CAS ML
Supervised Learning 2

Dr. Mark Rowan – Rowan Cognitive Data Science Solutions

Learning goals

- Understand difference between regression and classification
- Understand basics of logistic regression and transformation of problem into prediction of class probability
- Learn appropriate loss functions for classification
- Understand classification model assessment criteria and how to apply them
- Understand multiclass classification as an extension of single class classification
- Understand issues of working with imbalanced class data and strategies to resolve
- Gain an overview of non-linear classification algorithms
- Gain an introduction to ensemble methods
- Understand data leakage and strategies to diagnose and avoid it
- Apply learnings in a practical class project

Format: morning lecture, afternoon guided hands-on group work

Recap

Recap

- Gradient descent
- Linear regression
- Imputation
- Data scaling
- Loss functions
- Feature selection
- Bias-variance, how to diagnose over- and under-fitting
- Train/test/validation and cross-validation
- Regularisation
- Model selection

Classification vs regression

Predicting discrete classes vs continuous values

Regression is a technique to predict continuous values such as sales, temperature, or price.

Classification is used to predict discrete classes such as Yes/No, True/False, or Red/Green/Blue.

Many real-world use cases:

- Fraud detection: classify credit card transactions as fraudulent or not.
- Credit Risk Assessment: discretize credit scores to classify customers as low, medium, or high risk.
- Customer Segmentation: discretize purchase amounts to classify customers as low, medium, or high spenders.
- Medical Diagnosis: discretize blood pressure measurements to classify patients as normal, pre-hypertensive, or hypertensive.
- Sentiment analysis: classify customer reviews as positive, negative, or neutral.
- Image recognition: classify objects in images as people, cars, animals, or buildings.

# Discretization of continuous variables

Any regression problem can be converted into a classification problem by discretizing the target variable, e.g. into bins.

- Simplifies the modeling process by reducing the complexity of the data.
- Can improve the accuracy of classification models by reducing the noise and capturing important patterns in the data.
- Makes the model more interpretable and understandable by creating meaningful categories.

Logistic regression

Transforming a linear regression into a classification

- In linear regression, dependent variable y is continuous → we will turn this into a class label!
- Binary response variable usually takes 0 or 1, T or F.
- Logistic regression allows us to predict the *probability* of the output class, given the input features
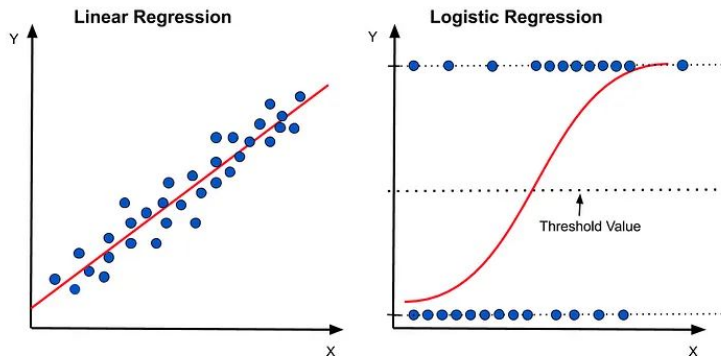
Logistic function

- The logistic function, also known as the sigmoid function, is used to transform the output of a linear equation into a value between 0 and 1.
- It has an S-shaped curve that increases from 0 to 1 as the input value increases from negative to positive infinity.
- Logistic function is used to map the linear combination to a probability value between 0 and 1.

$$f(z) = \frac{1}{1 + e^{-z}}$$

z is the linear regression! i.e.

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n$$
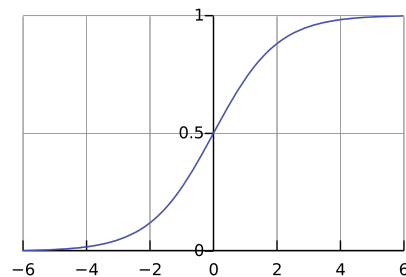
Training a logistic regression model

- Similar to linear regression: find the best values for the weights that minimize the difference between the predicted output and the actual output.
- Needs a different cost function, *cross-entropy*, and an optimization algorithm, gradient descent.
- The cost function measures the difference between the predicted and actual probabilities, and the optimization algorithm adjusts the weights to minimize this difference.
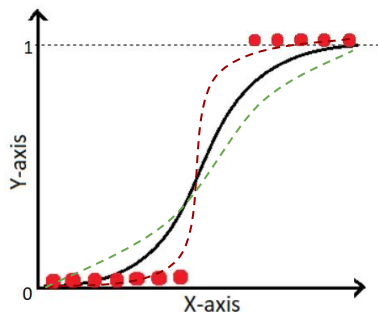
$$L = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} y_{ij} \, log(p_{ij})$$

i = samples/observations
y = sample label (1/0 for binary class)
j = index for classes
p_ij = [0,1] prediction for i sample and class j

Decision boundary (prediction threshold)

- The decision boundary is the line that separates the two classes based on the predicted probabilities.
- The threshold for deciding which class an input belongs to is usually set to 0.5, but can be adjusted based on the needs of the task.
- Inputs with probabilities greater than the threshold are classified as one class, while inputs with probabilities less than the threshold are classified as the other class.
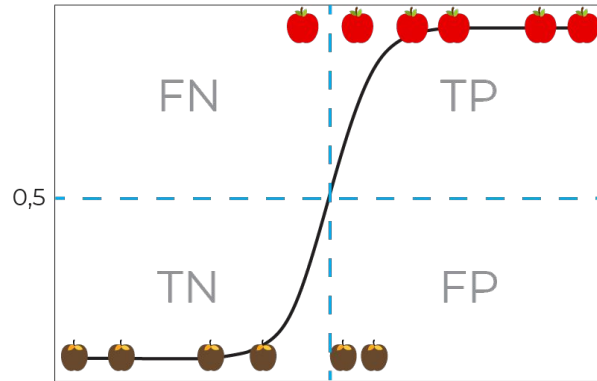

https://www.dataschool.io/roc-curves-and-auc-explained

Further reading

https://machinelearningmind.wordpress.com/2019/11/25/logistic-regression-explained/
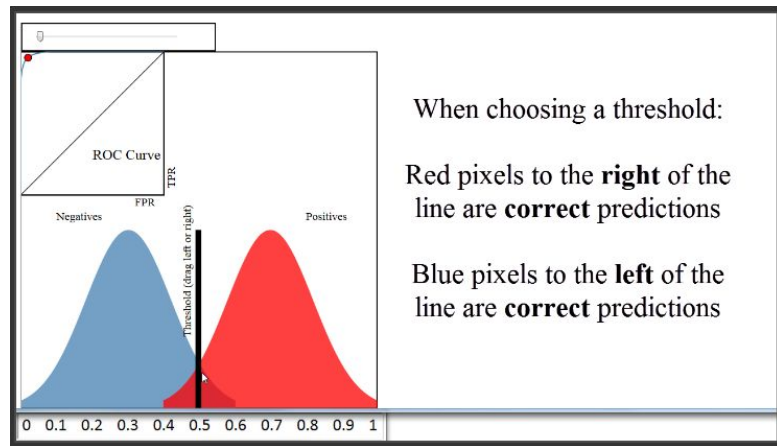
Model assessment

