.0000	0.0000	0.0000	0.0000
66	0	0.0043	-0.0043	0.0000	0.0000	0.0000	0.0000
67	0	0.0028	-0.0028	0.0000	0.0000	0.0000	0.0000
68	0	0.0022	-0.0022	0.0000	0.0000	0.0000	0.0000
Sum				24.0524	2.7236	-0.6646	1.6484
Sum of squared errors (Sum/2)				12.0262			

$$\mathbf{w}[j] \leftarrow \mathbf{w}[j] + \alpha \times \sum_{i=1}^n ((t_i - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i) \times (1 - \mathbb{M}_{\mathbf{w}}(\mathbf{d}_i)) \times \mathbf{d}_i[j])$$

New Weights (after Iteration 2)

$\mathbf{w}[0]:$	-2.8380	$\mathbf{w}[1]:$	-1.0416	$\mathbf{w}[2]:$	-2.2271
------------------	---------	------------------	---------	------------------	---------

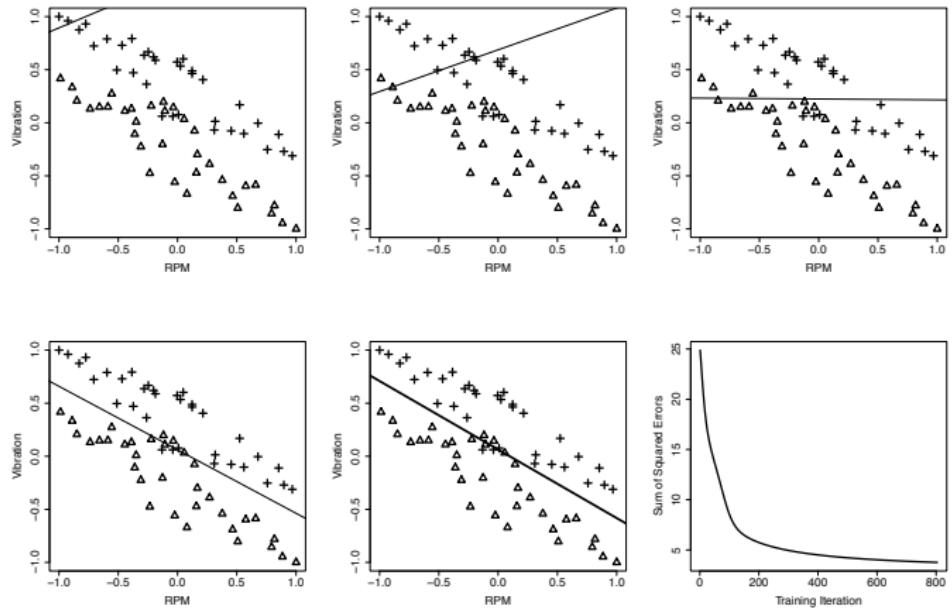


Figure: A selection of the logistic regression models developed during the gradient descent process for the extended generators dataset in Table 35 [35]. The bottom-right panel shows the sum of squared error values generated during the gradient descent process.

- The final model found is:

$$\mathbb{M}_w(\langle \text{RPM}, \text{VIBRATION} \rangle)$$

$$= \frac{1}{1 + e^{(-0.4077 + 4.1697 \times \text{RPM} + 6.0460 \times \text{VIBRATION})}}$$

Modeling Non-linear Relationships

Table: A dataset describing grass growth on Irish farms during July 2012.

ID	RAIN	GROWTH	ID	RAIN	GROWTH	ID	RAIN	GROWTH
1	2.153	14.016	12	3.754	11.420	23	3.960	10.307
2	3.933	10.834	13	2.809	13.847	24	3.592	12.069
3	1.699	13.026	14	1.809	13.757	25	3.451	12.335
4	1.164	11.019	15	4.114	9.101	26	1.197	10.806
5	4.793	4.162	16	2.834	13.923	27	0.723	7.822
6	2.690	14.167	17	3.872	10.795	28	1.958	14.010
7	3.982	10.190	18	2.174	14.307	29	2.366	14.088
8	3.333	13.525	19	4.353	8.059	30	1.530	12.701
9	1.942	13.899	20	3.684	12.041	31	0.847	9.012
10	2.876	13.949	21	2.140	14.641	32	3.843	10.885
11	4.277	8.643	22	2.783	14.138	33	0.976	9.876

