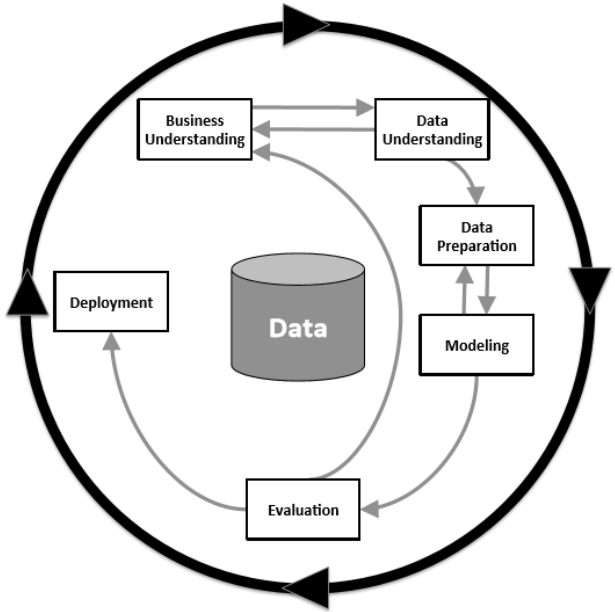
Advanced Data & Network Mining

Modelling - Supervised Learning Classification

2023-24

terri.hoare@dbs.ie

Recap on Challenges and Methodology
CRISP-DM (Cross Industry Standard Process for Data Mining)



The CRISP-DM process, including the six key phases and the important relationships between them (adapted from Wirth and Hipp, 2000, repr. in Kelleher *et al.*, 2020, p.14)

Data MiningTaxonomy
Matching Problems to Data Mining Algorithms

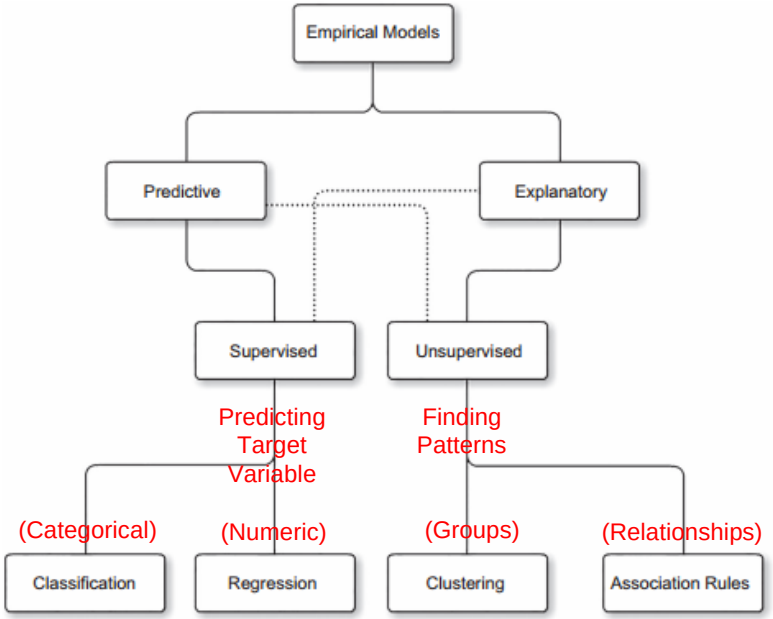
Classification
Predicting a Categorical Target Variable **(supervised)**

Regression
Predicting a Numeric Target Variable **(supervised)**

Association
Unsupervised process for finding relationships between Items

Clustering
Unsupervised process for finding meaningful groups in Data

Data Mining
Taxonomy of Algorithms



K-NN
Decision Trees
Rule Induction
Naïve Bayes
ANN + DeepLearning
SVM + FLM
Ensemble – RF + GBT

Linear Reg.
GLM
Logistic Reg.

K-means
X-Means
K-Medoids
DBSCAN
SOMs
Anomaly-Distance
Anomaly-Density
Anomaly-Local Outlier factor

Apriori
FP Growth
Collaborative Filtering
Content Based Filtering
Graph

Classification
Selected Algorithms (7)

- **k-Nearest Neighbors**

A lazy learner where no model is generalized. Any new unknown data is compared against similar known data point in the training set

- **Decision Trees**

Partitions the data into smaller subsets where each subset contains (mostly) responses of one class (either “yes” or “no”)

- **Rule Induction**

Models the relationship between input and output by deducing simple IF/THEN rules from a data set

- **Naïve Bayesian**

Predicts the output class based on Bayes theorem by calculating class conditional probability and prior probability

Continued…

Classification
Selected Algorithms (7) – Continued

- **Artificial Neural Networks**

A computational and mathematical model inspired by the biological nervous system. The weights in the network learn to reduce the error between actual and prediction

- **Support Vector Machines**

Essentially a boundary detection algorithm that identifies / defines multi-dimensional boundaries separating data points belonging to different classes

- **Ensemble Models**

Leverages wisdom of the crowd. Employs a number of independent models to make a prediction and aggregates the final production
(e.g. AdaBoost, **Random Forest**)

Classification Models
k-Nearest Neighbors

- “Lazy Learner” – memorises the entire training set and works by “look-up”.
- Non-parametric method. No building of a relationship function that must “generalise” well on unseen data
- “Birds of a feather flock together”. Similar records congregate in a neighbourhood in n-dimensional space with the same Target Class labels

Classification Models
k-Nearest Neighbors : Case Study Example

PREDICTING THE TYPE OF FOREST

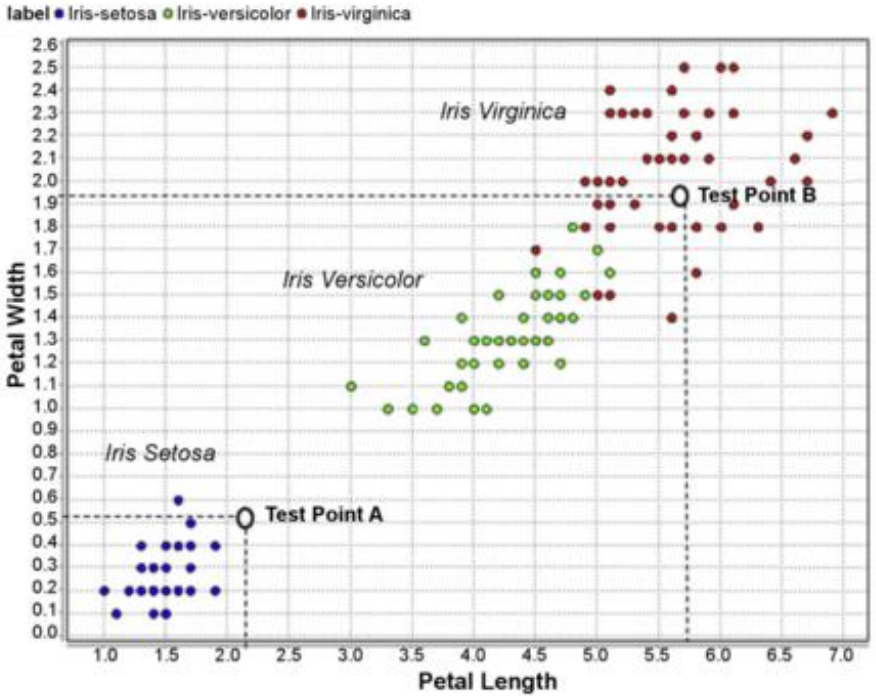
Satellite imaging and digital image processing have provided a wealth of data about almost every part of the earth's landscape. There is strong motivation for forestry departments, government agencies, universities, and research bodies to understand the makeup of forests, species of trees and their health, biodiversity, density, and forest condition. Field studies for developing forest databases and classification project are quite a tedious task and expensive. However, we can aid this process by leveraging satellite imagery, limited field data, elevation models, aerial photographs, and survey data [McInerney, 2005]. The objective is to classify whether the landscape is a forest or not and further predict the type of trees and species.

The approach to classifying the landscape involves dividing the area into land units (e.g., a pixel in a satellite image) and creating a vector of all the measurements for the land unit, by combining all the available data sets. Each unit's measurements are then compared against the measurements of known pre-classified units. For a given pixel, we can find a pixel in the pre-classified catalog, which has measurements very close to the measurement of the pixel to be predicted. Say the pre-classified pixel with the closest measurement corresponds to birch trees. Thus, we can predict the pixel area to be a birch forest. Every pixel's measurement is compared to measurements of the pre-classified data set to determine the like pixels and hence same forest types. This is the core concept of the k-nearest neighbor algorithm that is used to classify the landscape areas. [Haapanen et al., 2001]

(Kotu and Deshpande, 2019, p.99-100)

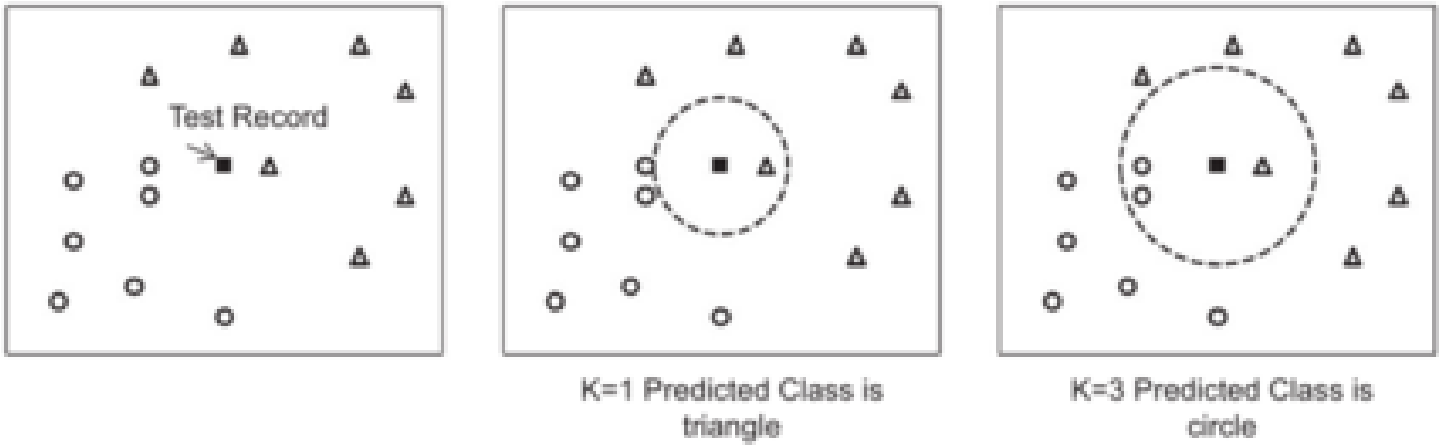
http://sustain.stanford.edu/predicting-poverty