True and false positives and negatives

Confusion matrix

Predictions

Actuals

	36	14
	26	24

Predictions   Actuals

36 ✅

24 ✅

14 ❌

26 ❌

Which are the following?
- TP
- FP
- TN
- FN

Adjusting the decision threshold
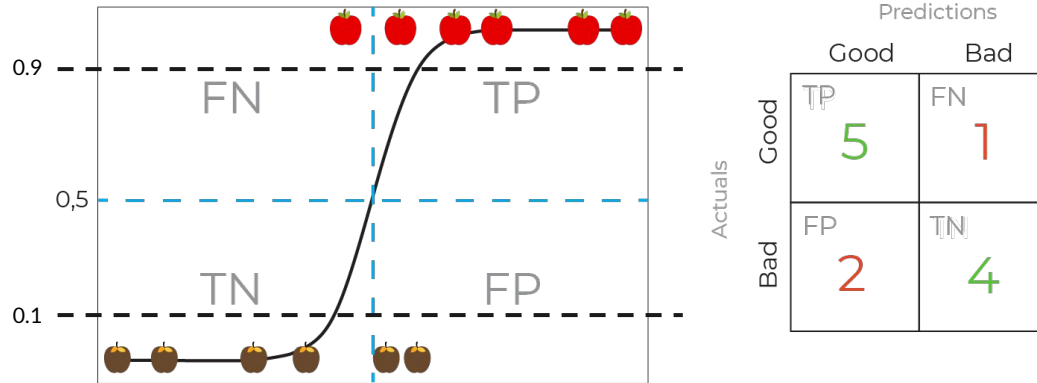
Update the confusion matrix based on the alternative thresholds

https://openclassrooms.com/en/courses/6401081-improve-the-performance-of-a-machine-learning-model/6519011-evaluate-the-performance-of-a-classification-model

Precision and recall

relevant elements

false negatives | true negatives

true positives | false positives

retrieved elements
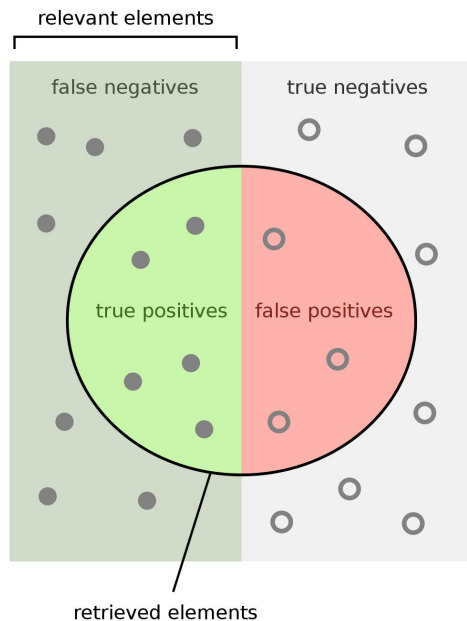
How many retrieved items are relevant?

$$\text{Precision} = \frac{}{}$$

Aka *specificity*

How many relevant items are retrieved?

$$\text{Recall} = \frac{}{}$$

Aka *sensitivity*

Further metrics

What's the deal with Accuracy?

- Accuracy can be misleading if the class distribution is imbalanced.
- Predicting always the majority class can result in good accuracy when there are more majority class instances than minority class instances.
- Precision and Recall together provide better indicators for evaluating model performance.
- F1 score is generally more useful than accuracy, especially if there is an uneven class distribution.

Accuracy examined

Majority class classifier

- TP = ?
- FP = ?
- TN = ?
- FN = ?
- Precision = ?
- Recall = ?
- Accuracy = ?
- F1 = ?

Actual ○○○○○○○ ..99.. ●

Predicted ○○○○○○○ ..99.. ○

Positive class

Solution

- True Positive (TP) = 0
- False Positive (FP) = 0
- True Negative (TN) = 99
- False Negative (FN) = 1

Actual ⚪⚪⚪⚪⚪⚪⚪ ..99.. ⚫

Predicted ⚪⚪⚪⚪⚪⚪⚪ ..99.. ⚪

Precision   = TP / (TP + FP) = 0/0 = 0

Recall   = TP / (TP + FN) = 0/1 = 0

Accuracy   = (TP + TN) / (TP + TN + FP + FN) = (0 + 99) / (0 + 99 + 0 + 1) = 99/100 = 0.99

F1   = (2 x TP) / (2 x TP + FP + FN) = 0 / (0 + 0 + 1) = 0

What if the negative class is the single fraudulent transaction or infected person we want to detect?

ROC and AUC

- Lowering the positive prediction threshold leads to more positive predictions:
  - increasing TP, but at the cost of higher FP
  - i.e. higher recall but for lower precision
- It's a trade-off!
- We can characterise the trade-off between TP and FP using the ROC curve
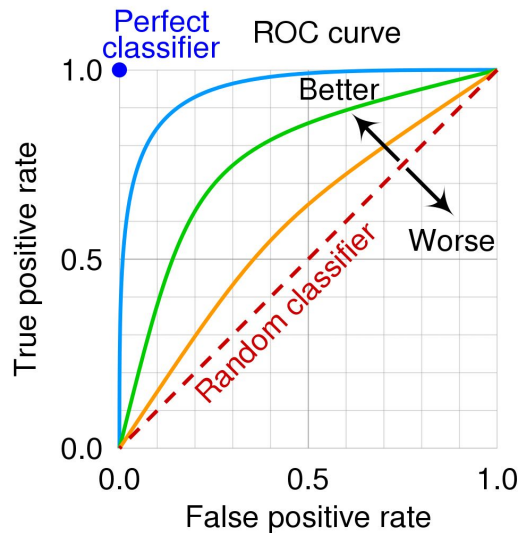
ROC and AUC

