MACHINE LEARNING AND OPTIMIZATION FOR TESTING

DI Dr Branka Stojanovic

ORGANISATION

Unit	Date	Type/Scope	Content
1	06.10.2023	2 LE	Introduction – AI/ML and testing
2	13.10.2023	2 EXE	Introduction – data science and Python
3	20.10.2023	2 LE	Machine learning, testing and data preparation
4	20.10.2023	2 EXE	Data preparation and Python; Homework assignments
5	10.11.2023	2 LE	Supervised Machine Learning
6	10.11.2023	2 EXE	Supervised Machine Learning
7	17.11.2023	2 LE	Unsupervised Machine Learning
8	17.11.2023	2 EXE	Unsupervised Machine Learning
9	01.12.2023	2 LE	Neural Networks and Deep Learning
10	01.12.2023	2 EXE	Neural Networks and Deep Learning
11	15.12.2023	2 LE	Final project introduction and assignments
12	12.01.2024	2 EXE	Hands-on - Final project consultation; Homework discussion
13	19.01.2024	2 LE	Final project tutorial and results presentations and discussion
14	19.01.2024	2 EXE	Final project results demonstrations
15	26.01.2024	2 LE	Recap and Q&A
16	09.02.2024	1 EXM	Exam

3. SUPERVISED MACHINE LEARNING

OUTLINE

3.1 Supervised Machine Learning

3.2 Regression

3.3 Classification

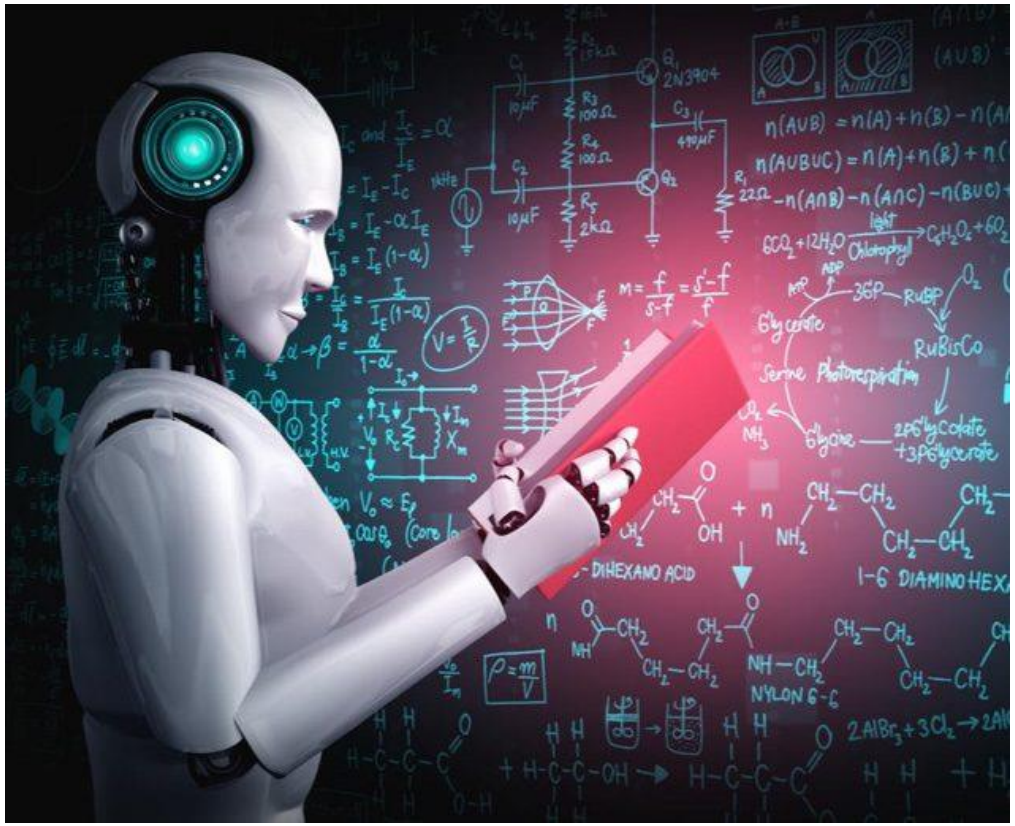


Image source: https://www.eweek.com/enterprise-apps/what-is-artificial-intelligence