Classification Models
k-Nearest Neighbors - Continued



Two-dimensional plot of Iris dataset: petal length and petal width. Classes are stratified with colours adapted from (Kotu and Deshpande, 2019, p.101)

Classification Models
k-Nearest Neighbors : Handling Outliers - Defining k



(Left) Dataset with a record of unknown class. (Middle) Decision boundary with k = 1 around unknown class record. (Right) Decision boundary with k = 3 around unknown test record.adapted from (Kotu and Deshpande, 2019, p.102)

Classification Models
k-Nearest Neighbors : Proximity – Distance Weights

- Weights are implemented so that the closest neighbours have more say in the outcome of the predicted target class than the further neighbours. Weights are assigned to all the neighbours with the weights increasing as the neighbours get closer to the data point.

Classification Models
k-Nearest Neighbors : Normalising Inputs

Pair (Credit Score, Income)

Which pair is more similar?
Pair A (500, $40 000) and (600, $40 000)
Pair B (500, $40 000) and (500, $39 800)

Pair A, substantial difference in credit score is substantial
Pair B has only a 0.5% difference in Income
However as absolute difference in Pair B greater, need to normalise

Normalisation methods
Eg. Range, Z-transformation, proportion, interquartile range

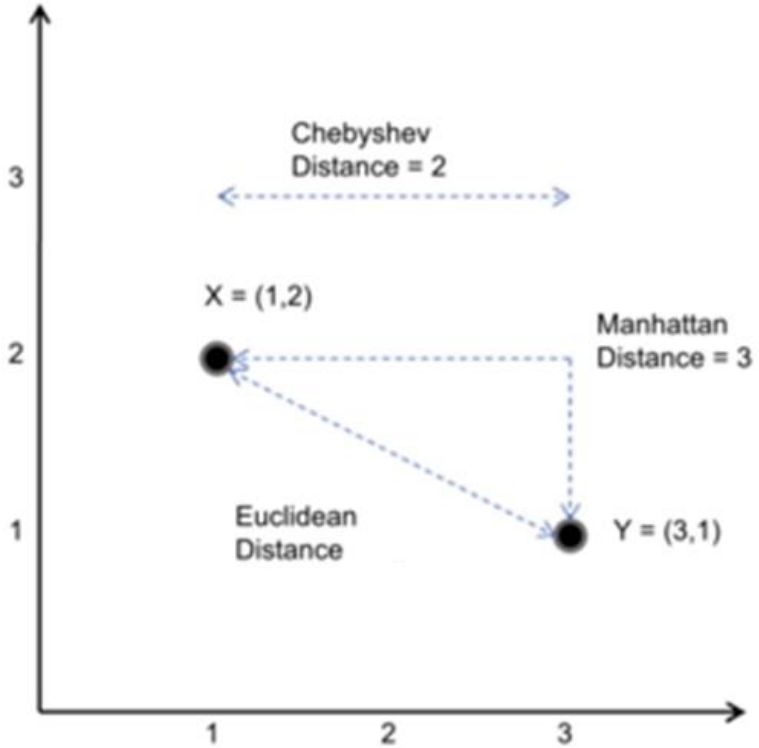
Z-transformation (most commonly used)
Rescaling by subtracting the mean from each value and dividing by the standard deviation, resulting in a transformed set of values with mean of 0 and standard deviation of 1

E.g. Iris data set sepal length (cm)
[4.3 , 7.9] std deviation 0.828 transformed
[-1.86 , 2.84] std deviation 1

Classification Models
k-Nearest Neighbors : Measures of Proximity

- **Distance**
- **Correlation Similarity**
- **Simple Matching Coefficient**
- **Jaccard Similarity**
- **Cosine Similarity**

Classification Models
k-Nearest Neighbors : Measures of Proximity - Distance



Distance measures adapted from (Kotu and Deshpande, 2019, p.104)

$$d = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

Minkowski or **p-norm** Distance

p=1 (Manhatten) “taxi cab”
(2-1)+(3-1) = 3

p=2 (Euclidean) most common
SQRT ($(2-1)^2 + (3-1)^2$) = 2.27

p=∞ (Chebyshev)
Max [(3-1), (1-2)] = 2

Classification Models
k-Nearest Neighbors : Measures of Proximity – Cont.

Correlation Similarity

Correlation similarity between two data points X and Y is a measure of the linear relationship between attributes X and Y

Pearson Correlation [-1, 1] perfect negative to perfect positive with 0 being no relationship.

Eg. Pearson Correlation for data points X (1,2,3,4,5) and Y (10,15,35,40,55) is 0.98

Simple Matching Coefficient

Used for Binary data. Based on simultaneous occurrence of 0 or 1 with respect to total occurrences. $(\text{SMC}) = \frac{\text{matching occurences}}{\text{total occurences}}$

Eg. X (1,1,0,0,1,1,0) and Y(1,0,0,1,1,0,0) = 4/7 = 0.57

Classification Models
k-Nearest Neighbors : Measures of Proximity – cont.

Jaccard Similarity

Used in applications such as text mining where a document vector contains the attributes (words) and non-occurrence is of no interest and ignored.

$$\text{Jaccard coefficient} = \frac{\textit{common occurences}}{\textit{total occurences}}$$

Eg. X (1,1,0,0,1,1,0) and Y (1,0,0,1,1,0,0) = 2/5 = 0.4

Cosine Similarity

Used in text mining where the document vector contains the number of occurrences of a word

Eg. X (1,2,0,0,3,4,0) and Y (5,0,0,6,7,0,0)

$$x \cdot y = \sqrt{1*5+2*0+0*0+0*6+3*7+4*0+0*0} = 5.1$$
$$\|x\| = \sqrt{1*1+2*2+0*0+0*0+3*3+4*4+0*0} = 5.5$$
$$\|y\| = \sqrt{5*5+0*0+0*0+6*6+7*7+0*0+0*0} = 10.5$$
$$\text{Cosine similarity } (|x \cdot y|) = \frac{x \cdot y}{\|x\| \|y\|} = \frac{5.1}{5.5 * 10.5} = 0.08$$

Classification Models
k-Nearest Neighbors : How to Implement

Step 1 – Data Preparation
Normalize all attributes (z-transformation most common)
Split dataset into training and test sets (equally using shuffled sampling)

Step 2 – Modeling and Parameters
Model k-NN with parameters k, weighted vote, measure type, measure

Step 3 – Implement Evaluation Method
Apply Model learned on Training Data set trained Model to the Test Data
Evaluate Performance comparing the predicted class with the labelled class for all of the Test records

Step 4 – Execution and Interpretation
Review the Output Results including the **k-NN Model** and **Performance vector** (confusion matrix with correct and incorrect predictions for all of the dataset) . Accurate prediction of 72 of 75 records.

Classification Models
k-Nearest Neighbors : Discussion Points

- Needs normalised data points
- Robust to missing values. E.G. In Iris dataset if missing sepal length in a test record, k-NN becomes three instead of four-dimensional
- Lazy learner, memorises data set, no generalizable abstraction built
- Build is quick but deployment slow as must find distance between test record and all training records. Not suitable for time sensitive applications such as online advertisement or fraud detection
- Can be applied to categorical attributes 0 - different, 1 – same. Ordinal values can be converted to numeric values to better leverage the distance function

Classification Models
k-Nearest Neighbors : Summary

Model
Entire training dataset is the model

Input
No restrictions. However, distance calculations work better with numeric data. Data needs to be normalised

Output
Prediction of target variable which is categorical

Pros
Requires little time to build model. Handles missing values in the unknown record gracefully. Works with non-linear relationships

Cons
Deployment runtime and storage requirements expensive. Arbitrary selection of the value of k. No description of model

Use Cases
Image processing applications where slower response time is acceptable

Classification Models
Decision Trees

One of the most intuitive and frequently used data mining algorithms

CART (Classification and Regression Trees)
Classification Trees used to separate a dataset into classes belonging to the response variable usually Yes/No or 1/0
Regression Trees used when the response variable is numeric or continuous

CART predictors may be either categorical or numeric

Decision Trees are built up recursively by increasing the information (reducing the uncertainty) contained in the reduced data set following each split

Classification Models
Decision Trees : Reducing Uncertainty

A box can contain one of three coloured balls (red, yellow, blue). Without opening the box, how many questions would remove the uncertainty?

Two binary yes/no questions. Maximum number questions needed to reduce uncertainty is $\log_2 T$ where T is the number of possible outcomes. If only one color then $\log_2 1 = 0$ that is no uncertainty.

Entropy is a measure of uncertainty. In the example of the box, if there are T events with equal probability of occurrence P, then $T = 1/P$ and Entropy = $\log_2 (1/P) = -\log_2 P$.

If the probability is not identical, a weighted formula applies: -

$$H = -\sum p_x \log_2(p_x)$$

Classification Models
Decision Trees : Reducing Uncertainty

Many real-world business problems can be thought of as extensions to this “uncertainty reduction” example.