- Vary the decision threshold from 0 → 1 (all positive vs no positive) and plot the TP rate vs the FP rate
- "Perfect" classifier gets all TPs with no FPs
- Random classifier is scatter-gun: as many TPs as FPs (assuming balanced classes)
- Choice of decision threshold is a decision based on the needs of the problem (e.g. consider covid tests)
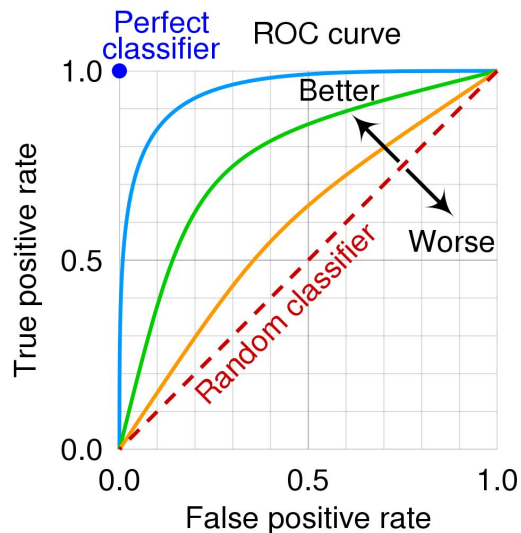
AUC

- AUC is simply "area under (the ROC) curve", i.e. a measure of the "top-leftness" of the ROC curve
- Steeper curves (closer to top-left corner) are better classifiers with higher AUC
- Random classifier gets AUC of 0.5 → **why**?
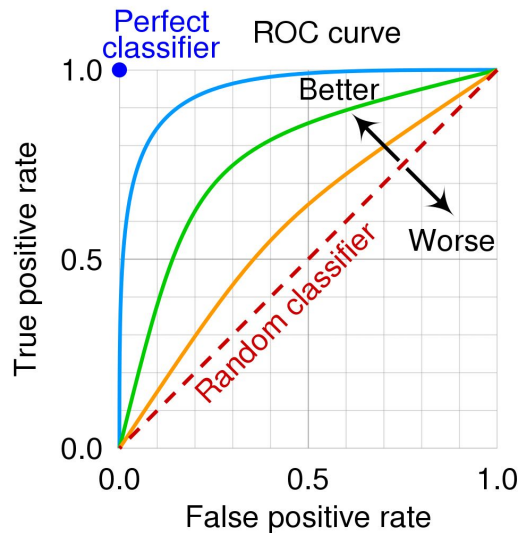
Multi-class classification

One-vs-all classification

- Use one-vs-all classifiers to train a separate binary classifier for each class, treating it as the positive class and all other classes as the negative class.
- During inference, each classifier produces a probability score for its corresponding class, and the class with the highest score is chosen as the prediction.

One-vs-all (one-vs-rest):

Class 1: Green
Class 2: Blue
Class 3: Red
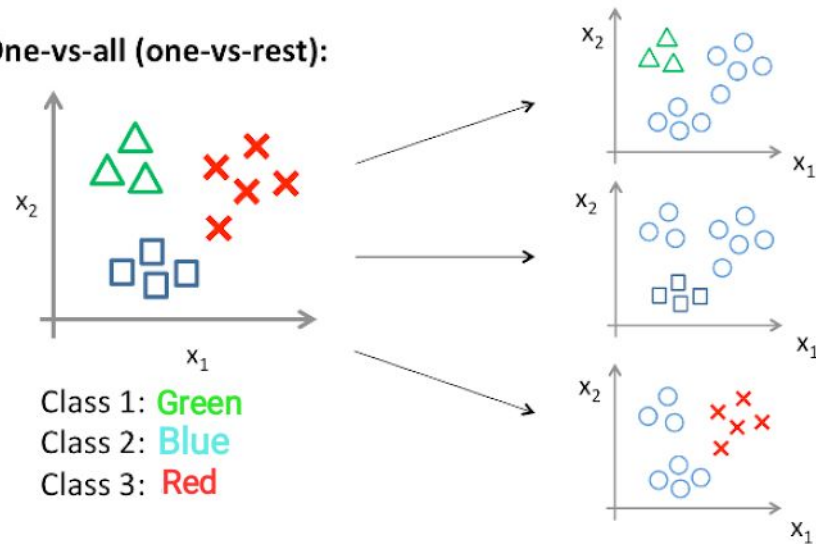
Softmax

- Softmax regression is a generalization of logistic regression that allows for multiple classes.
- It models the probability distribution over all classes using a softmax function, which maps an input vector to a probability distribution that sums to one.
- During training, the model learns a set of weights that maximize the likelihood of the correct class labels given the input data.
- During inference, the model predicts the class with the highest probability score.
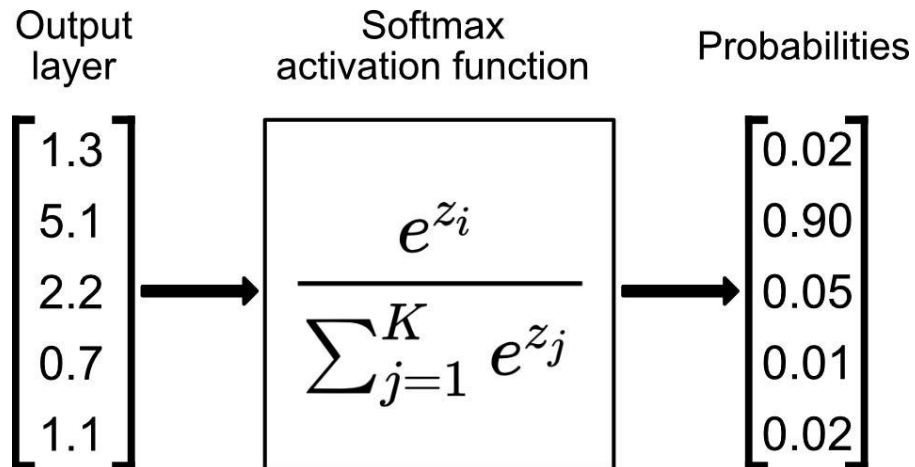
Output layer

$$\begin{bmatrix} 1.3 \\ 5.1 \\ 2.2 \\ 0.7 \\ 1.1 \end{bmatrix}$$

Softmax activation function

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Probabilities

$$\begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$$

Assessment of multi-class models

- For each class in turn, consider the TP, FP, TN, and FN rates
- Build a confusion matrix from the perspective of each class in turn

Example: class: 1

https://www.v7labs.com/blog/confusion-matrix-guide

Assessment of multi-class models

- For each class in turn, consider the TP, FP, TN, and FN rates
- Build a confusion matrix from the perspective of each class in turn