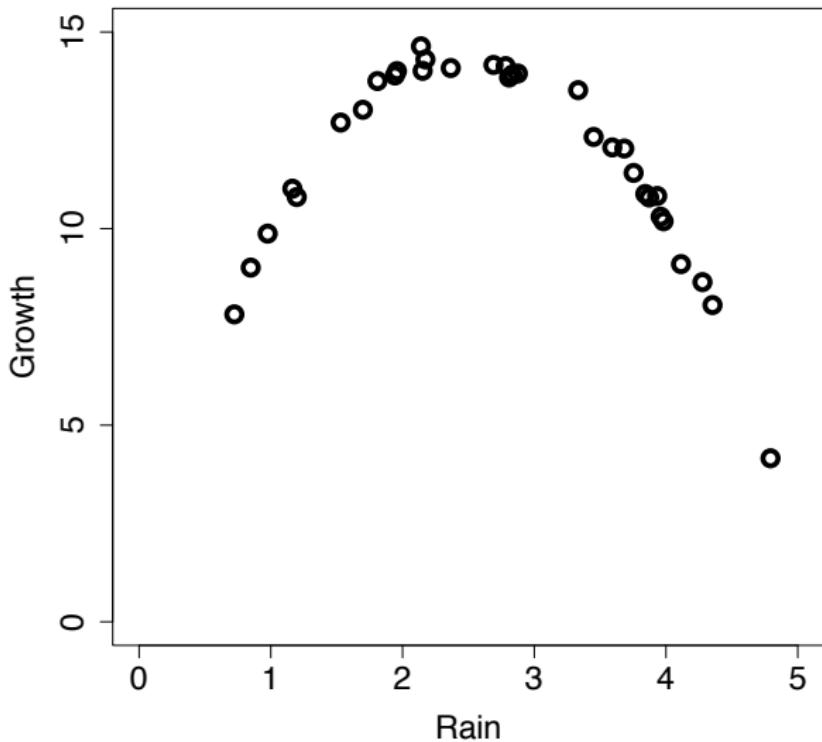


Figure: A scatter plot of the RAIN and GROWTH feature from the grass growth dataset.

- The best linear model we can learn for this data is:

$$\text{GROWTH} = 13.510 + -0.667 \times \text{RAIN}$$

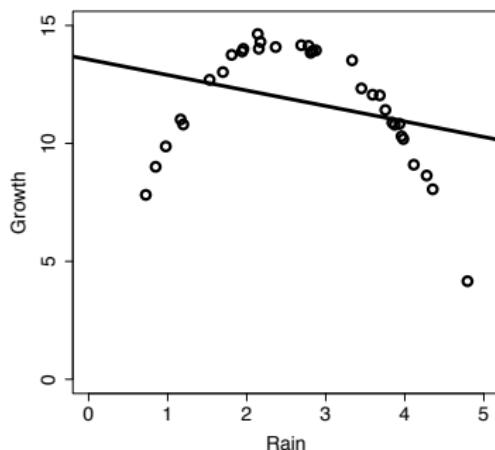


Figure: A simple linear regression model trained to capture the relationship between the grass growth and rainfall.

- In order to handle non-linear relationships we transform the data rather than the model using a set of basis functions:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \sum_{k=0}^b \mathbf{w}[k] \times \phi_k(\mathbf{d}) \quad (10)$$

where ϕ_0 to ϕ_b are a series of b basis functions that each transform the input vector \mathbf{d} in a different way.

- The advantage of this is that, except for introducing the mechanism of basis functions, we do not need to make any other changes to the approach we have presented so far.

- The relationship between rainfall and grass growth in the grass growth dataset can be accurately represented as a **second order polynomial** through the following model:

$$\text{GROWTH} = \mathbf{w}[0] \times \phi_0(\text{RAIN}) + \mathbf{w}[1] \times \phi_1(\text{RAIN}) + \mathbf{w}[2] \times \phi_2(\text{RAIN})$$

where

$$\phi_0(\text{RAIN}) = 1$$

$$\phi_1(\text{RAIN}) = \text{RAIN}$$

$$\phi_2(\text{RAIN}) = \text{RAIN}^2$$

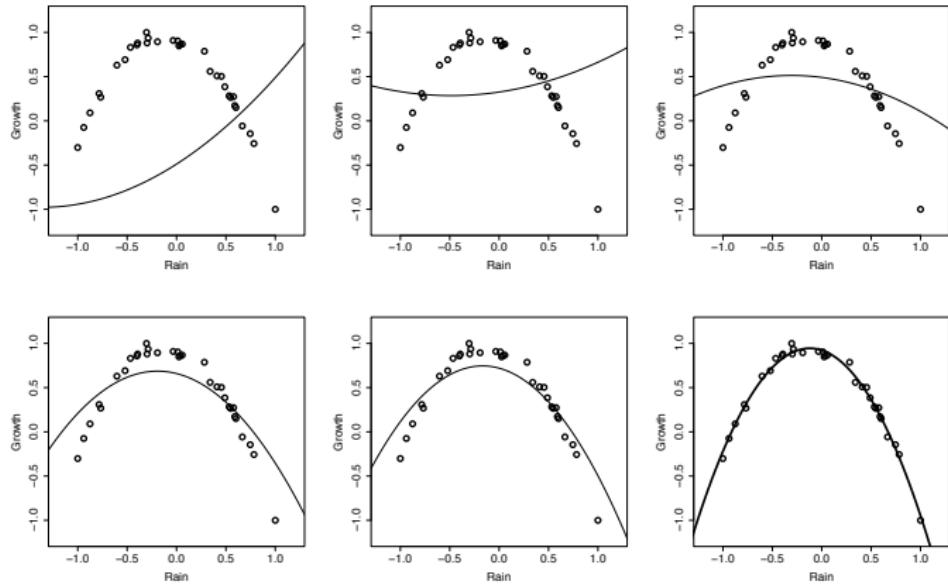
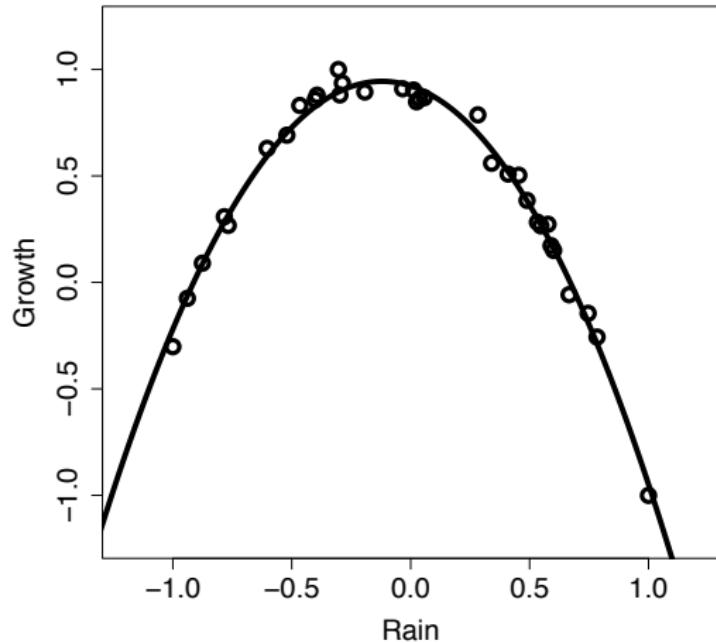