For example, knowing only a handful of characteristics such as the length of a loan, borrower’s occupation, annual income, and previous credit behaviour, we can use several of the available predictive analytics techniques to rank the riskiness of a potential loan, and by extension, the interest rate of the loan.

This is nothing but a more sophisticated uncertainty reduction exercise, similar in spirit to the ball-in-a-box problem.

Decision trees embody this problem-solving technique by systematically examining the available attributes and their impact on the eventual class or category of a sample.

Classification Models
Entropy and a worked example using Golf Dataset

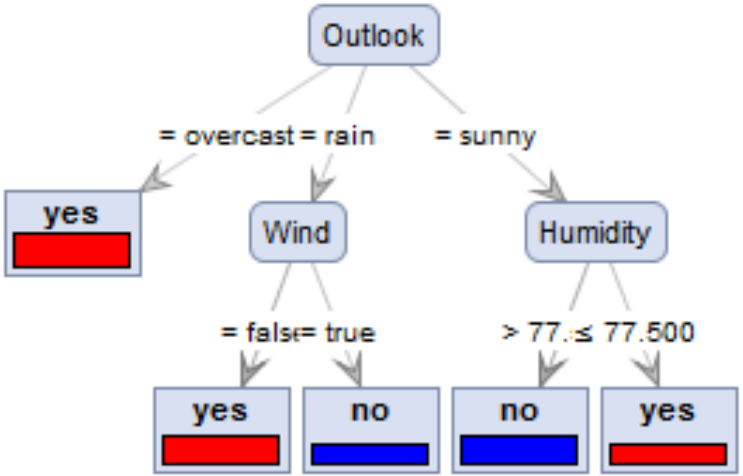
Row No.	Play	Outlook	Temperature	Humidity	Wind
1	no	sunny	85	85	false
2	no	sunny	80	90	true
3	yes	overcast	83	78	false
4	yes	rain	70	96	false
5	yes	rain	68	80	false
6	no	rain	65	70	true
7	yes	overcast	64	65	true
8	no	sunny	72	95	false
9	yes	sunny	69	70	false
10	yes	rain	75	80	false
11	yes	sunny	75	70	true
12	yes	overcast	72	90	true
13	yes	overcast	81	75	false
14	no	rain	71	80	true

Information Gain All Attributes

Temperature	0.029
Humidity	0.102
Wind	0.048
Outlook	0.247

H(Outlook:overcast) = - (0/4)$\log_2$(0/4) – (4/4)$\log_2$(4/4) = 0.0
Similarly H(Outlook:sunny) = 0.971 and H(Outlook:rain) = 0.971
I(Outlook) = (4/14) * 0 + (5/14) * 0.971 + (5/14) * 0.971 = 0.693
I(Outlook, no partition) = - (5/14)$\log_2$(5/14) – (9/14)$\log_2$(9/14) = 0.940
Information gain Outlook = I(Outlook, no partition) - I(Outlook) = 0.940 – 0.693 = 0.247

Classification Models
Decision Trees : Rapid Miner Golf Dataset – Cont.



Classification Models
Decision Tree : Split Criteria?

Information Gain: computed as the information before the split minus the information after the split. Works well, unless there are a few variables that have a large number of values (or classes). Information gain is *biased* towards choosing attributes with a large number of values as root nodes.

For example, each customer ID is unique and thus the variable has too many values and the tree has no predictive value.

Classification Models
Decision Tree : Split Criteria?

Gain Ratio (default): A modification of Information Gain. Reduces the bias and is usually the best option. Takes into account the number of branches that would result before making the split. Corrects Information Gain by taking the intrinsic information of a split into account.

In the Golf example:- suppose each of the 14 examples had a unique ID attribute associated with it.
Intrinsic Information ID attribute is 14 * (–1/14) * log (1/14)) = 3.807.
Gain Ratio is obtained by dividing the Information Gain for an attribute by its intrinsic information. Attributes that have very high intrinsic information (high uncertainty) tend to offer low gains upon splitting and hence will not be automatically selected. Note log base 2.

Gini Index: Also used but does not have too many advantages over gain ratio.

Classification Models
Decision Tree : Split Criteria?

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.362
Gain ratio: 0.247/1.577	0.156	Gain ratio: 0.029/1.362	0.021

Humidity		Windy	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

Intrinsic Information Outlook (R Code)
- 5/14 * log (5/14,2) - 4/14 * log (4/14,2) - 5/14 * log (5/14,2)
= 1,577

Classification Models
Decision Tree : When to Stop Splitting The Data?

No attribute satisfies a minimum information gain threshold.

A maximal depth is reached: as the tree grows larger , not only does interpretation get harder but run into situation called “overfitting”.

There are less than a certain number of examples in the current subtree: another a mechanism to prevent overfitting.

Classification Models
Decision Tree : Managing Overfitting by Pruning

- Overfitting occurs when a model tries to memorise the data instead of generalising the relationship between input and output variables. Overfitting often has the effect of performing very well on the training data set but performing poorly on any new data previously unseen by the model.
- Overfitting can be prevented by **pruning**

Pre-pruning - stopping when no attribute satisfies a minimum information gain threshold or a maximal depth is reached or there are less than a certain number of examples in the current subtree

Post-pruning – allow the tree to grow as deep as the data will allow and then trim or prune those branches that do not effectively change the classification error rates

- Post-pruning can sometimes be better to avoid missing small but potentially significant relationships between attribute values and classes. However, a drawback is the wasted computations when the tree needs to be trimmed back

Classification Models
Decision Tree : Other Parameters

- **Minimal Gain Value**.

Can theoretically take any value from 0 upwards. In practice, a minimal gain of 0.2 to 0.3 is considered good. Default is 0.1.

- **Minimal size for a split, minimal leaf size, maximal depth.**
  Determined by the size of the data set. Can be tuned by using an Optimization Routine.

Classification Models
Decision Trees : Business Application - Credit Scoring

- Types of situations where Credit Scoring are applied include
  - **Prospect filtering** Identifying which prospects to extend credit to & determining how much credit would be an acceptable risk
  - **Default risk detection** Deciding if a particular customer is likely to default on a loan
  - **Bad debt collection** Sorting out those debtors who will yield a good cost (of collection) to benefit (of receiving payment) performance

- We will use the well-known German Credit dataset from the University of California-Irvine Machine Learning data repository to look at building a decision tree using Rapid Miner for addressing a **prospect filtering problem**

Classification Models
Decision Trees : Raw German Credit Data

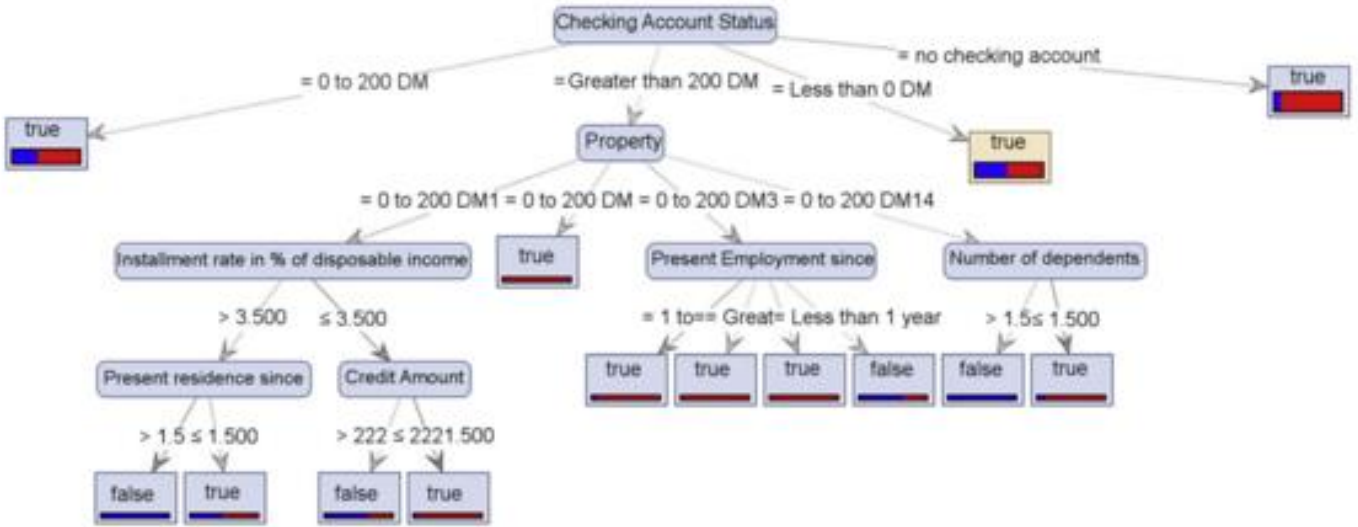
Checking Account Status	Duration in Month	Credit History	Purpose	Credit Amount	Savings Account/ Bonds	Present Employment Since	Credit Rating
A11	6	A34	A43	1169	A65	A75	1
A12	48	A32	A43	5951	A61	A73	2
A14	12	A34	A46	2096	A61	A74	1
A11	42	A32	A42	7882	A61	A74	1
A11	24	A33	A40	4870	A61	A73	2
A14	36	A32	A46	9055	A65	A73	1
A14	24	A32	A42	2835	A63	A75	1
A12	36	A32	A41	6948	A61	A73	1
A14	12	A32	A43	3059	A64	A74	1
A12	30	A34	A40	5234	A61	A71	2
A12	12	A32	A40	1295	A61	A72	2
A11	48	A32	A49	4308	A61	A72	2

1000 samples, 20 attributes with look-up encodings for values (eg. purpose of car A40 (new car) and A41 (used car), 1 label (target) binominal attribute (credit rating) which can take the value 1 (good) or 2 (bad). 70% of samples fall into the “good” credit rating class.