Example: class: 2

https://www.v7labs.com/blog/confusion-matrix-guide
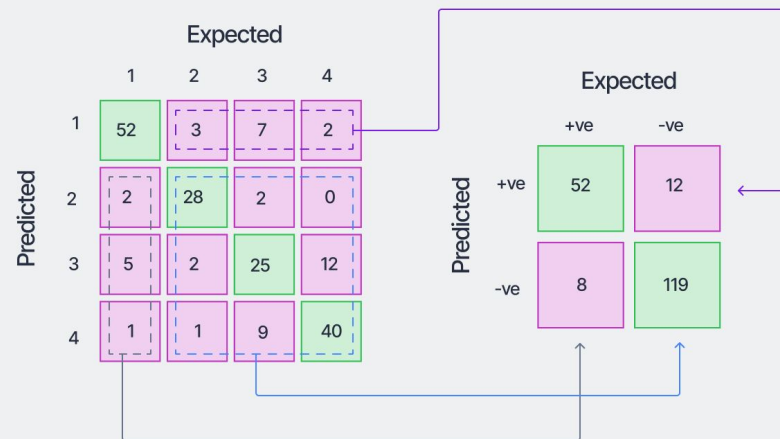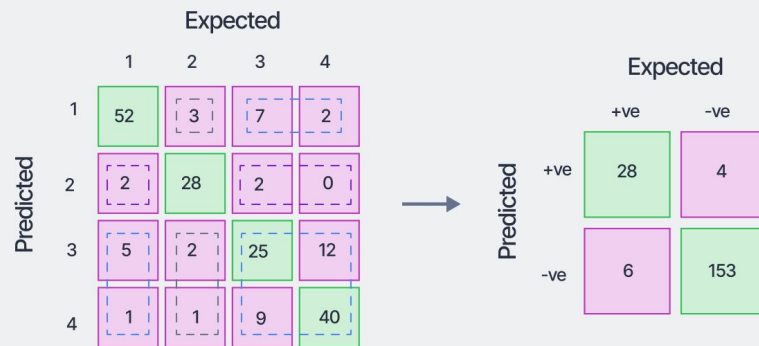
Assessment of multi-class models

Calculate the class-wise precision, recall, F1

How to report the overall performance of the modelling approach?

1. "micro" average precision / recall / F1

2. weighted "macro" average precision / recall / F1

Class	Precision (%)	Recall (%)	F1-Score (%)
1	81.25	86.67	83.87
2	87.50	82.35	84.85
3	56.82	58.14	57.47
4	78.43	74.07	76.19

https://www.v7labs.com/blog/confusion-matrix-guide

V7

Assessment of multi-class models: "micro" measures

Calculate "micro" average precision / recall / F1 directly:

net TP = 52+28+25+40 = 145
net FP = (3+7+2)+(2+2+0)+(5+2+12)+(1+1+9) = 46
net FN = (2+5+1)+(3+2+1)+(7+2+9)+(2+0+12) = 46

Micro precision = net TP/(net TP+net FP) = 145/(145+46) = 75.92%

Micro recall = net TP/(net TP+net FN) = 75.92%

Micro precision = micro recall = micro F1 = accuracy = 75.92%

https://www.v7labs.com/blog/confusion-matrix-guide

V7
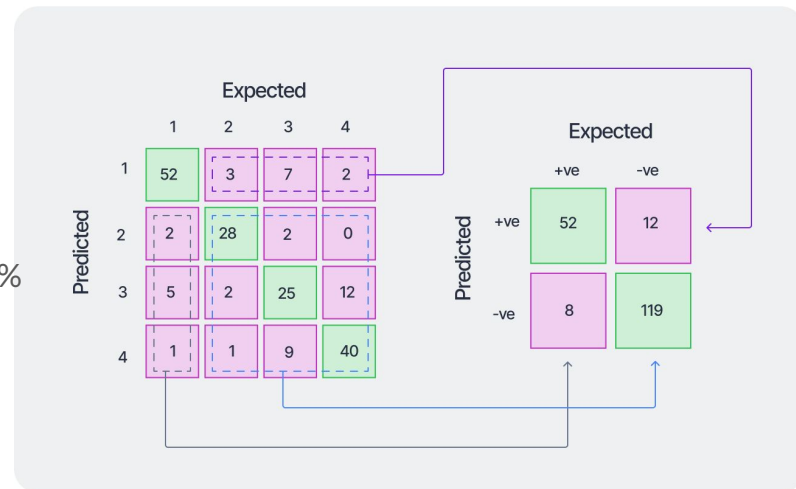
Assessment of multi-class models: "macro" measures

Calculate "macro" average precision / recall / F1 directly from the grid:

- But different classes are different sizes.
- We can take this into account with weighted measures:

$$Weighted\ Precision = \frac{81.25 \times 60 + 87.50 \times 34 + 56.82 \times 43 + 78.43 \times 54}{64 + 32 + 44 + 51}\% = 76.07\%$$

$$Weighted\ Recall = \frac{86.67 \times 60 + 82.35 \times 34 + 58.14 \times 43 + 74.07 \times 54}{64 + 32 + 44 + 51}\% = 75.92\%$$

$$Weighted\ F1 - Score = \frac{83.87 \times 60 + 84.85 \times 34 + 57.47 \times 43 + 76.19 \times 54}{64 + 32 + 44 + 51}\% = 75.93\%$$

Class	Precision (%)	Recall (%)	F1-Score (%)
1	81. 25	86. 67	83. 87
2	87. 50	82. 35	84. 85
3	56. 82	58. 14	57. 47
4	78. 43	74. 07	76. 19
	76.00%	75.31%	75.60%

https://www.v7labs.com/blog/confusion-matrix-guide

v7

Varying decision threshold again
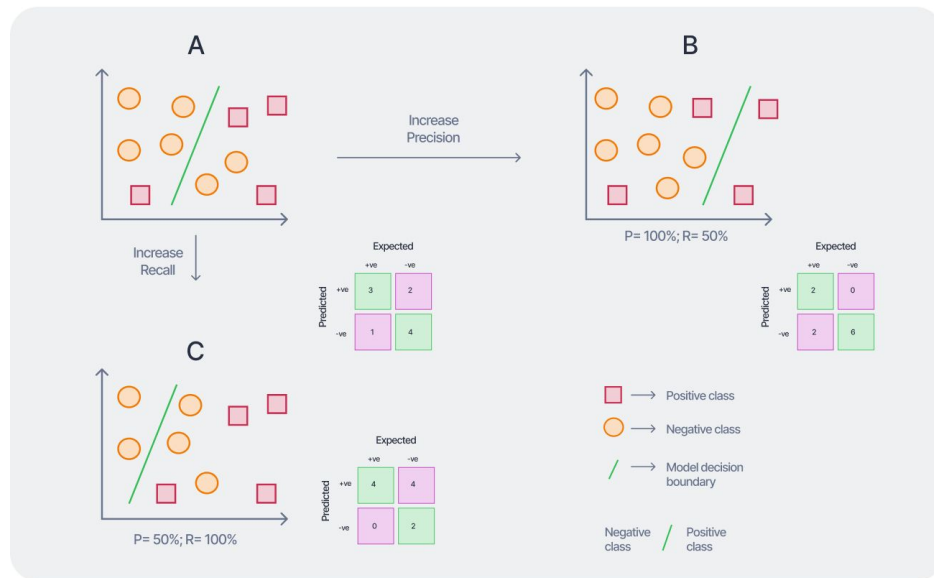
	(A)	(B)	(C)
TPR	75%	50%	100%
FPR	33%	0%	67%

Further reading:
https://www.v7labs.com/blog/confusion-matrix-guide

Working with imbalanced classes

What's the issue?