OUTLINE

3.1 Supervised Machine Learning
- Overview
- Advantages/ disadvantages
- Types

3.2 Regression
- Linear regression
- Model optimization
- How to check performance?

3.3 Classification
- Logistic regression
- KNN
- Model optimization
- How to check performance?

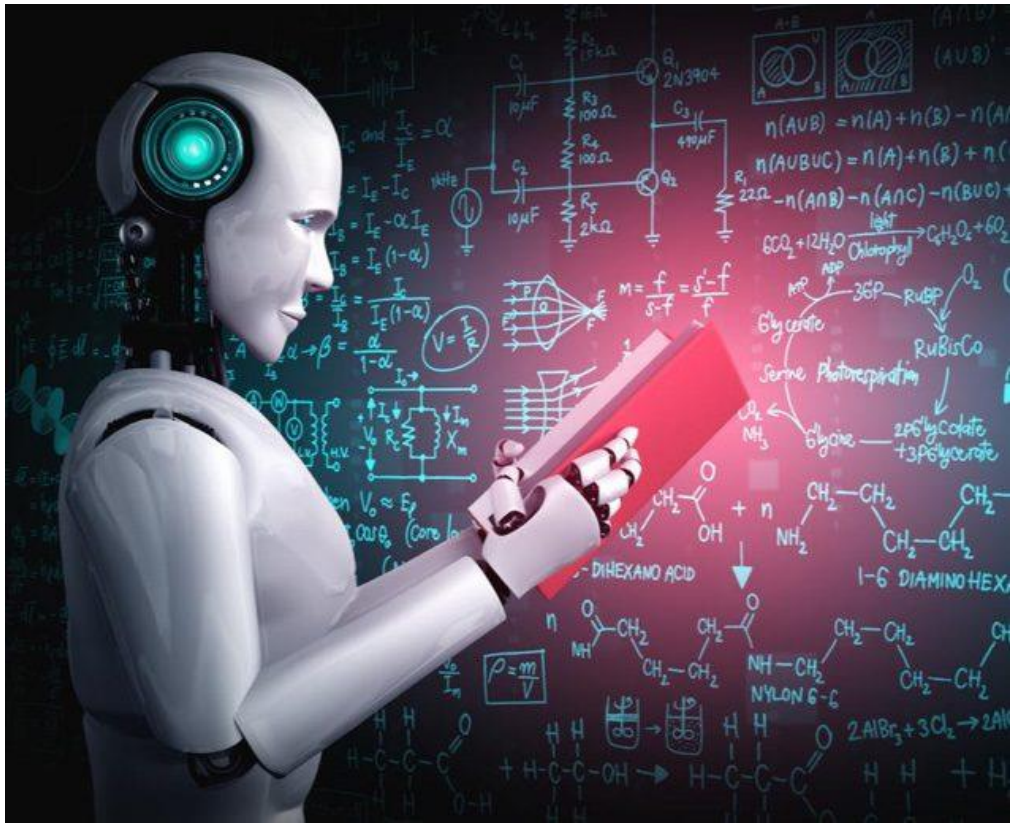


Image source: https://www.eweek.com/enterprise-apps/what-is-artificial-intelligence

3.1 SUPERVISED MACHINE LEARNING

- In Supervised Learning, the machine learns under **supervision**
- ML model is able to **predict** with the help of a labeled dataset
- A **labeled dataset** is one where you already know the **target** answer

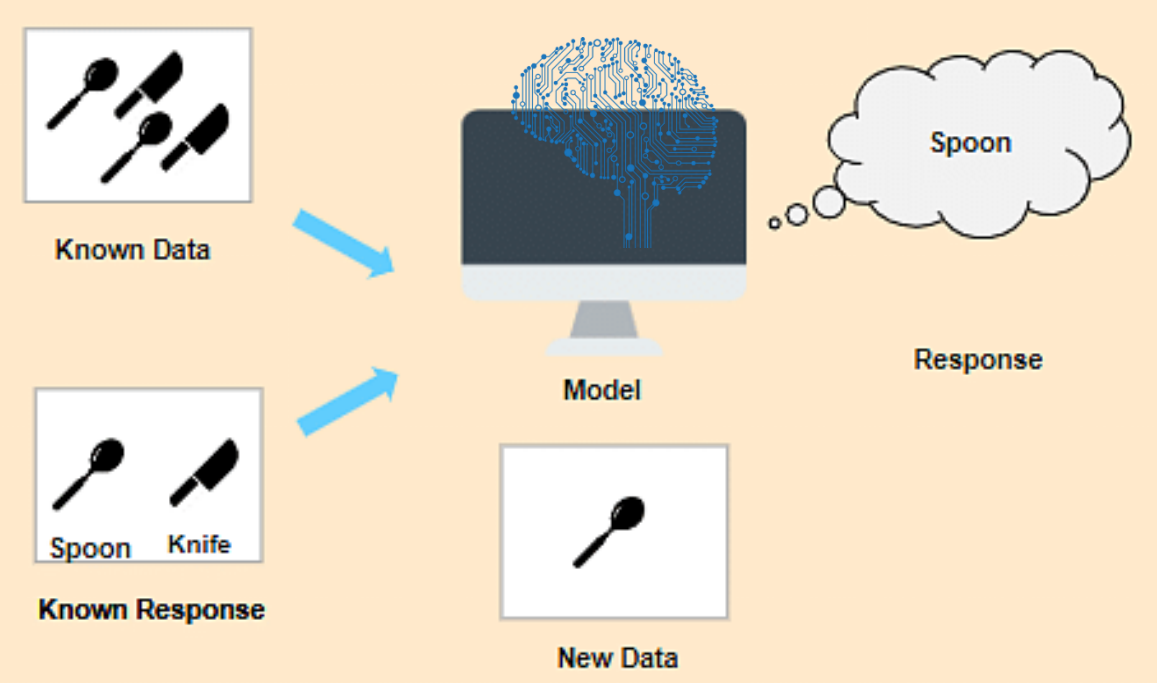


Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

Overview

- Supervised learning models are trained using **labeled data**, also known as **training data**, to **predict** results

- Consider we have a dataset with data on both cats and dogs

- The model's main function is to **recognize the new input** data when evaluated using a new input data set that was not used for the subsequent training