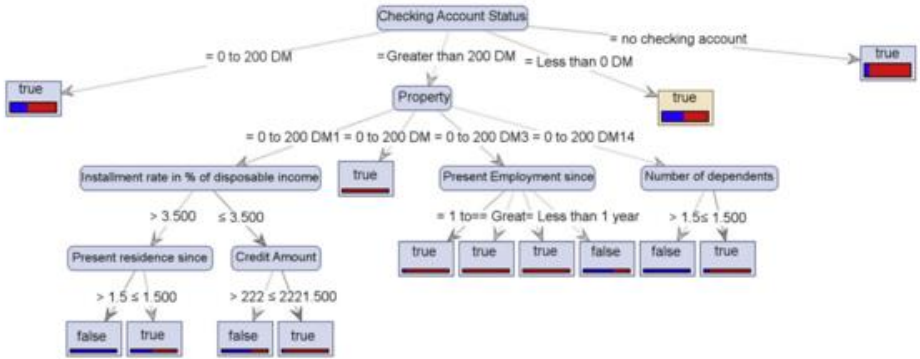
Classification Models
Decision Trees : German Credit Risk - RapidMiner - Cont.

www.LearnPredictiveAnalytics.com 04_Class_4.1_decision_tree_german_credit_risk.rmp

Credit Risk Rating = Good?



Classification Models
Decision Trees : German Credit Risk - RapidMiner - Cont.



- "Checking Account Status" attribute manages to classify nearly 94%
- "Checking Account Status" = "no checking account", chances true 88%
- Tree unable to pick out true/false cases if "Checking Account Status" is Less than (0 DM) or (0 to 200 DM)
- Numerous terminal leaves frequencies of occurrence as low as 2 implies tree suffers from overfitting

Classification Models
Decision Trees : German Credit Risk - RapidMiner - Cont.

www.LearnPredictiveAnalytics.com 04_Class_4.1_decision_tree_german_credit_risk.rmp

Optimizing Decision Tree Parameters to improve accuracy and class recall

criterion	gain	tree depth	accuracy	recall
gain_ratio	0.010	12	0.780	0.914
gini_index	0.010	5	0.760	0.871
gini_index	0.039	5	0.760	0.871
gini_index	0.068	5	0.760	0.871
gini_index	0.097	5	0.760	0.871
gini_index	0.126	5	0.760	0.871
gini_index	0.155	5	0.760	0.871
gini_index	0.184	5	0.760	0.871
gini_index	0.213	5	0.760	0.871
gini_index	0.242	5	0.760	0.871
gini_index	0.271	5	0.760	0.871
gini_index	0.300	5	0.760	0.871
gini_index	0.010	18	0.760	0.857
gini_index	0.039	18	0.760	0.857

Classification Models
Decision Trees : Discussion Points

- Easy to interpret and explain to non-technical users
- Relatively little effort in data preparation, normalisation not required, not sensitive to missing values and outliers, non-linear relationships between attributes do not affect tree performance
- Feature selection implicitly performed using Information Gain
- Key disadvantage is that without proper pruning or limiting tree growth tend to over-fit the data
- Selecting parameters can be challenging

Classification Models
Decision Trees : Summary

Model
Set of rules to partition data set based on values of different predictors

Input
No restriction on variable type for predictors

Output
Label categorical for classification tree, numeric for regression tree

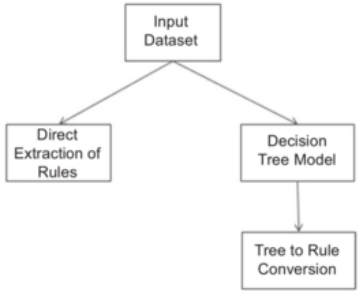
Pros
Intuitive to explain to non-technical business users. Normalising predictors not necessary. Minimal data preparation.

Cons
Tends to over-fit the data. Small changes in input data can yield substantially different trees. Selecting right parameters can be challenging. Divides dataset in rectilinear fashion

Use Cases
Marketing segmentation, fraud detection

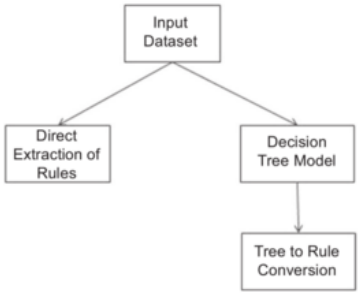
Classification Models
Rule Induction

- Rule Induction is a data mining process of deducing if-then rules from a data set.
- Rule Induction models are used both for predictive classification of unknown data as well as for descriptive explanations of patterns in data
- They are commonly used in predicting and preventing machine breakdowns, in many cases superseding the rules of thumb of the machine expert. They are both easily deployed by automated systems as well as easily understandable to line workers
- Rules can be directly extracted from the data set or derived from the previously built decision trees from the same data set



Approaches to Rule Generation adapted from (Kotu and Deshpande, 2019, p.90)

Classification Models
Rule Induction - Cont.



Approaches to Rule Generation adapted from (Kotu and Deshpande, 2019, p.90)

- Tree to Rule Conversion is a passive or indirect approach that derives a rule set from a previously built classifier decision tree model
- Direct Extraction uses sequential covering, an iterative procedure of extracting rules from the data set. The RIPPER Algorithm (Repeated Incremental Pruning to Produce Error Reduction) is an implementation of this method
- RIPPER is a “greedy” algorithm that splits the data set typically in a rectilinear fashion but also uses complex and curved decision boundaries to generate mutually exclusive and exhaustive rule sets.

Classification Models
Rule Induction : How to Implement

Step 1 – Data Preparation

Rule Induction can input both numeric and polynominal data types
Split dataset into training and test sets (equally using shuffled sampling)

Step 2 – Modelling and Parameters

Tree to Rules for derivation from decision tree model or
Rule Induction RIPPER with initial parameters (criterion = Information Gain, sample ratio = 0.9, pureness = 0.9, minimal prune benefit = 0.25)

Step 3 – Implement Evaluation Method

Apply Model learned on Training data set to the Test data set
Evaluate **Performance** comparing the predicted class with the labelled class for all of the Test records

Step 4 – Execution and Interpretation

Review the Output Results including:-

- **Rule Model** listing rules (note class distribution covered from the training data set is shown in parentheses after the rules)
- **Performance vector** (confusion matrix with correct and incorrect predictions for all of the dataset)

Classification Models
Induction Rules : Use Case

PREDICTING AND PREVENTING MACHINE BREAKDOWNS

A machine breakdown in the field almost always results in disruption of a manufacturing process. In a large-scale process like an oil refinery, chemical plants, etc., it causes serious financial damage to the company and manufacturers of the machines. Let's assume the machine under consideration is a motor. Rather than waiting for the machine to break down and react, it is much preferable to diagnose the problem and prevent the breakdown before a problem occurs. Large-scale machines track thousands of real-time readings from multiple parts of the machine (Such machines connected to networks that can gather readings and act based on smart logic are called Internet of Things (IoT).). One of the solutions is to leverage how these readings are trending and develop a rule base which says, for example, *if the cylinder temperature continues to report more than 852°C, then the machine will break down in the near future*. These types of the rules are simple to interpret, don't require an expert to be around to take further action, and can be deployed by automated systems.

Developing learned rules requires historical analysis of all the readings that lead up to a machine failure (Langley & Simon, 1995). These learned rules are different and in many cases supersede the rule of thumb assumed by the machine expert. Based on the historic readings of a failure event and nonfailure events, the learned rule set can predict the failure of the machine and hence can alert the operator of imminent future breakdowns. Since these rules are very simple to understand, these preventive measures can be easily deployed to line workers. This use case demonstrates the need of not only a predictive data model, but also a descriptive model where the inner working of the model can be easily understood by the users. A similar approach can be developed to prevent customer churn, or loan default, for example.

(Kotu and Deshpande, 2019, p.90)

Classification Models
Rule Induction : Discussion Points

- Rule learners are the simplest form of data mining model to indicate the most powerful predictor in a given set of attributes
- Rules can overlearn the example set and pruning may be necessary. Also, like other “greedy” algorithms the result may not always be the most globally optimal
- Rule Induction is commonly used in predicting and preventing machine breakdowns, in many cases superseding the rules of thumb of the machine expert. Rules derived are both easily deployed in automated systems as well as being easily understandable to line workers

Classification Models
Rule Induction : Summary

Model
Set of rules that contain an antecedent (inputs) and consequent (output class)

Input
No restrictions. Accepts categorical, numeric and binary inputs

Output
Prediction of target variable which is categorical

Pros
Model can be easily explained to business users. Easy to deploy in almost any tools and applications

Cons
Divides dataset in rectilinear fashion

Use Cases
Manufacturing, applications where description of model is necessary

Classification Models
Naïve Bayesian

Naïve Bayes algorithm is drawn from statistics and probability theory.

Named after Reverend Thomas Bayes “Essay Towards Solving a Problem in the Doctrine of Chances” (1763). One of most influential and important concepts in statistics and probability theory, it provides a mathematical expression for how a degree of subjective belief changes to account for new evidence.

It leverages the probabilistic relationship between the factors (attributes) and the class label (outcome).

Classification Models
Naïve Bayesian

Example

The likelihood of the average person defaulting on their mortgage loan is 2%. However, if their credit history is excellent, then the likelihood of their default would be less than average. Further, if their annual salary is above average w.r.t. loan value, then the likelihood of default falls further.

As more evidence on the factors impacting the outcome are obtained, improved guesses can be made about the outcome using probability theory.

Classification Models
Naïve Bayesian

The algorithm makes a **strong, "naïve" assumption** about the independence between the attributes which may not always hold true.

For Example: -
In some cases, we can assume annual income and credit score are independent of each other, however, in many cases we just don't know. If one of the factors for the default rate is home value, then both the annual income and home value factors are correlated and not independent. Homeowners with high income tend to buy more expensive houses.

However, in many cases, the simplicity and robustness of the algorithm offsets the limitations around the independence assumption.