- Imbalanced classes occur when one class has much fewer instances than another.
- **Common in many real-world datasets, e.g. fraud detection, disease diagnosis, etc.**
- Models may be biased towards the majority class and fail to identify the minority class.
- This can result in false negatives (actual positive instances are incorrectly classified as negative) and false positives (actual negative instances are incorrectly classified as positive).
- Severity of the issue depends on the class imbalance ratio, with larger ratios leading to greater challenges.

Actual ⬭⬭⬭⬭⬭⬭⬭ ..99.. ⬤

Predicted ⬭⬭⬭⬭⬭⬭⬭ ..99.. ⬭

Solutions

- **Downsampling**: This involves reducing the number of instances in the majority class to balance the class distribution.
- **Upsampling**: This involves increasing the number of instances in the minority class to balance the class distribution.
- **Synthetic data generation (interpolation)**: This involves generating synthetic examples of the minority class to increase its representation.
- **Cost-sensitive learning**: This involves assigning different misclassification costs to different classes to reflect the importance of correctly identifying each class.

Downsampling vs upsampling

Downsampling

- Random downsampling: simply sample from the majority class n times, where n is the number of samples in the minority class.
- Can remove important information and lead to over-fitting.
- Stratified downsampling selects instances based on **similarity to instances in the minority class.**
- Can help to preserve important patterns in the data while still reducing the imbalance.
- Various strategies: nearest neighbour sampling, cluster centroid sampling, Tomek links.

Downsampling is a relatively simple approach but may not always be the best choice, particularly when the minority class is very small, and important patterns may be lost.

Upsampling

- Upsampling means increasing the number of minority class instances to balance the class distribution.
- Either by duplicating existing instances or by generating synthetic instances of the minority class.
- In random upsampling, instances from the minority class are randomly duplicated until the class distribution is balanced ("sampling with replacement").
- However, can lead to over-fitting and may not capture important patterns in the data.
- An alternative approach is synthetic upsampling, where synthetic instances of the minority class are generated using techniques such as SMOTE (Synthetic Minority Over-sampling Technique).

Synthetic upsampling can help preserve important patterns in the data while still increasing the representation of the minority class.

Interpolation with SMOTE

- SMOTE (Synthetic Minority Over-sampling TEchnique)
    - Find the k-nearest neighbours of a minority point x
    - Choose one neighbour at random x'. Create new sample **s = x + (x' − x) × N(0,1)**
    - In Python: **pip install imblearn**

https://iq.opengenus.org/smote-for-imbalanced-dataset/
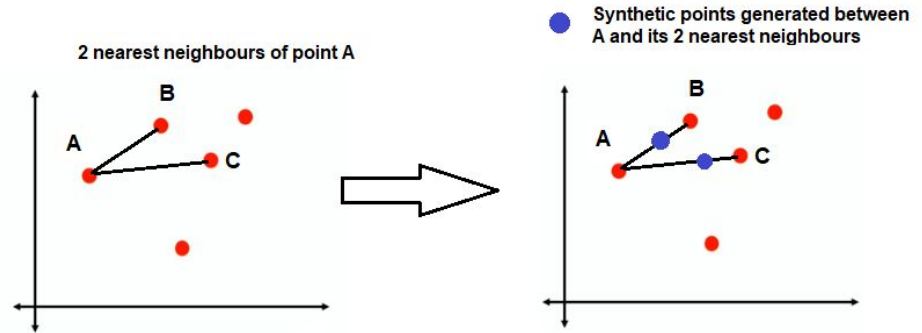
Model assessment in imbalanced class conditions

- Important to carefully evaluate the performance of the model on both the minority and majority classes to ensure that the model is not biased towards one or the other.
- Evaluation metrics such as precision, recall, F1-score, and area under the ROC curve are often more appropriate than simple accuracy in imbalanced classification problems.

Non-linear classification models

Limitations of linear methods

- Linear methods are suitable for modeling relationships between variables when there is a linear relationship between the outcome variable and the predictors.
- However, in many cases, the relationship between the outcome and predictors is not linear, but rather has a more complicated, non-linear shape.
- In cases like this, a linear regression model would be unable to accurately capture the relationship between variables.
- Instead, we would need a more flexible, non-linear model that can capture the non-linear relationship between variables.
- Applies to logistic regression too! (It's still a linear model).
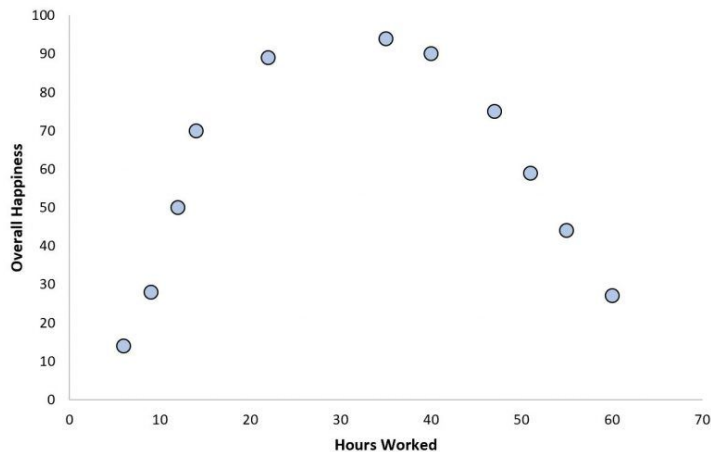
k-nearest neighbour