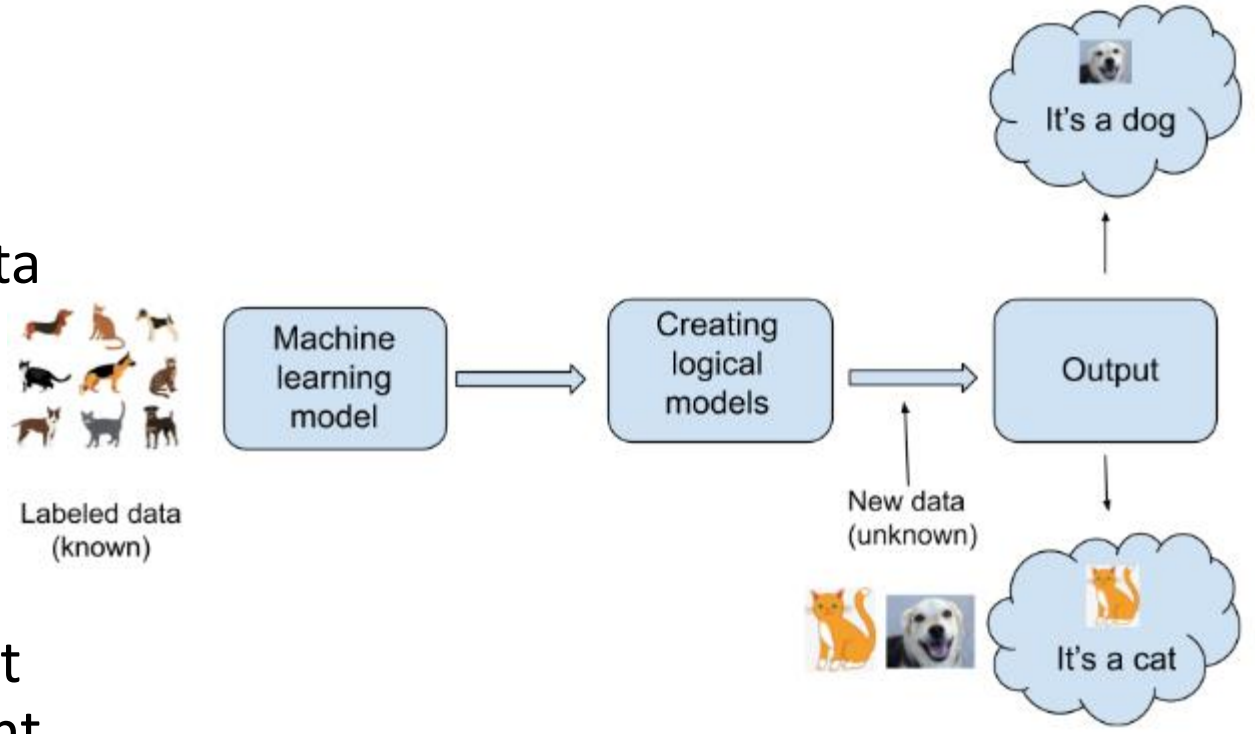


Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

Advantages

- Supervised learning resolves **various computation issues** encountered in the **real world**, including spam detection, object and image identification, etc.

- Supervised learning uses **past experience** to optimize performance and predict the outputs

- The **training data** can be **reused** unless there is any feature change

Disadvantages

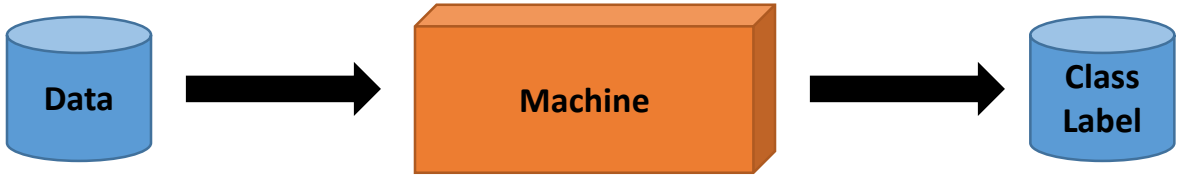
- **Computation time**, or running time, is huge for supervised learning
- Supervised learning models frequently need **updates**
- **Pre-processing** of data is a big challenge for predicting the output
- It is easy to **overfit** supervised algorithms
  - It happens when a statistical model matches its training data

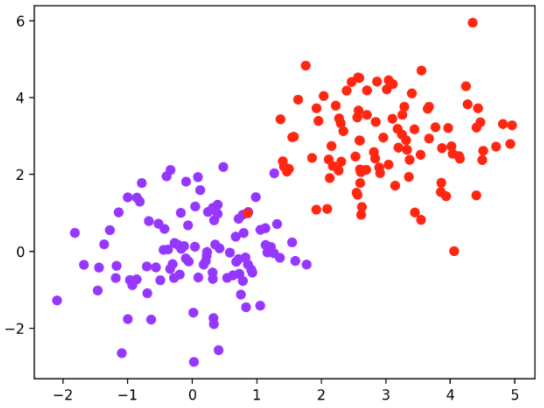
Real-Life Applications

- **Risk Assessment**
  - Supervised learning is used to assess the risk in financial services or insurance domains in order to minimize the risk portfolio of the companies
- **Image Classification**
  - Image classification is one of the key use cases of demonstrating supervised machine learning
  - E.g., Facebook can recognize your friend in a picture from an album of tagged photos
- **Fraud Detection**
  - To identify whether the transactions made by the user are authentic or not
- **Visual Recognition**
  - The ability of a machine learning model to identify objects, places, people, actions, and images

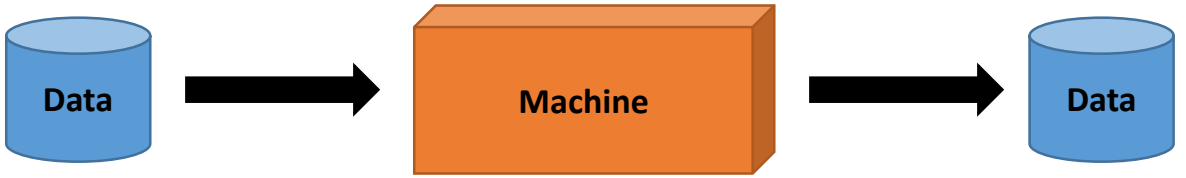
Types

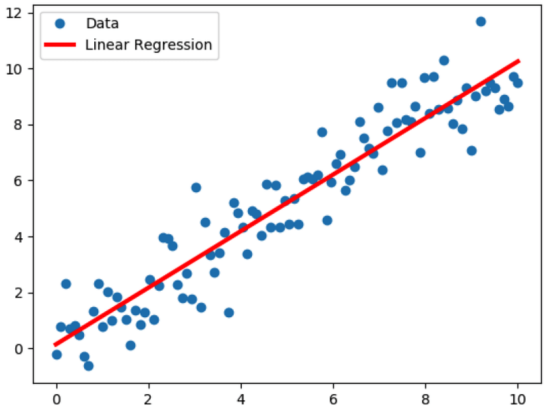
- **Classification**





- **Regression**





OUTLINE

3.1 Supervised Machine Learning

- Overview
- Advantages/ disadvantages
- Types

3.2 Regression

- Linear regression
- Model optimization
- How to check performance?

3.3 Classification

- Logistic regression
- KNN
- Model optimization
- How to check performance?