Classification Models
Naïve Bayesian : How it Works

Assume

X (X_1, X_2,…) is the evidence (attribute set), Y the outcome (label class)

$$P(Y|X) = \frac{P(Y) * P(X|Y)}{P(X)} \qquad P(Y|X) = \frac{P(Y) * \prod_{i=1}^{n} P(X_i|Y)}{P(X)}$$

P(Y) is the **prior probability**, the default rate of a home mortgage (2%)

P(Y|X) is the **conditional or posterior probability**, the average rate of default given that an individual's credit history is known.

P(X|Y) is the **class conditional probability**, the probability of an excellent credit rating given that the default is a "yes"

P(X) is basically the probability of the evidence, the proportion of individuals with a given credit rating. Since it is the same for every class value, we don't have to calculate it and can assume it as constant

To classify a new record, we compute P(Y|X) for each class of Y and see which probability "wins"

Classification Models
Naïve Bayesian : How it Works – Golf Dataset

Step 1: Calculating Prior Probability P(Y)

Note: Important that the data set used for data mining is **representative** of the population if sampling is used. Random sampling will not be compatible.

No.	Temperature X_1	Humidity X_2	Outlook X_3	Wind X_4	Play (Class Label) Y
1	high	med	sunny	false	no
2	high	high	sunny	true	no
3	low	low	rain	true	no
4	med	high	sunny	false	no
5	low	med	rain	true	no
6	high	med	overcast	false	yes
7	low	high	rain	false	yes
8	low	med	rain	false	yes
9	low	low	overcast	true	yes
10	low	low	sunny	false	yes
11	med	med	rain	false	yes
12	med	low	sunny	true	yes
13	med	high	overcast	true	yes
14	high	low	overcast	false	yes

P(Y = no) = 5/14
P(Y = yes) = 9/14

Classification Models
Naïve Bayesian : How it Works – Golf Dataset - Cont.

Step 2: Calculating Class Conditional Probability P(X_i|Y)

Temperature (X_1)	P(X_1|Y = no)	P(X_1|Y = yes)
high	2/5	2/9
med	1/5	3/9
low	2/5	4/9

Humidity (X_2)	P(X_1|Y = no)	P(X_1|Y = yes)
high	2/5	2/9
low	1/5	4/9
med	2/5	3/9

Outlook (X_3)	P(X_1|Y = no)	P(X_1|Y = yes)
overcast	0/5	4/9
Rain	2/5	3/9
sunny	3/5	2/9

Wind (X_4)	P(X_1|Y = no)	P(X_1|Y = yes)
false	2/5	6/9
true	3/5	3/9

(Kotu and Deshpande, 2019, p.115-116)

Classification Models
Naïve Bayesian : How it Works – Golf Dataset - Cont.

No.	Temperature X_1	Humidity X_2	Outlook X_3	Wind X_4	Play (Class Label) Y
Unlabeled Test	high	low	sunny	false	?

(Kotu and Deshpande, 2019, p.116)

P(Y= yes | X) = P(Y = yes) *
{P(Temp = high | Y = yes) * P(Humidity = low | Y = yes) *
P(Outlook = sunny | Y = yes) * P(Wind = false | Y =yes)} /
P(X)
= 9/14 * {2/9 * 4/9 * 2/9 * 6/9} / P(X)
= 0.0094 / P(X)

P(Y = no | X) = 5/14 * {2/5 * 4/5 * 3/5 * 2/5}
= 0.0274 / P(X)

Normalise both estimates by dividing both by (0.0094 + 0.0274)

Likelihood of (Play = yes) = 26%
Likelihood of (Play = no) = 74%

Hence the prediction for the unlabelled test record will be Play = no

Classification Models
Naïve Bayesian : Handling Issues

Issue 1: Incomplete Training Set
Problems when an attribute value in the testing record has no example in the training record resulting in a zero-class conditional probability and a zero-posterior probability. To mitigate a technique called **Laplace correction** is applied which adds a controlled error in all class conditional probabilities so as to avoid a zero case

Issue 2: Continuous Attributes
Where the values are not nominal values, probability densities rather than probabilities can be used

Play Value		Humidity X_2	Temperature X_3
Y = no	Mean	74.60	84.00
	Deviation	7.89	9.62
Y = yes	Mean	73.00	78.22
	Deviation	6.16	9.88

$$P(\text{temperature} = 78|Y = \text{yes}) = 0.04$$
$$P(\text{temperature} = 78|Y = \text{no}) = 0.05$$

Issue 3: Attribute Independence
The independence of two categorical (nominal) attributes can be tested by the chi-square test for independence and strongly correlated attributes removed

Classification Models
Naïve Bayesian : How to Implement

Step 1 – Data Preparation

Naïve Bayesian can input both numeric and nominal attributes

Split dataset into training and test sets using random sampling as it is especially important for the training set to be representative and proportional to the underlying data set

Step 2 – Modelling and Parameters

Naive Bayes has one check parameter (inclusion of Laplace correction is recommended for smaller data sets and is the default)

Step 3 – Implement Evaluation Method

Apply Model learned on Training data set to the Test data set

Evaluate **Performance -Classification** comparing the predicted class with the labelled class for all of the Test records

Step 4 – Execution and Interpretation

Model Description with probability density functions for input attributes and distribution tables

Performance vector

Labelled data set with confidence for each label class which indicates the likelihood of each label class value

Classification Models
Naïve Bayesian : Discussion Points

- Robust against outliers and missing values
- Stratified sampling required as the training data set must be proportionately representative of the underlying data
- Used in text and document mining. Foundational element in spam email detection
- Good for use in proof of concept and benchmarking against other models
- Co-dependent attributes should be removed during pre-processing
- Leverages new information as it arrives and tries to make best prediction considering new evidence

Classification Models
Naïve Bayesian : SPAM Detection

PREDICTING AND FILTERING SPAM EMAIL

Spam is unsolicited bulk email sent to a wide number of email users. At best it is an annoyance to recipients but many of the spam emails hide a malicious intent by hosting false advertisements or redirecting clicks to phishing sites. Filtering spam email is one of the essential features provided by email service providers and administrators. The key challenge is balance between incorrectly flagging a legitimate email as spam (false positive) versus not catching all the spam messages. There is no perfect spam filtering solution and spam detecting is a catch-up game. The spammers always try to deceive and outsmart the spam filters and email administrators fortify the filters for various new spam scenarios. Automated spam filtering based on algorithms provides a promising solution in containing spam and a learning framework to update the filtering solutions (Prosess Software, 2013).

Some words occur in spam emails more often than in legitimate email messages. For example, the probability of occurrence for words like free, mortgage, credit, sale, Viagra, etc. is higher in spam mails than in normal emails. We can calculate the exact probabilities if we have a sample of previously known spam emails and regular emails. Based on the known word probabilities, we can compute the overall probability of an email being spam based on all the words in the email and the probability of each word being in spam versus regular emails. This is the foundation of Bayesian spam filtering systems (Zdziarski, 2005). Any wrongly classified spam messages that are subsequently reclassified by the user is an opportunity to refine the model, making spam filtering adaptive to new spam techniques. Though recent spam reduction uses a combination of different algorithms, Bayesian-based spam filtering remains one of the foundational elements of spam prediction systems (Sahami et al., 1998).

(Kotu and Deshpande, 2019, p112-113)

Classification Models
Naïve Bayesian : Summary

Model

A lookup table of probabilities and conditional probabilities for each attribute with an output class

Input

No restrictions. However, probability calculation works better with categorical attributes

Output

Prediction of probability for all class values, along with the winning class

Pros

Time required to model and deploy is minimum. Great algorithm for benchmarking. Strong statistical foundation

Cons

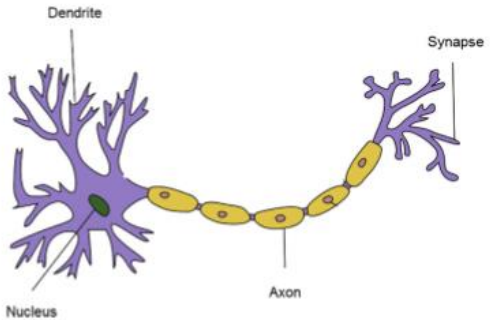
Training dataset needs to be representative sample of population and needs to have complete combinations of input and output. Attributes need to be independent

Use Cases

Spam detection, text mining

Classification Models
Artificial Neural Networks

A mathematical process that closely resembles the biological process of a neuron. There are 100 billion neurons in the human brain. Neurons are composed of a cell body, dendrite, and axon. Neuron cells transmit information through electrical and chemical signals. The interconnection between one neuron and another happens through a synapse, and electrochemical signals are sent from one neuron to another. There are about 100 trillion synapses in a human brain.

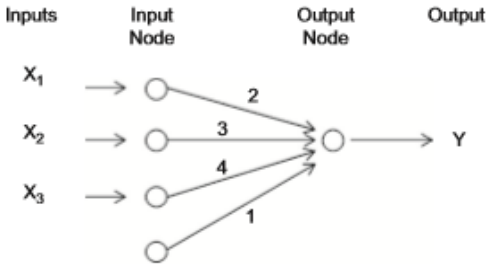


Anatomy of a neuron. Modified from original "Neuron Hand-tuned." Original uploader: Quasar Jarosz at en.wikipedia.org. Transferred from en.wikipedia.org to Commons by user Faigl.ladislav using CommonsHelper. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons

Classification Models
Artificial Neural Networks cont.