- k-NN finds the k-nearest neighbors of a new data point in the training set using a distance metric, and predicts the class or value based on the majority or average of the neighbors.
- Choice of k affects the trade-off between variance and bias.
- k-NN is simple and powerful, and useful when there is no clear functional relationship between predictors / outcome variable.
- But it can be computationally expensive for large datasets.
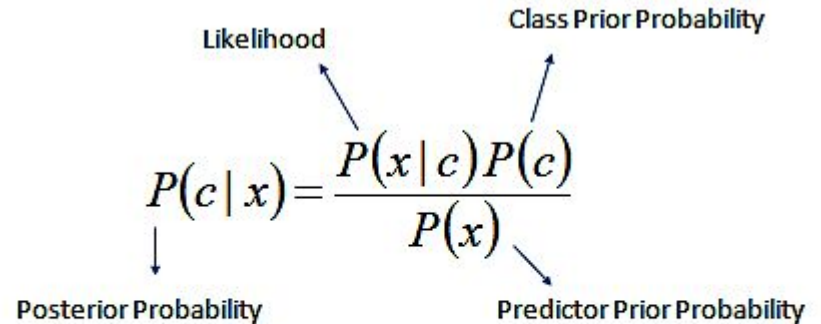
What happens when k = 1 ? What happens when k = N ?
How could we use this for regression?

https://medium.com/analytics-vidhya/k-nearest-neighbor-the-maths-behind-it-how-it-works-and-an-example-f1de1208546c
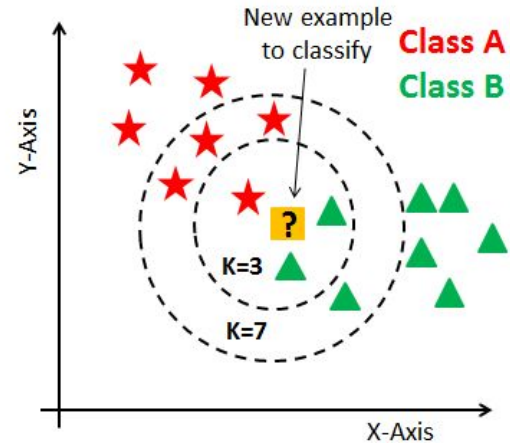
Naive Bayes

- Naive Bayes works by calculating the probability of a new data point belonging to each class based on its feature values.
- Using Bayes' theorem for calculating conditional probabilities.
- Naive Bayes assumes that the features are independent of each other, which means that the value of one feature does not affect the value of another feature.

Likelihood

Class Prior Probability

$$P(c \mid x) = \frac{P(x \mid c) P(c)}{P(x)}$$

Posterior Probability

Predictor Prior Probability

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Naive Bayes example

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

What is the probability that we play if the weather is sunny today?

P(Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

P(Sunny | Yes) = 3/9 = 0.33
P(Sunny) = 5/14 = 0.36
P(Yes)= 9/14 = 0.64   → P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60

Neural networks

- Modelled after the human brain, made up of interconnected layers of nodes, or artificial neurons, that process and transmit information.
- The nodes in a neural network are organized into layers, including an input layer, one or more hidden layers, and an output layer.
- Each node receives inputs from the previous layer, applies a function to the inputs, and then passes the result (1 or 0) to the next layer.
- Weights between layers are adjusted through a process called backpropagation in order to minimize error between predicted and desired outputs.
- Neural networks are capable of learning complex patterns and can be used for image recognition, speech processing, even game playing.
- Neural networks with very many hidden layers are "deep neural networks".
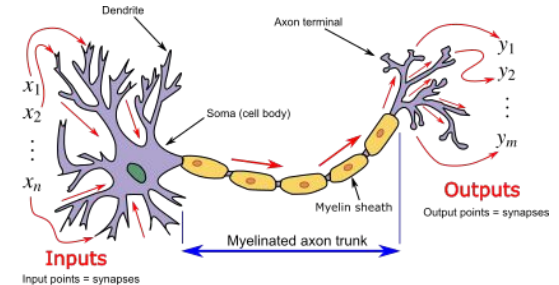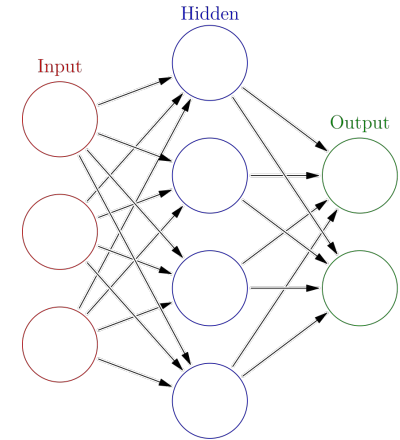
Support vector machine

- SVM is based on finding the best hyperplane that separates data into different classes, with the widest possible margin between them.
- The hyperplane is selected by maximizing the distance between the hyperplane and the closest data points of each class. These data points are called support vectors.
- SVM is especially effective in handling high-dimensional data where the number of features is much greater than the number of observations.

Introduction to SVM
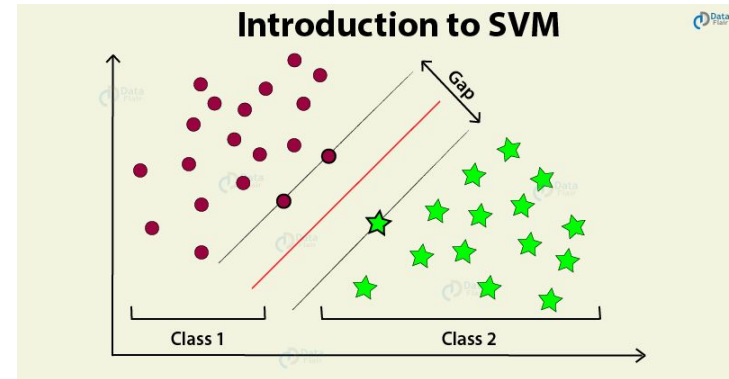
Support vector machine: kernels

- Kernels are a mathematical technique used in support vector machines.
- They help transform the input data into a higher dimensional space where it is easier to separate the classes.
- Kernels map the original features to a new feature space, where the transformed data may be linearly separable.
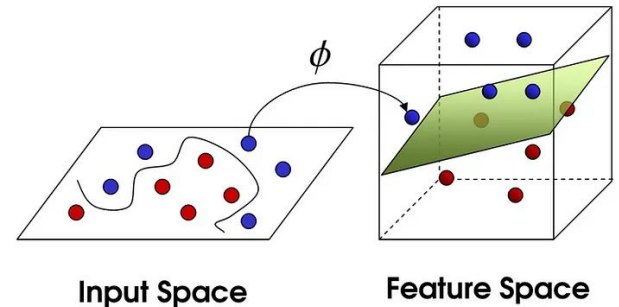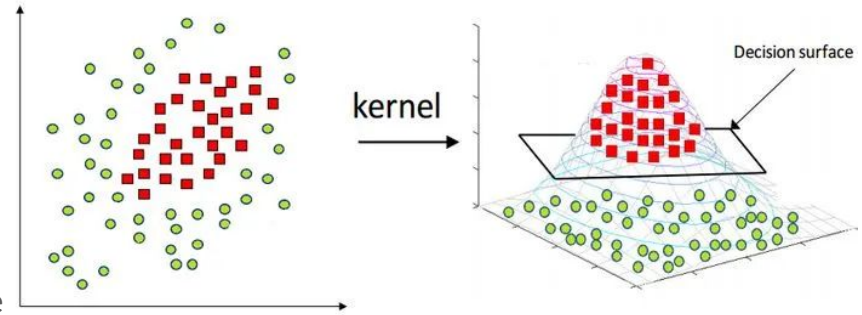
Decision tree