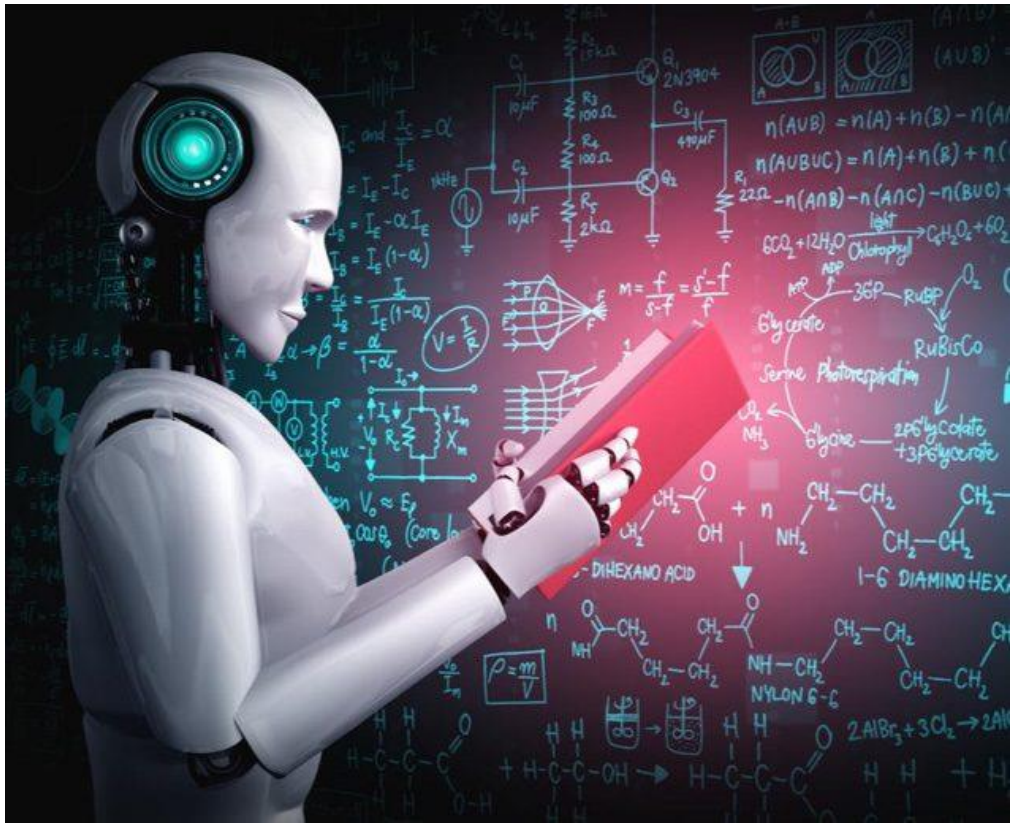


Image source: https://www.eweek.com/enterprise-apps/what-is-artificial-intelligence

3.2 REGRESSION

- A regression algorithm is used to figure out the connection between dependent and independent variables
  - Dependent variables are responsible for predicting and forecasting
  - Independent variables are those which affect the analysis
- It is used to make projections
- Example: consider we have variable one as humidity and variable two as the temperature, where the temperature will be the independent variable and humidity will act as the dependent variable
  - Humidity and temperature are correlated as temperature increases, humidity decreases, and vice versa

Temperature

Humidity

Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

Overview

- Regression is a method of modelling a target value based on independent predictors
- This method is mostly used for forecasting and finding out cause and effect relationship between variables
- Example: stock prediction - breaking down past information on stock costs and trends to recognize patterns
- Some popular regression algorithms in supervised learning:
  - **Linear Regression**
  - Regression Trees
  - Non-Linear Regression
  - Bayesian Linear Regression

Linear regression

$$y = (a_1 * x) + a_0$$

- Simple linear regression is a type of regression analysis where the number of independent variables is one and there is a linear relationship between the **independent** (**x**) and **dependent** (**y**) variable
- The red line in the above graph is referred to as **the best fit straight line**
- The motive of the linear regression algorithm is to find the best values for $\mathbf{a_0}$ and $\mathbf{a_1}$