In an Artificial Neural network, a neuron is a node. An ANN models the relationship between input and output variables by developing a mathematical explanation that closely resembles the biological process of a neuron. Neural networks **learn** through adaptive adjustments of weights between nodes

$$Y = 1 + 2X_1 + 3X_2 + 4X_3$$

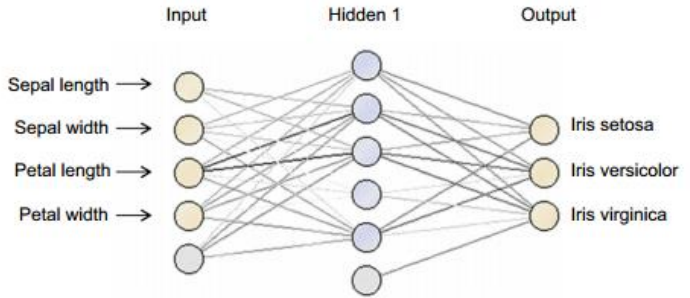


Model topology adapted from (Kotu and Deshpande, 2019, p.125)

In this topology X_1 is the input value and passes through a node, denoted by a circle. Then the value of X_1 is multiplied by its weight (2). Similarly, all other attributes (X_2 and X_3) go through a node and scaling transformation. The Output layer performs aggregation, transfer (scale to range), and activation functions. This simple two-layer topology is called a perceptron.

Classification Models
Artificial Neural Networks cont.

An Artificial Neural network is typically used for modelling non-linear, complex relationships between input and output variables. This is made possible by the introduction of **hidden layers**. A hidden layer applies an activation function. The topology of a neural network for modelling the Iris data using a feed-forward artificial neural network with one hidden layer is shown below.



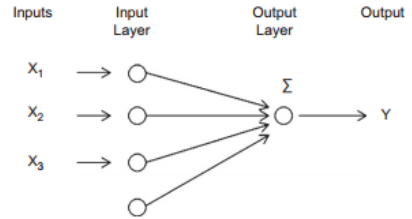
Topology of a neural network model for Iris dataset adapted from (Kotu and Deshpande, 2019, p.127)

The activation function used in the output function allows for a linear transformation for a particular range of values and a non-linear transformation for the rest of the values. As almost any mathematical continuous relationship between input and output variables can be modelled, a multi-layer ANN is called a **universal approximator.**

Classification Models
Artificial Neural Networks : How it Works

Step:1 Determine the Topology and Activation Function

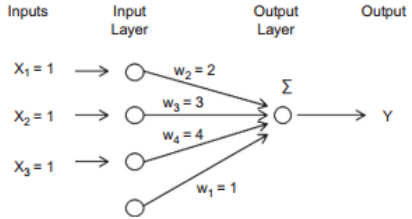
Assume a data set with three numeric input attributes (X_1, X_2, X_3) and one numeric output Y. To model the relationship, use a topology of two layers with a simple aggregation function (no transfer function in this case).



Two layer toplogy with summary aggregation adapted from (Kotu and Deshpande, 2019, p.128)

Step 2: Initiation

Assume weights as below and a training record with all the inputs as 1 and known output as 15.

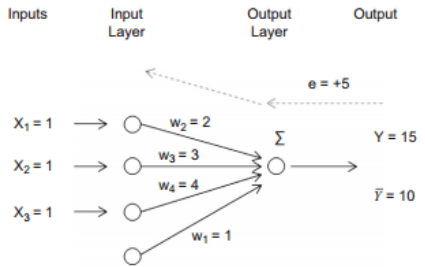


Initiation and first training record adapted from (Kotu and Deshpande, 2019, p.129)

Classification Models
Artificial Neural Networks : How it Works cont.

Step:3 Calculating Error

Predicted output is 10. Actual output is 15. Therefore, error from example training record is 5.



Neural network error back propagation adapted from (Kotu and Deshpande, 2019, p.130)

Classification Models
Artificial Neural Networks : How it Works cont.

Step 4: Weight Adjustment through Back-propagation

$$w = w' + \lambda * e$$

Weight adjustment is the most important part of learning. Error calculated is passed back in reverse direction. Weights of the links are adjusted by a **fraction ƛ** of the error. **ƛ** is called the learning rate and takes a value from 0 to 1. A value close to 1 results in a drastic change to the model for each training record whereas a value close to 0 results in smaller changes and less correction (typically applied nearing the end of the learning cycle to avoid sensitivity to outliers from degrading model at this stage). The new weight is the sum of the old weight and the product of learning rate and proportion of error. This cycle continues until all training examples have been iteratively processed

Classification Models
Artificial Neural Networks : How to Implement

Step 1 – Data Preparation

Using Iris data set, the four attributes are all numeric (no transformation to numeric required) and output has three classes. The ANN will have four input and three output nodes.

Split dataset into training and test sets equally

Step 2 – Modelling and Parameters

Neural Net with default parameter values for **Hidden Layer** (number layers, size hidden layer, names layers), **Training cycles** (number times a training cycle repeated), **Learning rate (ƛ)**. [0,1]. Value 0 means that the new weight will be more based on previous weight. Value 1 means that the new weight will mainly be based on the error correction), **Momentum** (prevents local maxima by adding a fraction of previous weight to current weight, **Decay** (makes value of learning rate closer to zero for last few records), **Shuffle** (random sort sequence), **Normalize** (real values of input), **Error epsilon** (error threshold disallowing error reducing to zero to prevent over-fitting.

Classification Models
Artificial Neural Networks : Discussion Points

- ANN require strict pre-processing for dealing with missing values. Categorical data must be converted to binary or real values.
- Redundant correlated attributes not a problem for ANN
- Can yield local optima, need to set the momentum parameter to weight the update
- Complex to understand the model. In some data mining applications the explanation of the model is as important as the model itself and in these cases decision trees, induction rules and regression better
- Building a good ANN model with optimised parameters takes time. No consistent guidelines on number hidden layers and nodes with each hidden layer

Classification Models
Artificial Neural Networks : Discussion Points

- Once built, straightforward to implement with fast execution
- ANN good option for modelling highly non-linear relations with fast real-time performance

Classification Models
ANN : Optical Character Recognition

OPTICAL CHARACTER RECOGNITION

Character recognition is the process of interpreting handwritten text and converting it into digitized characters. It has a multitude of practical applications in our everyday life, including converting handwritten notes to standardized text, automated sorting of postal mail by looking at the zip codes (postal area codes), automated data entry from forms and applications, digitizing classic books, license plate recognition, etc. How does it work?

In its most basic form, character recognition has two steps: digitization and development of the learning model. In the digitization step, every individual character is converted to a digital matrix, say 12x12 pixels, where each cell takes a value of either 0 or 1 based on the handwritten character overlay. The input vector now has 144 binary attributes (12x12) indicating the information of the handwritten characters. Let's assume the objective is to decipher a numeric handwritten zip code (Matan & Kiang, 1990). We can develop an artificial neural network model which accepts 144 inputs and has 10 outputs, each indicating a digit from 0 to 9. The model has to be learned in such a way that when the input matrix is fed, one of the outputs shows the highest signal indicating the prediction for the character. Since a neural network is adaptable and relatively easy to deploy, it is increasingly getting used in character recognition, image processing, and related applications (Li, 1994). This specific use case is also an example where the explanatory aspect of the model is less important—maybe because no one knows exactly how the human brain does it. So there is less expectation that the model should be understandable as long as it works with acceptable performance. This also means ANN models are not easy to explain and in many situations this alone will remove them from consideration of the data mining techniques to use. We wish this wasn't the case!

(Kotu and Deshpande, 2019, p127)

Python Resource - https://github.com/tensorflow/tensorflow

Classification Models
Artificial Neural Networks : Summary

Model
A network topology of layers and weights to process input data

Input
All attributes should be numeric

Output
Prediction of target (label) variable, which is categorical

Pros
Good at modelling **nonlinear** relationships. Fast response time in deployment

Cons
No easy way to explain the inner working of the model. Requires pre-processing data. Cannot handle missing attributes

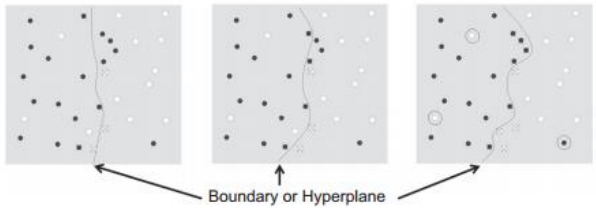
Use Cases
Image recognition, fraud detection, quick response time applications

Classification Models
Support Vector Machines

- Works on principle of fitting a boundary on a training sample to a region of points that are all alike (same class). Any new points are then either inside or outside this boundary. Once the boundary is established, all that is needed is the core set of points that can help identify and set the boundary

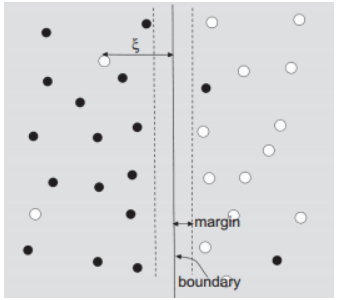
- There are a number of hyperplanes possible as below. The boundary which separates the points with the minimal misclassification is the best one. A boundary line that ensures that the average geometric distance between two regions (or classes) is maximised is even better. This distance is called the **margin** and the SVM Algorithm essentially runs an optimisation to maximise this margin.

Three different hyperplanes for the same set of training data. There are two classes in this dataset, which are shown as filled and open circles adapted from (Kotu and Deshpande, 2019, p.136)



Classification Models
Support Vector Machines cont.

Where it is not possible to cleanly separate the data, a penalty ξ is charged for every "contaminant" and the hyperplane with the minimum aggregate penalty is chosen.



Key concepts in SVM construction: boundary, margin, and penalty, ξ. SVM, Support Vector Machine adapted from (Kotu and Deshpande, 2019, p.137)

Classification Models
Support Vector Machines cont.

Where it is not possible to linearly separate the data, the data can be transformed with a **kernel** function (polynomial, radial, sigmoid) to enable linear separability.