- Decision tree classifiers build tree-like models for predicting class of new instances.
- The tree structure is built by recursively splitting the dataset based on the values of input features until each subset contains only instances of the same class ("purity"), or until the maximum depth of the tree is reached.
- At each node of the tree, a decision is made based on the value of a selected feature, using a criterion such as information gain or Gini impurity to determine the best split.
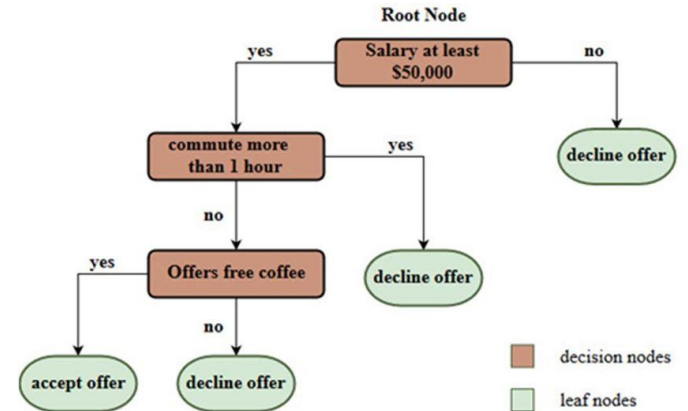
1. Find the variable split that best separates data, and divide the tree
2. Look for the next best split within each group using remaining features
3. Repeat until all leaf nodes are "pure" (single class), or maximum depth reached

Decision tree: inference

- Once the tree is built, new instances can be classified by following the branches of the tree from the root to a leaf node, where the predicted class label is based on the majority class of the training instances that reach that node.
- Decision trees can handle categorical and continuous input features, and can be used for regression by predicting a continuous output variable instead of a class label.
- Decision trees can suffer from overfitting if the tree is too complex, which can be mitigated by setting a maximum depth or pruning the tree after it is built.
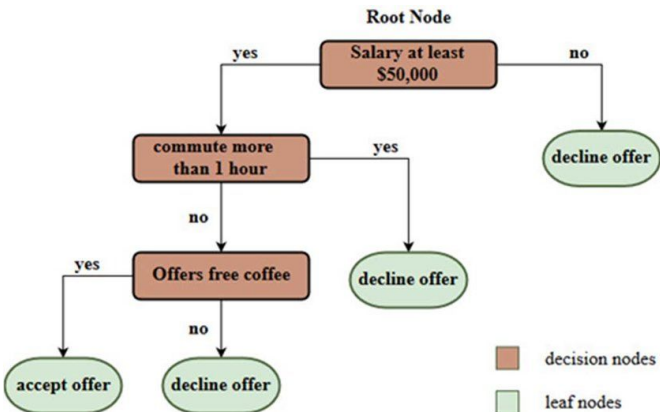
Ensemble methods

Weak classifiers

- Individual classifiers are susceptible to making incorrect predictions.
- But as a group, they may on average be correct.

What if we combine multiple classifiers, each specialising slightly differently?

→ prediction errors will not always be the same; variance of the combination is lower.

https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting
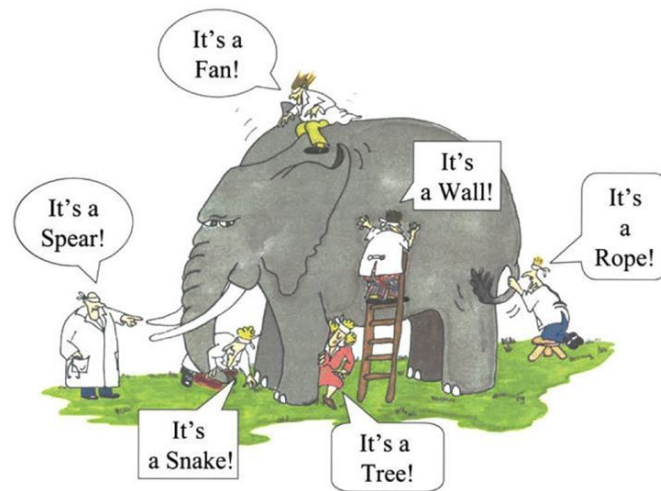
Ensemble methods

- Ensemble methods are a family of machine learning techniques that combine multiple models to improve predictive performance.
- Ensemble methods can be used for both classification and regression tasks.
- There are two main types of ensemble methods: bagging and boosting.

https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting
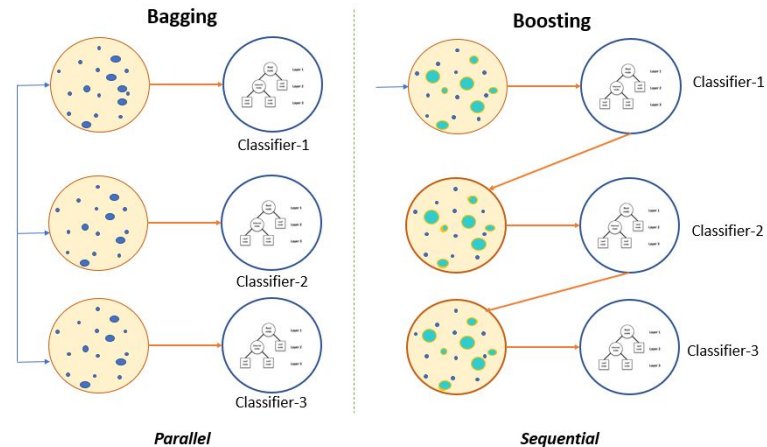
Bagging: random forest

- Bagging stands for **Bootstrap Aggregating,** where multiple models are trained on different subsets of the data.
- Random Forest is a popular bagging algorithm that combines decision trees with bootstrapping and random feature selection.
- In Random Forest, each tree is trained on a random subset of the data and a random subset of the features.
- The final prediction is made by aggregating the predictions of all the trees, either by majority vote for classification or by averaging for regression.