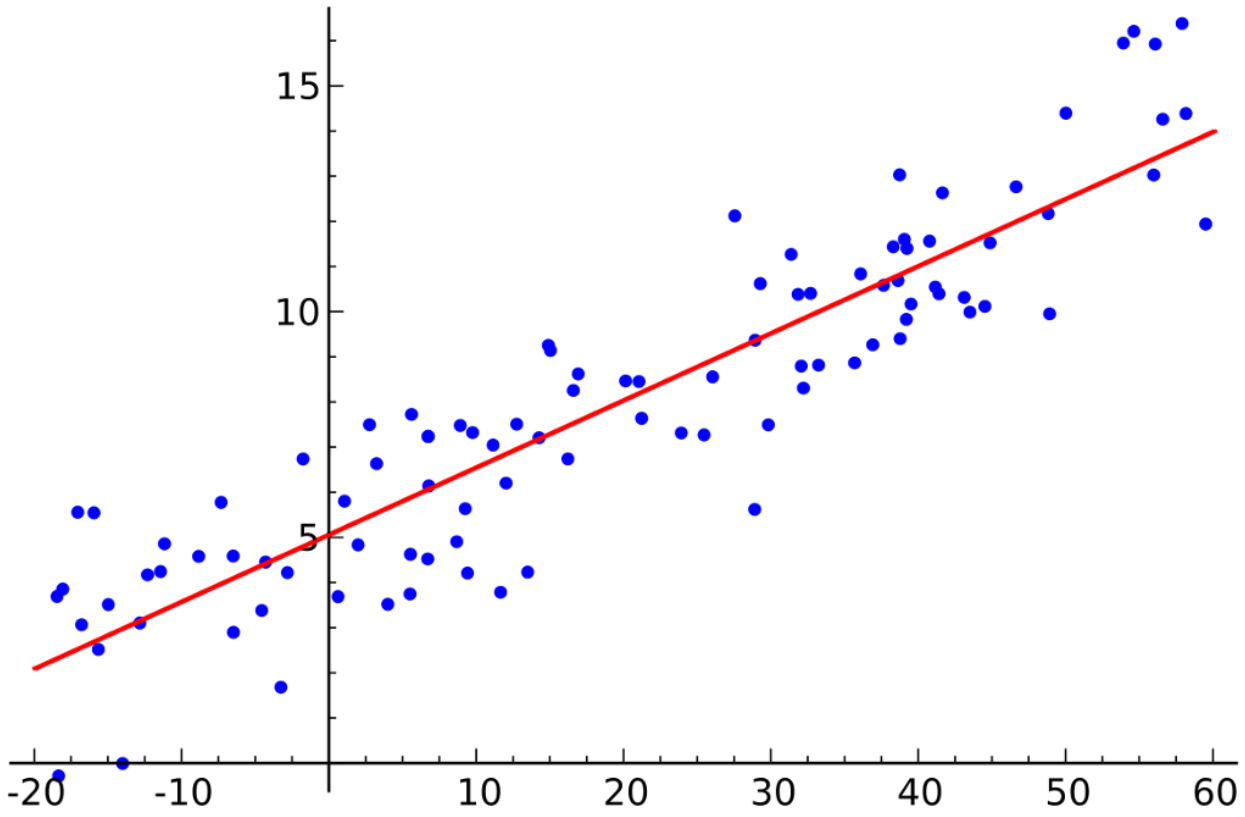


Image source: https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a

Model optimisation – Cost function

- The **cost function** helps us to figure out **the best possible values** for $\mathbf{a_0}$ and $\mathbf{a_1}$ which would provide **the best fit line** for the data points
  - → we convert this search problem into a **minimization problem** where we would like to minimize the error between the predicted value and the actual value

$$minimize \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

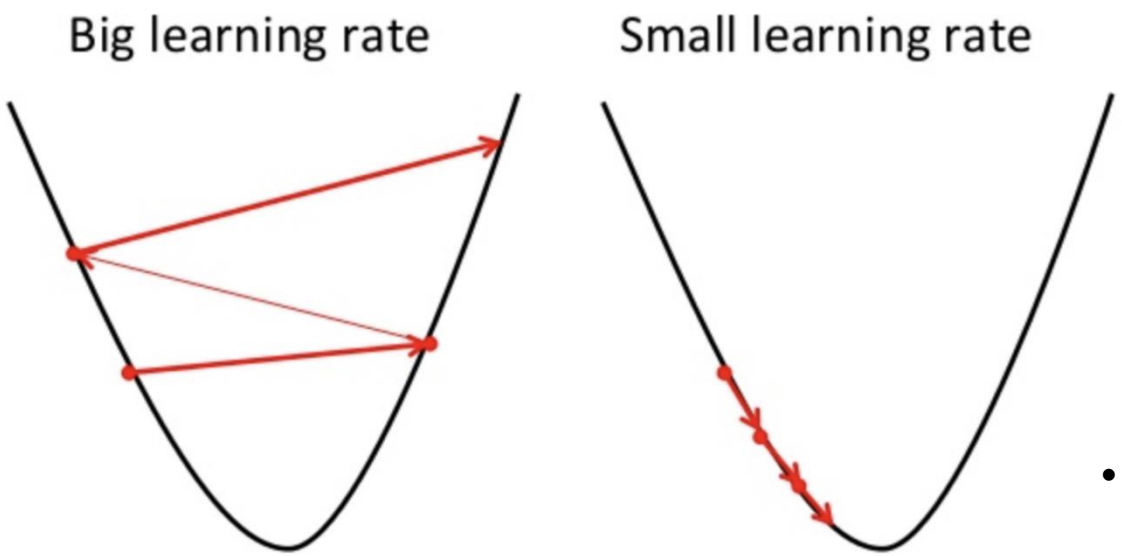
$$J = \frac{1}{n} \sum_{i=1}^{n} (pred_i - y_i)^2$$

- The difference between the predicted values and ground truth measures the error difference
- We square the error difference and sum over all data points and divide that value by the total number of data points
- This provides the average squared error over all the data points → therefore, this cost function is also known as the **Mean Squared Error (MSE)** function
- Now, using this MSE function we are going to change the values of $\mathbf{a_0}$ and $\mathbf{a_1}$ such that the MSE value settles at the minima

Image source: https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a

Model optimisation – Gradient Descent

- Gradient descent is a method of updating $\mathbf{a_0}$ and $\mathbf{a_1}$ to reduce the cost function (MSE). The idea is that we start with some values for $\mathbf{a_0}$ and $\mathbf{a_1}$ and then we change these values iteratively to reduce the cost. Gradient descent helps us on how to change the values.



- *To draw an analogy, imagine a pit in the shape of U and you are standing at the topmost point in the pit and your objective is to reach the bottom of the pit. There is a catch, you can only take a discrete number of steps to reach the bottom. If you decide to take one step at a time you would eventually reach the bottom of the pit, but this would take a longer time. If you choose to take longer steps each time, you would reach sooner but, there is a chance that you could overshoot the bottom of the pit and not exactly at the bottom.*
- In the gradient descent algorithm, the number of steps you take is the **learning rate**
- This decides on **how fast the algorithm converges to the minima**

Image source: https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a

Model optimisation – Gradient Descent

- To update $\mathbf{a_0}$ and $\mathbf{a_1}$, we take gradients from the cost function
- To find these gradients, we take partial derivatives with respect to $\mathbf{a_0}$ and $\mathbf{a_1}$

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \implies \frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \cdot x_i \implies \frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

$$a_0 = a_0 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$a_1 = a_1 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