Figure: A selection of the models developed during the gradient descent process for the grass growth dataset from Table 5 [46]. (Note that the RAIN and GROWTH features have been **range normalized** to the $[-1, 1]$ range.)



$$\text{GROWTH} = 0.3707 \times \phi_0(\text{RAIN}) + 0.8475 \times \phi_1(\text{RAIN}) + -1.717 \times \phi_2(\text{RAIN})$$

$$\text{GROWTH} = 0.3707 \times \phi_0(\text{RAIN}) + 0.8475 \times \phi_1(\text{RAIN}) + -1.717 \times \phi_2(\text{RAIN})$$

$$\phi_0(\text{RAIN}) = 1$$

$$\phi_1(\text{RAIN}) = \text{RAIN}$$

$$\phi_2(\text{RAIN}) = \text{RAIN}^2$$

- What is the predicted growth for the following RAIN values:
 - ➊ RAIN= -0.75
 - ➋ RAIN= 0.1
 - ➌ RAIN= 0.9

- Basis functions can also be used for
 - ➊ multivariable simple linear regression models in the same way, the only extra requirement being the definition of more basis functions.
 - ➋ to train logistic regression models for categorical prediction problems that involve non-linear relationships.

Table: A dataset showing participants' responses to viewing '*positive*' and '*negative*' images measured on the EEG P20 and P45 potentials.

ID	P20	P45	TYPE	ID	P20	P45	TYPE
1	0.4497	0.4499	negative	26	0.0656	0.2244	positive
2	0.8964	0.9006	negative	27	0.6336	0.2312	positive
3	0.6952	0.3760	negative	28	0.4453	0.4052	positive
4	0.1769	0.7050	negative	29	0.9998	0.8493	positive
5	0.6904	0.4505	negative	30	0.9027	0.6080	positive
6	0.7794	0.9190	negative	31	0.3319	0.1473	positive
	⋮				⋮		

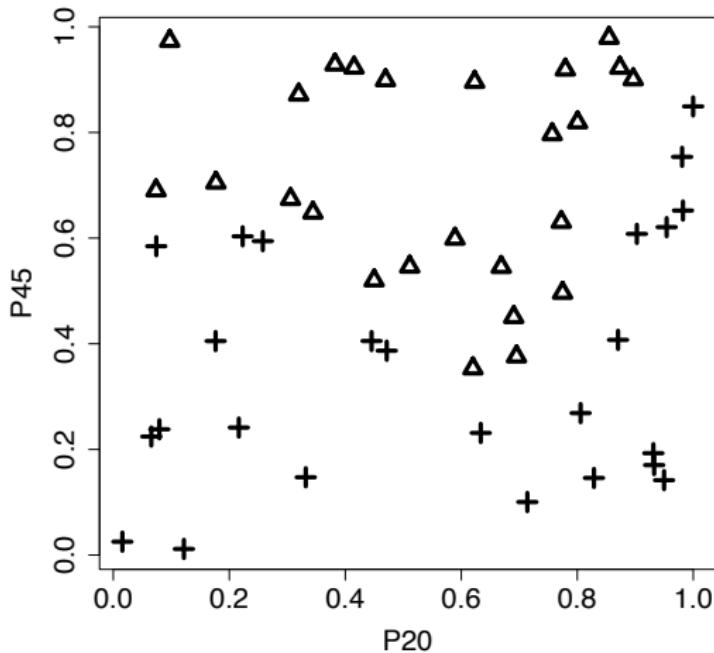


Figure: A scatter plot of the P20 and P45 features from the EEG dataset. 'positive' images are shown as crosses, and 'negative' images are shown as triangles.

- A logistic regression model using basis functions is defined as follows:

$$\mathbb{M}_{\mathbf{w}}(\mathbf{d}) = \frac{1}{1 + e^{-\left(\sum_{j=0}^b \mathbf{w}[j]\phi_j(\mathbf{d})\right)}} \quad (11)$$

- We will use the following basis functions for the EEG problem:

$$\phi_0(\langle P20, P45 \rangle) = 1 \quad \phi_4(\langle P20, P45 \rangle) = P45^2$$

$$\phi_1(\langle P20, P45 \rangle) = P20 \quad \phi_5(\langle P20, P45 \rangle) = P20^3$$

$$\phi_2(\langle P20, P45 \rangle) = P45 \quad \phi_6(\langle P20, P45 \rangle) = P45^3$$

$$\phi_3(\langle P20, P45 \rangle) = P20^2 \quad \phi_7(\langle P20, P45 \rangle) = P20 \times P45$$