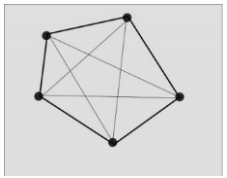
Below z = $\sqrt{(x^2 + y^2)}$

(Left) Linearly non-separable classes. (Right) Transformation to linearly separable adapted from (Kotu and Deshpande, 2019, p.137)

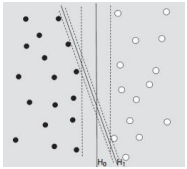
Classification Models
Support Vector Machines : How it Works

Step:1 Finding the boundary

Connect every point in one class of a data set to every other in that class the outline that emerges defines the boundary or convex hull of the class. Each class will have its own convex hull and because the classes are (assumed to be) linearly separable, the hulls do not intersect



(Left) A convex hull for one class of data adapted from (Kotu and Deshpande, 2019, p.139)



(Left) Both hyperplanes shown can separate data. It is intuitively clear that H0 is better adapted from (Kotu and Deshpande, 2019, p.140)

Step 2: Hyperplanes

Infinitely many hyperplanes exist expressed as $H = b_0 + w_0 \cdot x = 0$ where X is (X_1, X_2), the weight is (w_1, w_2) and b_0 is an intercept like term, the bias. The optimal hyperplane is found by maximising the margin = $2/(\sqrt{w_0} \cdot w_0)$ using quadratic programming, the w_0 can be expressed in terms of only a few of the training examples known as support vectors.

Step 3: Classify new test example

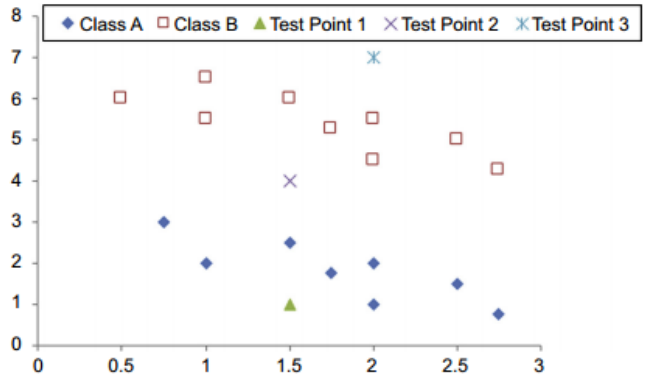
Substitute the test example x into the equation for the hyperplane. If it computes to +1 then it belongs to the positive class and if it computes to -1 then it belongs to the negative class.

Classification Models
SVM : How to Implement (linearly separable example)

A Simple Data Set to demonstrate SVM

Table 4.11 A Simple Data Set to demonstrate SVM

x_1	x_2	class
1.5	2.5	A
2	2	A
1	2	A
0.75	3	A
2	1	A
1.75	1.75	A
2.75	0.75	A
2.5	1.5	A
0.5	6	B
1.5	6	B
2	5.5	B
1	5.5	B
1	6.5	B
2	4.5	B
1.75	5.25	B
2.75	4.25	B
2.5	5	B
1.5	1	Test Point 1
1.5	4	Test Point 2
2	7	Test Point 3



04_Class_4.6_SVM_simple_example

Classification Models
SVM : How to Implement (linearly separable example)

04_Class_4.6_SVM_simple_example.rmp

Step 1 – Data Preparation

Ensure Class is a label of 04_Class_4.6_SVM_simple_example

Step 2 – Modelling and Parameters

SVM with default parameters

Step 3 – Implement Evaluation Method

Apply Model learned on Training data set applied on Test data set

Step 4 – Execution and Interpretation

Kernel Model output (b_0 = 0.051, w_1 = 0.370, w_2 = 1.351)

Output shows model has correctly classified (2, 7) as belonging to B class with confidence 92.6%, and (1.5, 1) to class A with confidence 88%. (1.5, 4) has been classified to class B with confidence 53.4%

Classification Models
SVM : How to Implement (linearly nonseparable example)

A Data Set to demonstrate non-linear SVM

04_Class_4.6_SVM_nonLinear_example

```
"x1","x2","y=x1SQ+x2SQ","ring"
2.417820342,-5.161796663,5.1,"inner"
1.087812139,5.391350921,5.7,"inner"
-4.999219209,0.088358927,5.9,"inner"
5.475108007,1.913946789,5.1,"inner"
4.718194075,1.936141697,5.9,"inner"
-4.94971385,2.619223703,5.6,"inner"
0.234943194,-5.895320322,5.7,"inner"
4.287846414,-3.115184285,5.3,"inner"
-5.292535862,0.281183472,5.9,"inner"
4.070904853,2.903055921,5.0,"inner"
-5.74195648,0.818496047,6.0,"inner"
0.85827424,5.83723953,5.0,"inner"
-4.236111995,4.106745081,5.5,"inner"
-1.635977329,5.355705199,6.0,"inner"
5.128259569,1.691435425,5.8,"inner"
-1.704098552,-4.806875089,6.0,"inner"
-1.278136317,5.349426844,5.9,"inner"
-2.554019612,5.095780983,5.2,"inner"
1.421831001,-5.726115315,5.8,"inner"
3.288772083,4.655531998,5.6,"inner"
2.455104614,5.364928828,5.6,"inner"
-0.665411186,-5.157250038,5.3,"inner"
-0.482436264,-5.177572332,5.2,"inner"
```

Classification Models
SVM : How to Implement (linearly nonseparable example)

04_Class_4.6_SVM_nonLinear_example.rmp

Step 1 – Data Preparation

Ensure Class is a label of 04_Class_4.6_SVM_nonLinear_example
Label **Ring** attribute, **Select attributes** (include special attributes checked), **Split Validation** (split ratio 0.7, sampling type stratified)

Step 2 – Modelling and Parameters

Split Validation (SVM) kernel type polynomial, default degree 2.0

Step 3 – Implement Evaluation Method

Apply Model learned on Training data set to the Test data set (30%)
Performance Classification

Step 4 – Execution and Interpretation

Without **kernel polynomial** the linear SVM gets barely 50% correct. With kernel polynomial able to classify points 100% accuracy

Classification Models
Support Vector Machines : Discussion Points

- Relatively recent technique (1995 AT&T Bell Labs)
- Draws equally from three major disciplines : computer science, statistics, and mathematical optimisation
- SVM's have been applied successfully to applications from image processing to fraud detection to text mining
- Robust model, small changes in data does not require expensive remodelling
- Resistant to overfitting, the boundary of classes within data sets can adequately be described by only a few support vectors

Classification Models
Support Vector Machines : Discussion Points

- Labelled test data with prediction confidences are the most useful results
- High computational costs

Classification Models
Support Vector Machines: Summary

Model

A vector equation that allows us to classify new data points into different regions (classes)

Input

All attributes should be numeric

Output

Prediction of target (label) variable, which can be categorical or numeric

Pros

Very robust against over-fitting. Small changes to input data do not affect boundary and thus do not yield different results. Good at handling nonlinear relationships

Cons

Computational performance during training phase can be slow. This may be compounded by the need to optimise parameter combinations

Use Cases

Optical character recognition, fraud detection, modelling “black swan” events

Classification Models
Ensemble Models

- In Supervised learning a model or hypothesis is selected (from an infinite number of hypotheses) that can map new input data to predicted output with the least error. The selected hypothesis can over-fit on a particular training set resulting in increased prediction error rates on unseen data. Different Classification Algorithms can also be biased towards certain attributes(features)

- Ensemble models improve the error rate and reduce the bias of individual models by aggregating (voting between) the predictions of several base models to produce the final model. The result is that a ‘strong’ learner can be produced from several ‘weak’ learners. Weighting each vote by the accuracy rate of the underlying base model can also give higher ‘say’ to the models with higher accuracy rates

- Most important technique for many practical classification problems

Classification Models
Ensemble Models Use Case

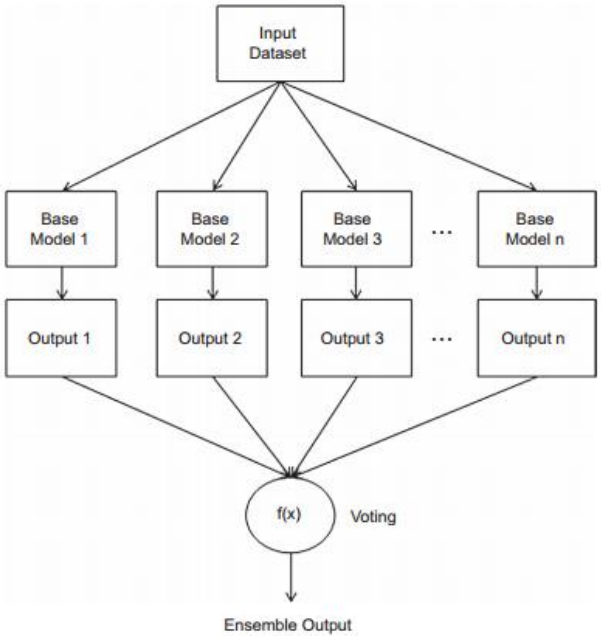
PREDICTING DROUGHT

Drought is a period of time where a region experiences far less than average water supply. With the onset of climate change, there has been an increase in frequency and duration of drought conditions in many parts of the world. Immediate drought is caused by the development of high-pressure regions, which inhibits the formation of clouds, which results in low precipitation and lower humidity. Predicting drought conditions in a region is a very challenging task. There is no clear start and end point for draught duration. There are too many variables that impact the climate patterns that lead to drought conditions. Hence, there is no strong model to predict drought well ahead of time (Predicting Drought, 2013). Predicting drought seasons in advance would provide time for regional administrations to mitigate the consequences of the drought.

Droughts involve a myriad factors including groundwater level, air stream flow, soil moisture, topology, and large-scale global weather patterns like El Nino and La Nina (Patel, 2012). With thousands of attributes and many unknown variables that influence the conditions for drought, there is no "silver bullet" massive model for predicting when drought is going to hit a region with a high degree of accuracy. What we have is many different "weak" models that use some of the thousands of attributes available, which make predictions marginally better than pure chance. These weak models may provide different drought predictions for the same region and time, based on the diverse input variables for each model. We can summarize the prediction by combining the predictions of individual models and take a vote. Ensemble models provide a systemic method to combine many weak models into one better model. Most of the data mining models deployed in production applications are ensemble models. Ensemble models greatly reduce generalization errors and improve the accuracy of the overall prediction, if certain conditions are met.