Model optimisation – Gradient Descent

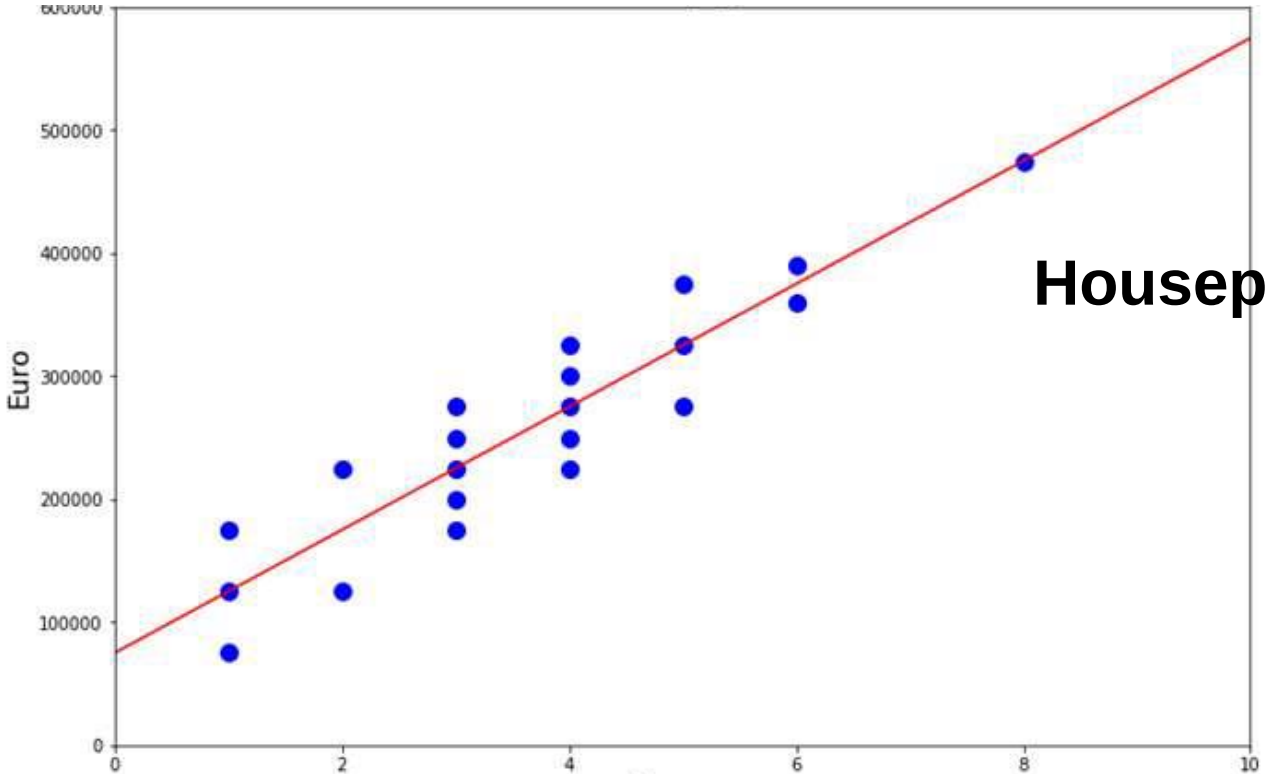
- The partial derivates are the gradients and they are used to update the values of $\mathbf{a_0}$ and $\mathbf{a_1}$

- **Alpha** is the **learning rate** which is a **hyperparameter** that you must specify

- A smaller learning rate could get you closer to the minima but takes more time to reach the minima, a larger learning rate converges sooner but there is a chance that you could overshoot the minima

Our house price example

- Calculate the formula for this line:

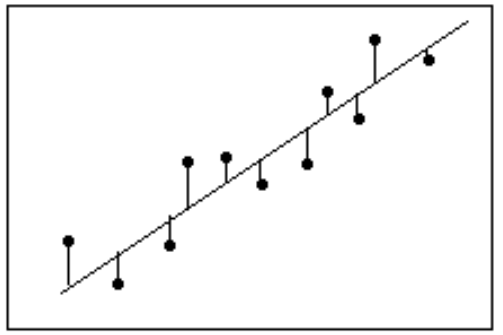


Houseprice = (50.000 * Room) + 75.000

Model performance – R-squared (R^2)

- R-squared is a statistical measure of **how close the data are to the fitted regression line**
- It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression

- The definition of R-squared is straight-forward
  - → it is the percentage of the response variable variation that is explained by a linear model
  - R-squared = Explained variation / Total variation



Definition: Residual = Observed value - Fitted value

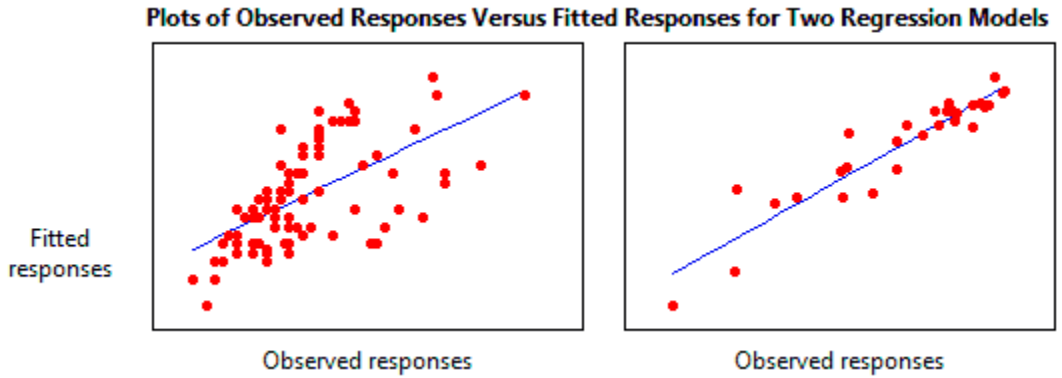
- R-squared is always between 0 and 100%:
  - 0% indicates that the model explains none of the variability of the response data around its mean
  - 100% indicates that the model explains all the variability of the response data around its mean

- In general, the **higher** the R-squared, the **better** the model **fits** your data

Image source: https://blog.minitab.com/en/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

Model performance – R^2

- Plotting fitted values by observed values graphically illustrates different R-squared values for regression models



- The regression model on the left accounts for 38.0% of the variance while the one on the right accounts for 87.4%
- The more variance that is accounted for by the regression model the closer the data points will fall to the fitted regression line
- Theoretically, if a model could explain 100% of the variance, the fitted values would always equal the observed values and, therefore, all the data points would fall on the fitted regression line

Image source: https://blog.minitab.com/en/adventures-in-statistics-2/regression-analysis-how-do-i-interpret-r-squared-and-assess-the-goodness-of-fit

OUTLINE

3.1 Supervised Machine Learning

- Overview
- Advantages/ disadvantages
- Types

3.2 Regression

- Linear regression
- Model optimization
- How to check performance?

3.3 Classification

- Logistic regression
- KNN
- Model optimization
- How to check performance?