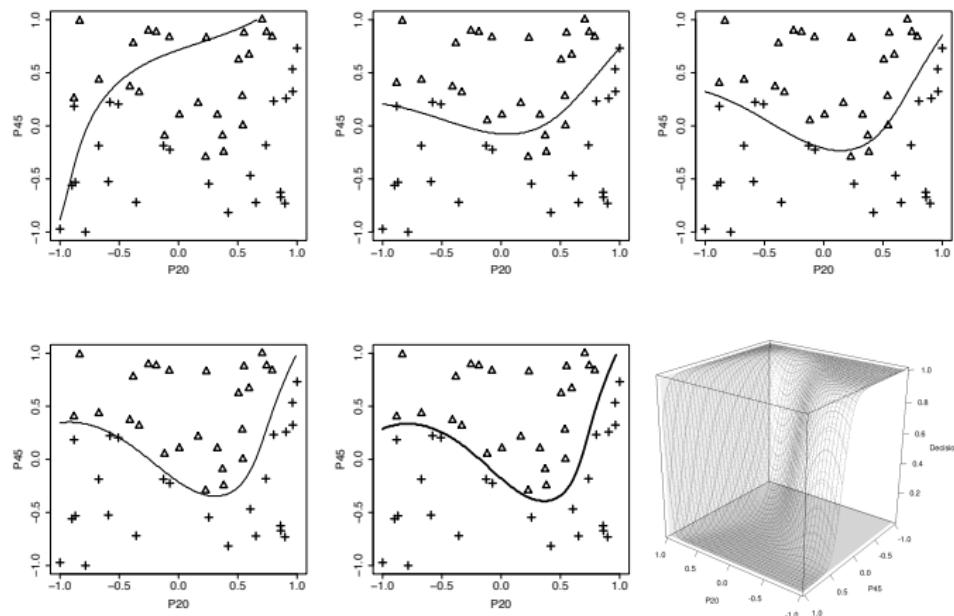
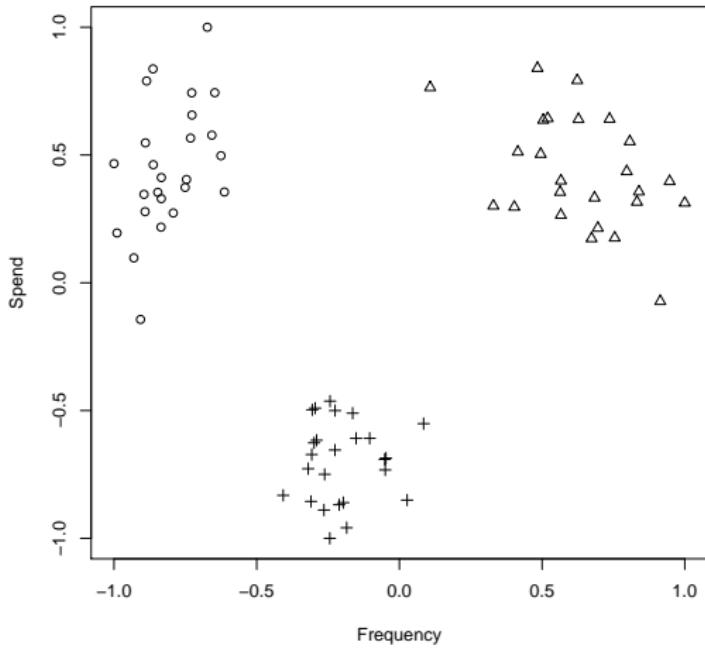
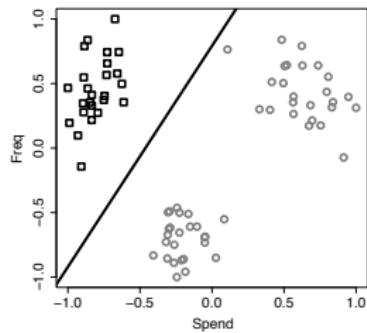


Figure: A selection of the models developed during the gradient descent process for the EEG dataset from Table 6^[55]. The final panel shows the decision surface generated.

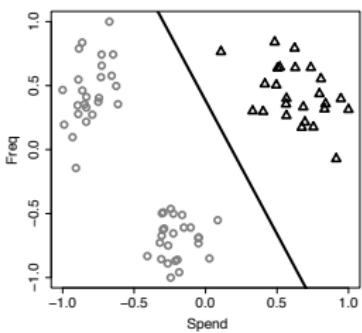
Multinomial Logistic Regression

Table: A dataset of customers of a large national retail chain.

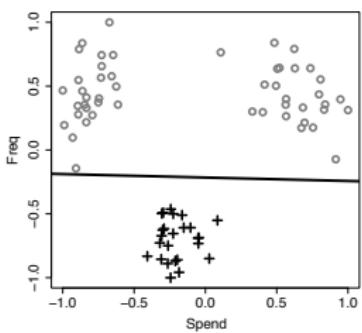




(a)



(b)



(c)

Figure: An illustration of three different **one-versus-all** prediction models for the customer type dataset in Table 7^[61] that has three target levels '*single*' (squares), '*business*' (triangles) and '*family*' (crosses).

- For r target feature levels, we build r separate logistic regression models M_{w_1} to M_{w_r} :

$$M_{w_1}(\mathbf{d}) = \text{logistic}(\mathbf{w}_1 \cdot \mathbf{d})$$

$$M_{w_2}(\mathbf{d}) = \text{logistic}(\mathbf{w}_2 \cdot \mathbf{d})$$

(12)

⋮

$$M_{w_r}(\mathbf{d}) = \text{logistic}(\mathbf{w}_r \cdot \mathbf{d})$$

where M_{w_1} to M_{w_r} are r different one-versus-all logistic regression models, and \mathbf{w}_1 to \mathbf{w}_r are r different sets of weights.