BAGGING LEARNING PROCEDURE

Original Data

Bootstrapping

Aggregating

Ensemble classifier

Bagging

$$f(x) = \frac{1}{B} \sum_{b=1}^{B} f_b(x)$$

Weak Learners

Generates Bootstrapping Sets
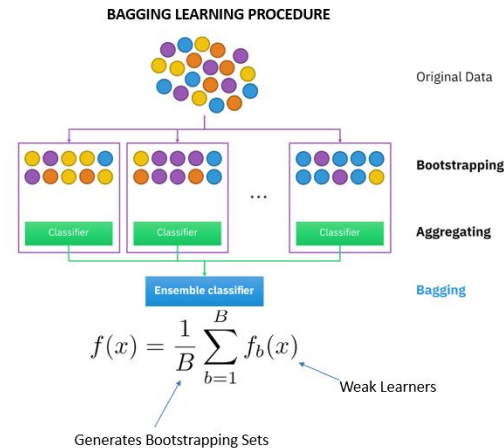
Boosting: gradient boosting

- Multiple models are trained sequentially, with each new model correcting the errors of the previous ones.
- **Gradient Boosting** is a popular algorithm combining decision trees with gradient descent optimization.
- In Gradient Boosting, each tree is trained on the residuals of the previous tree, i.e., the difference between the predicted and actual values.
- Final prediction is made by summing the predictions of all the trees, each weighted by a learning rate.

BOOSTING LEARNING PROCEDURE

Original Data — Weighted data — Weighted data

Classifer — Classifer — Classifer — Ensemble Classifer

Strong Learner   Weak Learners

$$f(x) = \sum_t \alpha_t h_t(x)$$

Weight calculated by considering the last iteration's error
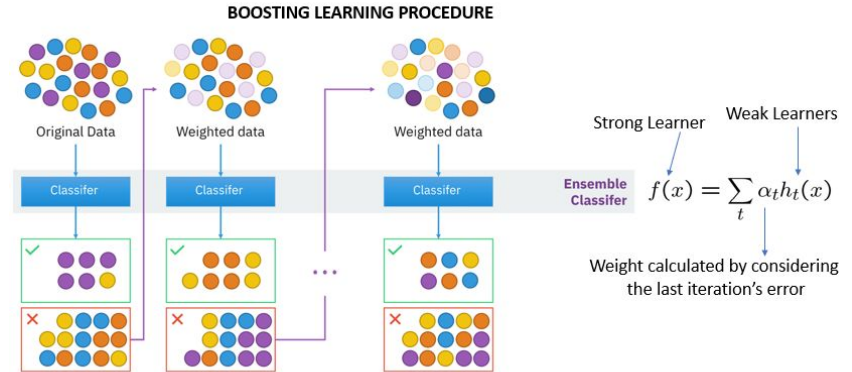
Pros and cons of bagging and boosting

- Bagging and Boosting can both improve the performance of a model and reduce overfitting.
- Bagging is relatively easy to parallelize and can handle high-dimensional data, but may not improve the accuracy if the base models are too similar.
- Boosting can achieve better performance than bagging by focusing on the difficult examples, but can be sensitive to noisy data and may be prone to overfitting if the learning rate is too high.

Data leakage

What is data leakage?

- Data leakage refers to a situation where information from the test set is used to train the model, leading to overly optimistic performance estimates.
- Can occur in various ways, such as using future information, using derived features, or using target leakage.
- Can lead to overfitting, poor generalization, and invalid conclusions.
- To avoid data leakage, we should only use variables that are available at the time of prediction, and we should avoid using the target variable or any variables that contain information that is not available in the testing phase.

Example

My team trained a customer churn model in a telecoms company.

- Attempt 1: 100% predictive power
  - Features of customer interactions with the call center was to blame – why?
- Attempt 2: better, but we noticed that account_number was a predictor
  - What might be happening here?

Diagnosing data leakage

- Diagnosing data leakage involves inspecting the data and the model to identify any sources of information leakage.
- One way to diagnose data leakage is to compare the performance of the model on the training and validation sets.
- If the performance on the validation set is much worse than on the training set, it may indicate data leakage.
- Another way to diagnose data leakage is to examine the features used by the model and check if any of them contain information from the test set.
- Resolving data leakage requires removing any sources of information that should not be used for training the model.

Hands-on: Group challenge

Group challenge: Titanic

Task: Build a model to predict whether a passenger survived the Titanic sinking. Use all you have learned.

Dataset: https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv
In Colab: !wget <url>

1. **Data Exploration**:   Look at the structure, features, and distributions of the dataset.
2. **Data Cleaning**:   Remove any missing values or outliers and transform categorical variables.
3. **Feature Engineering**:   Create new features that may help with the prediction task.
4. **Imputation**:   Fill in missing values using appropriate techniques.
5. **Feature Selection**:   Choose the most relevant features for the prediction task and remove irrelevant ones.
6. **Imbalanced Classes**:   Use oversampling or undersampling techniques to balance the classes.
7. **Cross-Validation**:   Split data into training/testing sets, evaluate model performance using cross-validation.
8. **Model Evaluation**:   Evaluate the models using appropriate evaluation metrics.
9. **Data Leakage**:   Check for data leakage and adjust models as necessary.
10. **Regularization**:   Evaluate the impact of regularization on model performance.
11. **Hyperparam. Search**:   Tune hyperparameters to optimize model performance.
12. **Model Selection**:   Compare model performance on your held-out testing set (*once only*!) to find best model.

Titanic variable names

- Survived: 0 = Dead, 1 = Survived
- Pclass: Ticket class with 1 = 1st class, 2 = 2nd class, 3 = 3rd class
- Sex: Gender of male or female
- Age: Age in years
- SibSp: Number of siblings/spouses aboard the Titanic
- Parch: Number of parents/children aboard the Titanic
- Ticket: Ticket number
- Fare: Passenger fare
- Cabin: Cabin number
- Embarked: Port of Embarkation with C = Cherbourg, Q = Queenstown, S = Southampton

Example

Wait until the end of the class, we will go through it together!

https://colab.research.google.com/drive/1t9M_FKeaGrEgSzjXS264pA5Bkxbc-oN?usp=sharing

Other solutions:

1. https://towardsdatascience.com/comprehensive-beginners-guide-to-kaggle-titanic-survival-prediction-competition-solution-21c5be2cec2c
2. Code for 1): https://github.com/chongjason914/kaggle-titanic/blob/main/titanic-tutorial.ipynb
3. https://towardsdatascience.com/machine-learning-with-the-titanic-dataset-7f6909e58280