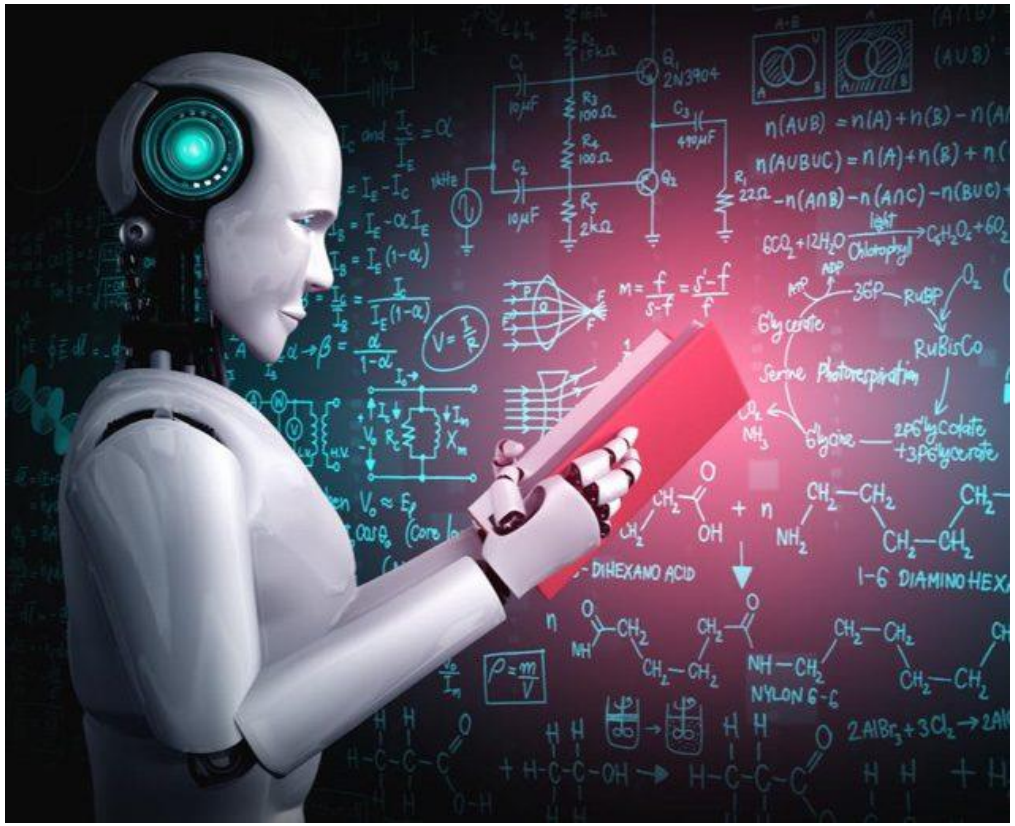


Image source: https://www.eweek.com/enterprise-apps/what-is-artificial-intelligence

3.3 CLASSIFICATION

- Classification is a process in which new observations are recognized and separated to categorize them

- You can classify something based on its features if you consider it a group of items, such as a collection of vegetables

For example, you could classify the potatoes, tomatoes, and peppers into A, B, and C categories

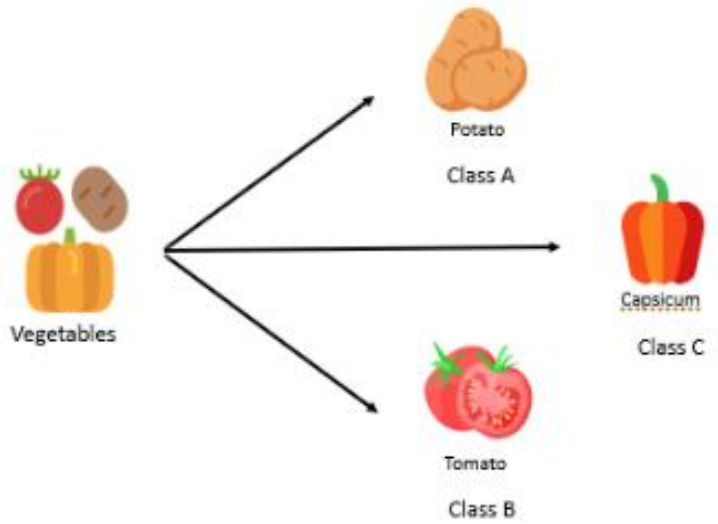


Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

Overview

- Based on training data, the classification algorithm is a supervised learning technique used to **categorize new observations**

- In classification, a program uses the **dataset** or observations provided to learn how to categorize new observations into various classes or groups
  - For instance, 0 or 1, red or blue, yes or no, spam or not spam, etc.
  - Targets, labels, or categories are terms that can be used to describe classes

- A **discrete output** function (y) is transferred to an input variable in the classification process (x)

Types of classification tasks

There are four different types of classification tasks in Machine Learning:

- **Binary** Classification
- **Multi-Class** Classification
- **Multi-Label** Classification
- **Imbalanced** Classification

Binary Classification

- Classification tasks with only **two class labels** are referred to as binary classification
  - Example: detection of spam email (spam or not)
- Binary classification problems often require two classes, one representing the **normal** state and the other representing the **abnormal** state
  - Example: the normal condition is "not spam," while the abnormal state is "spam"
  - Usually, **class label 0** is given to the class in the normal state, whereas **class label 1** is given to the class in the abnormal condition
- The following are **well-known** binary classification algorithms:
  - Support Vector Machines - SVM (natively binary)
  - **Logistic Regression** (natively binary)
  - Simple Bayes
  - Decision Trees

Multi-Class Classification

- Multi-class labels are used in classification tasks referred to as multi-class classification
  - Examples: categorization of faces, classifying plant species, character recognition
- The multi-class classification does not have the idea of normal and abnormal outcomes → instead, instances are grouped into one of several well-known classes
  - In some cases, the number of class labels could be rather high
  - Example: in a facial recognition system, a model might a model might include thousands or tens of thousands of faces
- **Well-known** algorithms:
  - **K Nearest Neighbours - KNN**
  - Simple Bayes
  - Decision trees
  - Random Forest classifier

Multi-Label Classification

- Multi-label classification problems are those that feature two or more class labels and allow for the prediction of **one or more class labels for each example**
  - Example: photo classification → here a model can predict the existence of many known things in a photo, such as “person”, “apple”, "bicycle," etc.
  - A particular photo may have multiple objects in the scene
- It is not possible to directly apply multi-label classification methods used for multi-class or binary classification