- To combine the outputs of these different models, we normalize their results using:

$$\mathbb{M}'_{\mathbf{w}_k}(\mathbf{d}) = \frac{\mathbb{M}_{\mathbf{w}_k}(\mathbf{d})}{\sum_{l \in \text{levels}(t)} \mathbb{M}_{\mathbf{w}_l}(\mathbf{d})} \quad (13)$$

where $\mathbb{M}'_{\mathbf{w}_k}(\mathbf{d})$ is a revised, normalized prediction for the one-versus-all model for the target level k .

- The r one-versus-all logistic regression models used are trained in **parallel**, and the **revised model outputs**, $\mathbb{M}'_{\mathbf{w}_k}(\mathbf{d})$, are used when calculating the sum of squared errors for each model during the training process.
- This means that the sum of squared errors function is changed slightly to

$$L_2(\mathbb{M}_{\mathbf{w}_k}, \mathcal{D}) = \frac{1}{2} \sum_{i=1}^n (t_i - \mathbb{M}'_{\mathbf{w}_k}(\mathbf{d}_i [1]))^2 \quad (14)$$

- The revised predictions are also used when making predictions for query instances. The predicted level for a query, \mathbf{q} , is the level associated with the one-versus-all model that outputs the highest result after normalization.
- We can write this as

$$\mathbb{M}(\mathbf{q}) = \operatorname{argmax}_{l \in \text{levels}(t)} \mathbb{M}'_{\mathbf{w}_l}(\mathbf{q}) \quad (15)$$

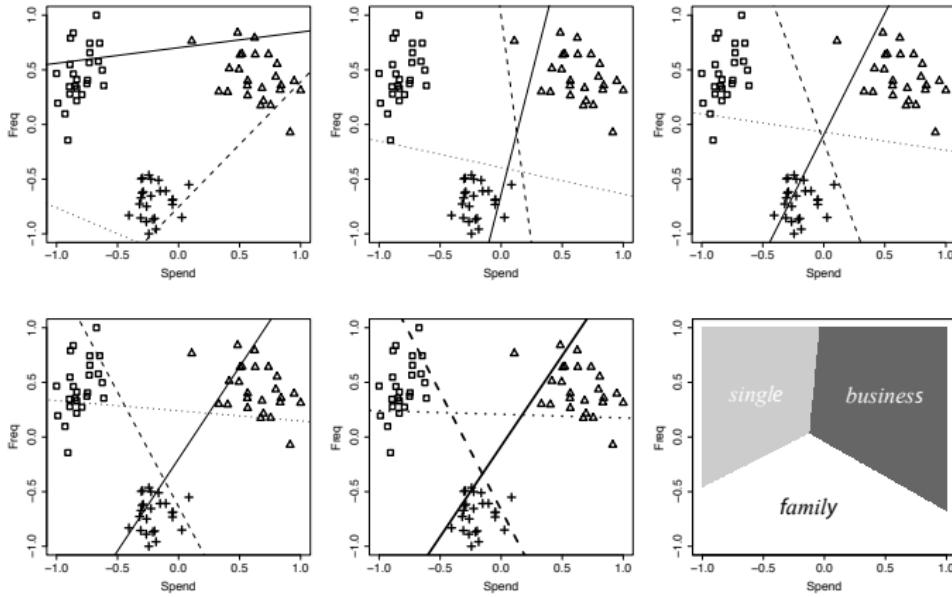
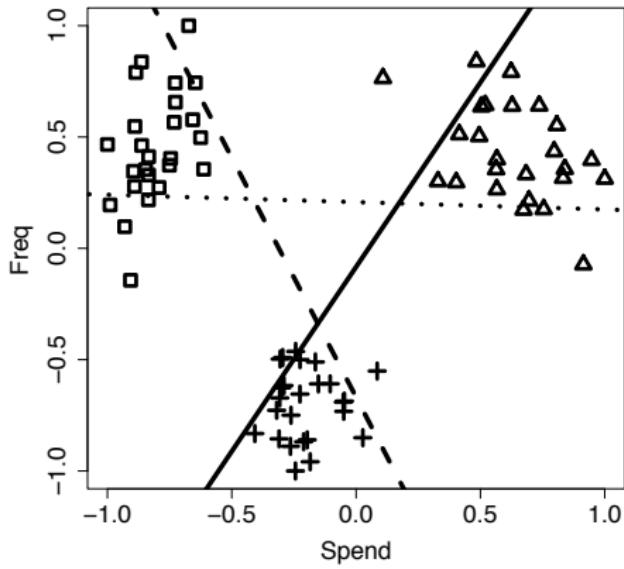


Figure: A selection of the models developed during the gradient descent process for the customer group dataset from Table 7 [61]. Squares represent instances with the '*single*' target level, triangles the '*business*' level and crosses the '*family*' level. (f) illustrates the overall decision boundaries that are learned between the three target levels.



$$\mathbb{M}_{\mathbf{w}'_{single}}(\mathbf{q}) = \text{Logistic}(0.7993 - 15.9030 \times \text{SPEND} + 9.5974 \times \text{FREQ})$$

$$\mathbb{M}_{\mathbf{w}'_{family}}(\mathbf{q}) = \text{Logistic}(3.6526 + -0.5809 \times \text{SPEND} - 17.5886 \times \text{FREQ})$$

$$\mathbb{M}_{\mathbf{w}'_{business}}(\mathbf{q}) = \text{Logistic}(4.6419 + 14.9401 \times \text{SPEND} - 6.9457 \times \text{FREQ})$$