(Kotu and Deshpande, 2019, p149-150)

Classification Models
Ensemble Models cont.



Ensemble model
adapted from (Kotu and Deshpande, 2019, p.149)

Classification Models
Ensemble Models : How they Work

In the binomial distribution, the probability of k successes in n independent trials each with a success rate of p is given by a probability mass function,

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$ and $$P(\text{ensemble wrong}) = P(k \geq \text{round}(n/2)) = \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k}$$

Example
On a board of 3 members where P(error board member) = 20%

P(error board) = P(2 members wrong) + P(3 members wrong) = 10.4%

Important that each member of the ensemble should be independent (diverse) and that the individual model error rate should be less than 50% (random chance) for binary classifiers.

Classification Models
Ensemble Models : How they Work

Ways to ensure base models are diverse: -

- Use different algorithms for building the models
- Use different algorithm parameters for different models
- Use different examples for training
- Use different sets of attributes from the training examples

Methods include for example:-
Voting, Bootstrap Aggregating (Bagging), Boosting (eg. AdaBoost) and Random Forests.

Classification Models
Ensemble Voting : How to Implement

Uses different algorithms for the base classifiers using the same training data set that has been pre-processed for all algorithms.

accuracy: 94.67%

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	20	0	0	100.00%
pred. Iris-versicolor	0	26	3	89.66%
pred. Iris-virginica	0	1	25	96.15%
class recall	100.00%	96.30%	89.29%	

04_Class_4.7_ensemble_voting.rmp

Classification Models
Ensemble Bagging : How to Implement

The Bagging method combines multiple hypotheses on the **same** input data prepared using **Bootstrapping** or sampling with replacement. Each base training set contains about 63% unique training records when compared to the original training set. Bagging improves the stability of unstable models that are highly dependent on even slight changes in the input data (decision trees and neural networks). Final predictions are by simple voting aggregation. Parameters **sample ratio** (train) and **iterations** (number base models).

accuracy: 93.33%

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	20	0	0	100.00%
pred. Iris-versicolor	0	26	4	86.67%
pred. Iris-virginica	0	1	24	96.00%
class recall	100.00%	96.30%	85.71%	

04_Class_4.7_ensemble_bagging.rmp

Classification Models
Ensemble Boosting : How to Implement

Boosting assigns a weight to each training record that is adaptively changed based on difficulty of classification. This results in an ensemble of models specialised in classifying both easy-to-classify and hard-to-classify records. A simple voting aggregation is performed for the final prediction.

accuracy: 94.67%

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	30	0	0	100.00%
pred. Iris-versicolor	0	22	3	88.00%
pred. Iris-virginica	0	1	19	95.00%
class recall	100.00%	95.65%	86.36%	

04_Class_4.7_ensemble_adaboost.rmp

Classification Models
Ensemble Boosting AdaBoost : Discussion Points

- AdaBoost is one of the most popular implementations of the boosting ensemble approach.

- An AdaBoost model assigns uniform weights to each training record to start and then iteratively, following the build of the next base classifier, updates the weights of the training records according to both the accuracy of the base classifier and the error rate of the predictions of the base classifier

- If the error rate is greater than 50%, the record weight for a data record is not updated and remains unchanged for the next base classifier build

Classification Models
Ensemble Random Forest : How to Implement

Random Forest does random selection of both attributes and training records. For each base ensemble decision tree model, a random sample is selected with replacement and then a random subset of all the attributes in the training set is considered when deciding how to split each node of the tree. Once all trees are built, for each new record, all the trees predict a class and vote with equal weight.

accuracy: 93.33%

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	30	0	0	100.00%
pred. Iris-versicolor	0	23	5	82.14%
pred. Iris-virginica	0	0	17	100.00%
class recall	100.00%	100.00%	77.27%	

04_Class_4.7_ensemble_randomforest.rmp

Classification Models
Discussion Points

- Most of the data mining models developed for production applications are built on ensemble models
- Random Forest models are very useful as baseline ensemble model for comparative purposes
- Applied in a wide range of applications including political forecasting, weather pattern modelling, media recommendation, and web page ranking
- All techniques (voting, bagging, boosting, random forest) have been proven to perform better than base models as long as the base models are diverse. Wisdom of the crowds makes sense as long as “group thinking” is controlled by promoting independence amongst base models

Classification Models
Ensemble Models : Summary

Model

A meta-model with individual base models and an aggregator

Input

Superset of restrictions from the base model used

Output

Prediction for all class values with a winning class

Pros

Reduces the generalisation error. Takes different search space into consideration

Cons

Achieving model independence is tricky. Difficult to explain the inner working of the model

Use Cases

Most of the practical classifiers are ensemble

References

- Han, J., Pei, J., Tong, H. (2022) *Data Mining Concepts and Techniques.* 4th edn. Burlington, MA: Morgan Kaufmann Publishers.
- Kelleher, J., MacNamee, B., D'Arcy, A. (2015) *Fundamentals of Machine Learning for Predictive Analytics.* Cambridge, MA: The MIT Press.
- Kotu, V. and Deshpande, B. (2019) *Data Science Concepts and Practice.* 2nd end. Burlington, MA: Morgan Kaufmann Publishers.
- Marr, B. (2022) *Data Strategy: How to Profit from a World of Big Data, Analytics and Artificial Intelligence.* 2nd edn. London: Kogan Page.
- Shearer, C. (2000) 'The CRISP-DM Model: The New Blueprint for Data Mining', *Journal of Data Warehousing*, 5(4), pp. 13-24.

References – Unit 3 – Churn Modelling

- Kelleher, J., MacNamee, B., D'Arcy, A. (2015) *Fundamentals of Machine Learning for Predictive Analytics.* Cambridge, MA: The MIT Press.

- Marr, B. (2022) *Data Strategy: How to Profit from a World of Big Data, Analytics and Artificial Intelligence.* 2nd edn. London: Kogan Page.

- Shearer, C. (2000) 'The CRISP-DM Model: The New Blueprint for Data Mining', *Journal of Data Warehousing*, 5(4), pp. 13-24.

Base Academic Papers

- Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46(3), 175185.
- Random forests. Machine Learning, 45, 532. Brieman, L. F. (1984). Classification and regression trees. Chapman and Hall.
- Cohen W.W. (1995). Fast effective rule induction. Machine learning. In: Proceedings of the twelfth international conference.
- Cortes, C. A. (1995). Support vector networks. Machine Learning, 273297.
- Cover, T. A. (1991). Entropy, relative information, and mutual information. In T. A. Cover (Ed.), Elements of information theory (pp. 1249). John Wiley and Sons.

Base Academic Papers

- Dietterich, T.G. Ensemble methods in machine learning. (2007). Retrieved from http://www.eecs. wsu.edu/Bholder/courses/CptS570/fall07/papers/Dietterich00.pdf..
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. Annals of Human Genetics, 7, 179188. Available from https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.
- Fletcher, R. (1987). Practical methods of optimization. New York: John Wiley
- Gashler, M., Giraud-Carrier, C., & Martinez T. (2008) Decision tree ensemble: small heterogeneous is better than large homogeneous. In: 2008 Seventh international conference on machine learning and applications (pp. 900905). doi:10.1109/ICMLA.2008.154.

Base Academic Papers

- Laine, A. (2003). Neural networks. Encyclopedia of computer science (4th ed., pp. 12331239). John Wiley and Sons Ltd.
- Langley, P., & Simon, H. A. (1995). Applications of machine learning and rule induction. Communications of the ACM, 38(11), 5464. doi:10.1145/219717.219768.
- Li, E. Y. (1994). Artificial neural networks and their business applications. Information & Management, 27(5), 303313. Available from https://doi.org/10.1016/0378-7206(94)90024-8.
- Matan, O., et al. (1990). Handwritten character recognition using neural network architectures. In: 4th USPS advanced technology conference (pp. 10031011)

Base Academic Papers

- Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., & Euler, T. (2006). YALE: Rapid prototyping for complex data mining tasks. In: Proceedings of the ACM SIGKDD international conference on knowledge discovery and data mining (Vol. 2006, pp. 935940). doi:10.1145/1150402.1150531.
- Montgomery, J. M., Hollenbach, F. M., & Ward, M. D. (2012). Improving predictions using ensemble Bayesian model averaging. Political Analysis, 20(3), 271291.
- Patel, P. Predicting the future of drought prediction. IEEE Spectrum. (2012). ,http://spectrum.ieee.org/energy/environment/predicting-the-future-of-drought-prediction. Retrieved April 26, 2014.
- Peterson, L. k-Nearest neighbors. Scholarpedia. (2009). Retrieved from ,http://www.scholarpedia. org/article/K-nearest_neighbor.

Base Academic Papers

- Polikar, R. (2006). Ensemble based systems in decision making. IEEE Circuits and Systems Magazine, 2145. National Drought Mitigation Center. Predicting drought. (2013). ,http://drought.unl.edu/ DroughtBasics/PredictingDrought.aspx. Retrieved April 26, 2014.
- Process Software, (2013). Introduction to Bayseian filtering. In: PreciseMail whitepapers (pp. 18). Retrieved from ,www.process.com.
- Quinlan, J. R. (1986). Induction of decision trees. Machine Learning, 1(1), 81106.
- Rish, I. (2001). An empirical study of the naïve Bayes classifier. In: IBM research report.

Base Academic Papers

- Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. Statistics and Computing, 14(3), 199222.
- Tan, P.-N., Michael, S., & Kumar, V. (2005). Classfication and classification: Alternative techniques. In P.-N Tan, S. Michael, & V. Kumar (Eds.), Introduction to data mining (pp. 145315). Boston, MA: Addison-Wesley.