- For a query instance with $\text{SPEND} = 25.67$ and $\text{FREQ} = 6.12$, which are normalized to $\text{SPEND} = -0.7279$ and $\text{FREQ} = 0.4789$, the predictions of the individual models would be

$$\begin{aligned}\mathbb{M}_{\mathbf{w}'_{single}}(\mathbf{q}) &= \text{Logistic}(0.7993 - 15.9030 \times (-0.7279) + 9.5974 \times 0.4789) \\ &= 0.9999\end{aligned}$$

$$\begin{aligned}\mathbb{M}_{\mathbf{w}'_{family}}(\mathbf{q}) &= \text{Logistic}(3.6526 + -0.5809 \times (-0.7279) - 17.5886 \times 0.4789) \\ &= 0.01278\end{aligned}$$

$$\mathbb{M}_{\mathbf{w}'_{business}}(\mathbf{q}) = ?$$

- For a query instance with $\text{SPEND} = 25.67$ and $\text{FREQ} = 6.12$, which are normalized to $\text{SPEND} = -0.7279$ and $\text{FREQ} = 0.4789$, the predictions of the individual models would be

$$\begin{aligned}M_{\mathbf{w}_{single}}(\mathbf{q}) &= \text{Logistic}(0.7993 - 15.9030 \times (-0.7279) + 9.5974 \times 0.4789) \\&= 0.9999\end{aligned}$$

$$\begin{aligned}M_{\mathbf{w}_{family}}(\mathbf{q}) &= \text{Logistic}(3.6526 + -0.5809 \times (-0.7279) - 17.5886 \times 0.4789) \\&= 0.01278\end{aligned}$$

$$\begin{aligned}M_{\mathbf{w}_{business}}(\mathbf{q}) &= \text{Logistic}(4.6419 + 14.9401 \times (-0.7279) - 6.9457 \times 0.4789) \\&= 0.0518\end{aligned}$$

- These predictions would be normalized as follows:

$$\mathbb{M}'_{\mathbf{w}'_{single}}(\mathbf{q}) = \frac{0.9999}{0.9999 + 0.01278 + 0.0518} = 0.9393$$

$$\mathbb{M}'_{\mathbf{w}'_{family}}(\mathbf{q}) = \frac{0.01278}{0.9999 + 0.01278 + 0.0518} = 0.0120$$

$$\mathbb{M}'_{\mathbf{w}'_{business}}(\mathbf{q}) = \frac{0.0518}{0.9999 + 0.01278 + 0.0518} = 0.0487$$

- This means the overall prediction for the query instance is '*single*', as this gets the highest normalized score.

Support Vector Machines

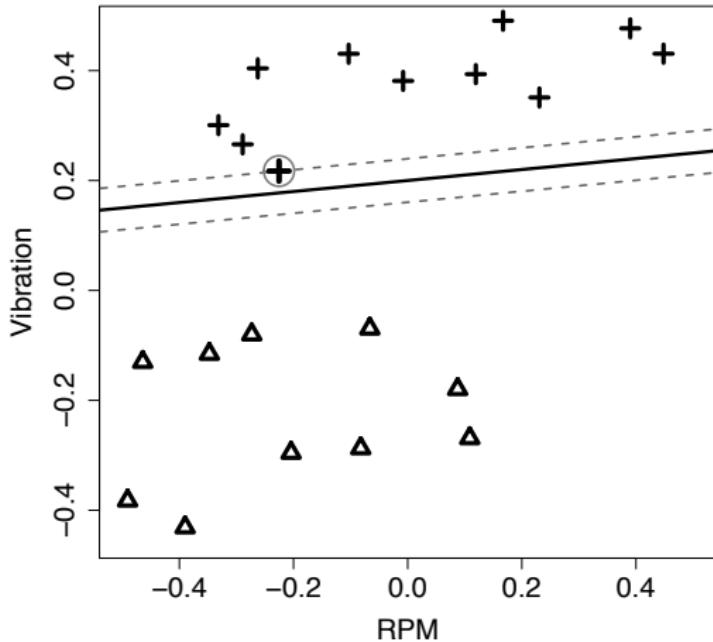


Figure: A small sample of the generators dataset with two features, RPM and VIBRATION, and two target levels, '*good*' (shown as crosses) and '*bad*' (shown as triangles). A decision boundary with a very small margin.

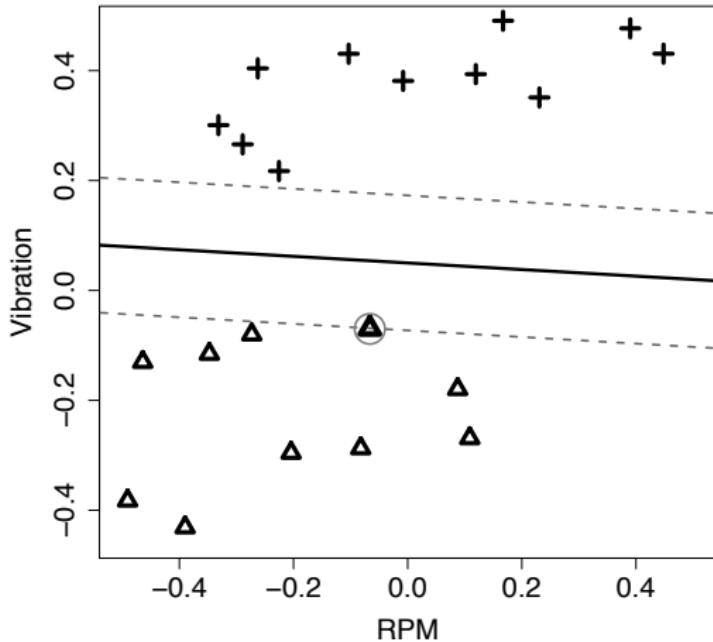


Figure: A small sample of the generators dataset with two features, RPM and VIBRATION, and two target levels, '*good*' (shown as crosses) and '*bad*' (shown as triangles). A decision boundary with a large margin.

- Training a support vector machine involves searching for the decision boundary, or **separating hyperplane**, that leads to the maximum margin as this will best separate the levels of the target feature.
- The instances in a training dataset that fall along the margin extents, and so define the margins, are known as the **support vectors** and define the decision boundary.

- We define the separating hyperplane as follows:

$$w_0 + \mathbf{w} \cdot \mathbf{d} = 0 \quad (16)$$

- For instances above a separating hyperplane

$$w_0 + \mathbf{w} \cdot \mathbf{d} > 0$$

and for instances below a separating hyperplane

$$w_0 + \mathbf{w} \cdot \mathbf{d} < 0$$

- We build a support vector machine prediction model so that instances with the negative target level result in the model outputting ≤ -1 and instances with the positive target level result in the model outputting $\geq +1$.
- The space between the outputs of -1 and $+1$ allows for the margin.

- A support vector machine model is defined as

$$\mathbb{M}_{\alpha, w_0}(\mathbf{q}) = \sum_{i=1}^s (t_i \times \alpha[i] \times (\mathbf{d}_i \cdot \mathbf{q}) + w_0) \quad (17)$$

where

- \mathbf{q} is the set of descriptive features for a query instance;
- $(\mathbf{d}_1, t_1), \dots, (\mathbf{d}_s, t_s)$ are s support vectors (instances composed of descriptive features and a target feature);
- w_0 is the first weight of the decision boundary;
- and α is a set of parameters determined during the training process (there is a parameter for each support vector $\alpha[1], \dots, \alpha[s]$).¹

¹These parameters are formally known as **Lagrange multipliers**.

- Training a support vector machine frames as a **constrained quadratic optimization problem**
- This type of problem is defined in terms of:
 - ➊ a set of constraints
 - ➋ an optimization criterion.

- The required **constraints** required by the training process are

$$w_0 + \mathbf{w} \cdot \mathbf{d} \leq -1 \text{ for } t_i = -1 \quad (18)$$

and:

$$w_0 + \mathbf{w} \cdot \mathbf{d} \geq +1 \text{ for } t_i = +1 \quad (19)$$

- We can combine these two constraints into a single constraint (remember t_i is always equal to either -1 or $+1$):

$$t_i \times (w_0 + \mathbf{w} \cdot \mathbf{d}) \geq 1 \quad (20)$$

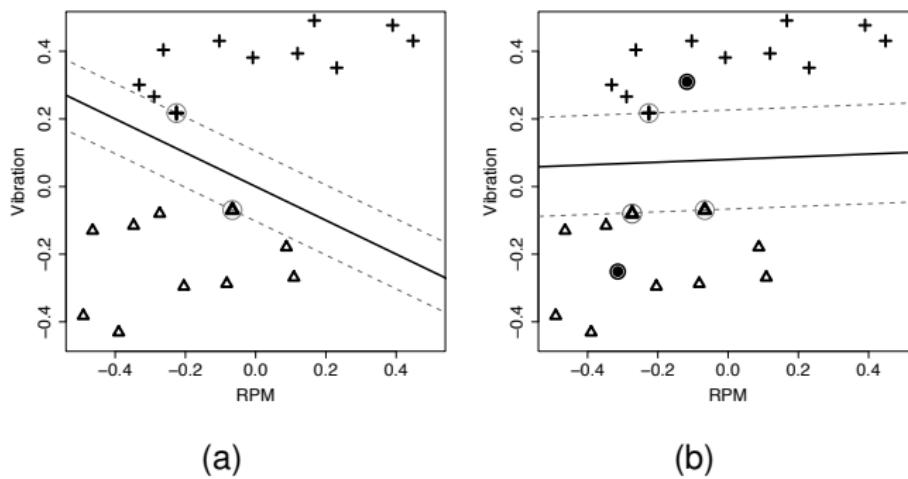


Figure: Different margins that satisfy the constraint in Equation (20)^[81]. The instances that define the margin are highlighted in each case. (b) shows the maximum margin and also shows two query instances represented as black dots.

- The **optimization** criterion used is defined in terms of the perpendicular distance from any instance to the decision boundary and is given by

$$dist(\mathbf{d}) = \frac{w_0 + \text{abs}(\mathbf{w} \cdot \mathbf{d})}{\|\mathbf{w}\|}$$

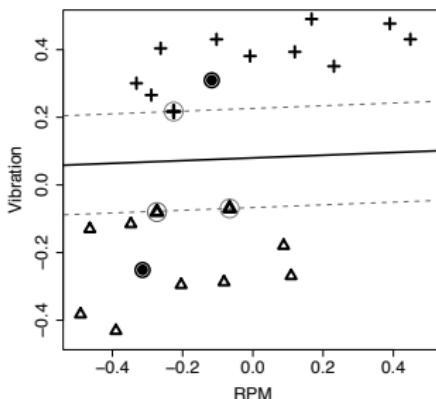
where $\|\mathbf{w}\|$ is known as the **Euclidean norm** of \mathbf{w} and is calculated as

$$\|\mathbf{w}\| = \sqrt{\mathbf{w}[1]^2 + \mathbf{w}[2]^2 + \dots + \mathbf{w}[m]^2}$$

- For instances along the **margin extents**,
 $\text{abs}(\mathbf{w} \cdot \mathbf{d} + w_0) = 1$.
- So, the distance from any instance along the margin extents to the decision boundary is $\frac{1}{\|\mathbf{w}\|}$, and because the margin is symmetrical to either side of the decision boundary, the size of the margin is $\frac{2}{\|\mathbf{w}\|}$.

- The goal when training a support vector machine is
 - maximize $\frac{2}{\|\mathbf{w}\|}$
 - subject to the constraint

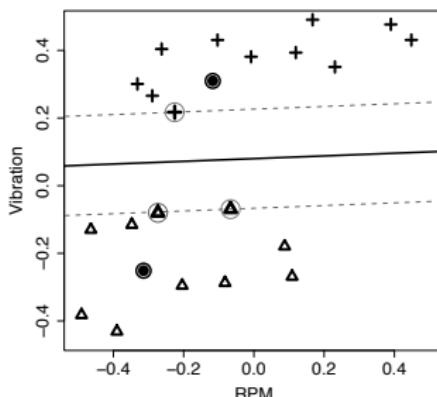
$$t_i \times (w_0 + \mathbf{w} \cdot \mathbf{d}) \geq 1$$



- The optimal decision boundary and associated support vectors for the example we have been following
- In this case '*good*' is the positive level and set to +1, and '*faulty*' is the negative level and set to -1.

- The descriptive feature values and target feature values for the support vectors in these cases are
 - $(\langle -0.225, 0.217 \rangle, +1)$,
 - $(\langle -0.066, -0.069 \rangle, -1)$,
 - $(\langle -0.273, -0.080 \rangle, -1)$.
- The value of w_0 is -0.1838 ,
- The values of the α parameters are

$$\langle 22.056, 6.998, 16.058 \rangle.$$



- The plot shows the position of two new query instances for this problem.
- The descriptive feature values for these queries are
 - $q_1 = \langle -0.314, -0.251 \rangle$
 - $q_2 = \langle -0.117, 0.31 \rangle$.

- For the first query instance, $\mathbf{q}_1 = \langle -0.314, -0.251 \rangle$, the output of the support vector machine model is:

$$\mathbb{M}_{\alpha, w_0}(\mathbf{q}_1)$$

$$\begin{aligned}&= (1 \times 23.056 \times ((-0.225 \times -0.314) + (0.217 \times -0.251))) - 0.1838 \\&\quad + (-1 \times 6.998 \times ((-0.066 \times -0.314) + (-0.069 \times -0.251))) - 0.1838 \\&\quad + (-1 \times 16.058 \times ((-0.273 \times -0.314) + (-0.080 \times -0.251))) - 0.1838 \\&= -2.145\end{aligned}$$

- The model output is less than -1 , so this query is predicted to be a '*faulty*' generator.
- For the second query instance, the model output is 1.592 , so this instance is predicted to be a '*good*' generator.

- **Basis functions** can be used with support vector machines to handle data that is not **linearly separable**
- To use basis functions we update Equation (20)^[81] to

$$t_i \times (w_0 + \mathbf{w} \cdot \phi(\mathbf{d})) \geq 1 \text{ for all } i \quad (21)$$

where ϕ is a set of basis functions applied to the descriptive features \mathbf{d} , and \mathbf{w} is a set of weights containing one weight for each member of ϕ .

- Typically, the number of basis functions in ϕ is larger than the number of descriptive features, so the application of the basis functions moves the data into a higher-dimensional space.
- The expectation is that a linear separating hyperplane will exist in this higher-dimensional space even though it does not in the original feature space.

- The prediction model in this case becomes

$$\mathbb{M}_{\alpha, \phi, w_0}(\mathbf{q}) = \sum_{i=1}^s (t_i \times \alpha[i] \times (\phi(\mathbf{d}_i) \cdot \phi(\mathbf{q})) + w_0) \quad (22)$$

- This equation requires a dot product calculation between the result of applying the basis functions to the query instance and to each of the support vectors which is repeated multiple times during the training process.

- A dot product is a computationally expensive operation,
- We can use a clever trick is used to avoid it:
 - the same result obtained by calculating the dot product of the descriptive features of a support vector and a query instance after having applied the basis functions can be obtained by applying a much less costly **kernel function**, *kernel*, to the original descriptive feature values of the support vector and the query.

- The prediction equation becomes

$$\mathbb{M}_{\alpha, \text{kernel}, w_0}(\mathbf{q}) = \sum_{i=1}^s (t_i \times \alpha[i] \times \text{kernel}(\mathbf{d}_i, \mathbf{q}) + w_0) \quad (23)$$

- A wide range of standard kernel functions can be used with support vector machines including:

Linear kernel $\text{kernel}(\mathbf{d}, \mathbf{q}) = \mathbf{d} \cdot \mathbf{q} + c$

where c is an optional constant

Polynomial kernel $\text{kernel}(\mathbf{d}, \mathbf{q}) = (\mathbf{d} \cdot \mathbf{q} + 1)^p$

where p is the degree of a polynomial function

Gaussian radial basis kernel

$\text{kernel}(\mathbf{d}, \mathbf{q}) = \exp(-\gamma \|\mathbf{d} - \mathbf{q}\|^2)$

where γ is a manually chosen tuning parameter

- 1 Interpreting Multivariable Linear Regression Models
- 2 Setting the Learning Rate Using Weight Decay
- 3 Handling Categorical Descriptive Features
- 4 Handling Categorical Target Features: Logistic Regression
- 5 Modeling Non-linear Relationships
- 6 Multinomial Logistic Regression
- 7 Support Vector Machines