- The so-called **multi-label versions of the algorithms**, which are specialized versions of the conventional classification algorithms, include:
  - Multi-label Gradient Boosting
  - Multi-label Random Forests
  - Multi-label Decision Trees

Imbalanced Classification

- The term "imbalanced classification" describes classification jobs where the **distribution** of examples within each class is **not equal**
- A majority of the training dataset's instances belong to the normal class, while a minority belong to the abnormal class, making imbalanced classification tasks binary classification tasks in general
  - Examples: clinical diagnostic procedures, detection of outliers, fraud investigation

- **Well-known** algorithms:
  - One class SVM
  - Isolation forest
  - Local outlier factor - LOF

Logistic regression

- Logistic regression is the most famous machine learning algorithm after linear regression
  - In a lot of ways, linear regression and logistic regression are similar
- But, the **biggest difference** lies in **what they are used for**.
- **Linear** regression algorithms are used to **predict**/forecast values but **logistic** regression is used for **classification** tasks
- It is a supervised learning classification technique that **forecasts** the **likelihood** of a **target variable**
- There will only be a choice between two classes → **1** or **yes**, representing success, and **0** or **no**, representing failure

Logistic regression - Sigmoid Function

- Logistic regression algorithm also uses a linear equation with independent predictors to predict a value
  - The predicted value can be anywhere between negative infinity to positive infinity

$$z = \theta_0 + \theta_1 \cdot x_1 + \theta \cdot x_2 + \cdots$$

- We need the output of the algorithm to be **class** variable, i.e. 0-no, 1-yes
- Therefore, we are **squashing** the output of the linear equation into a **range** of **[0,1]**
  - To squash the predicted value between 0 and 1, we use the **sigmoid function**

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

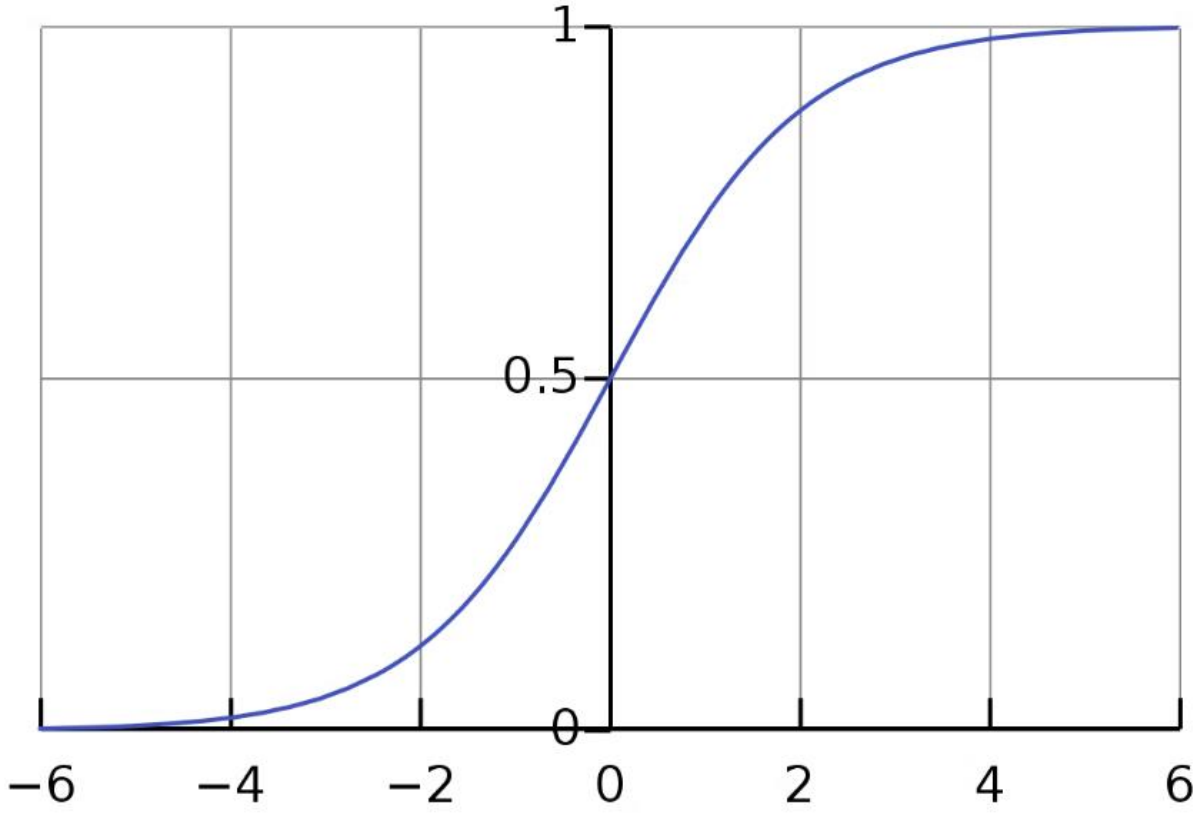
Source: https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36

Logistic regression - Sigmoid Function

- We take the output (z) of the linear equation and give to the function g(x) which returns a squashed value h, the value h will lie in the range of 0 to 1

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

As you can see from the graph, the sigmoid function becomes asymptote to y=1 for positive values of x and becomes asymptote to y=0 for negative values of x



Source: https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36

Model optimisation – cost function

- Since we are trying to predict class values, we cannot use the same cost function used in linear regression algorithm
- Therefore, we use a **logarithmic loss function** to calculate the cost for misclassifying

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$-\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Source: https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36

KNN

- K-Nearest Neighbors is one of the simplest supervised machine learning algorithms used for classification
- It classifies a data point based on its neighbors' classifications
- It stores all available cases and classifies new cases based on similar features
- K-Nearest Neighbor is a classification and prediction algorithm that is used to divide data into classes based on the distance between the data points

KNN, cont.

- K-Nearest Neighbor assumes that data points which are close to one another must be similar and hence, the data point to be classified will be grouped with the closest cluster

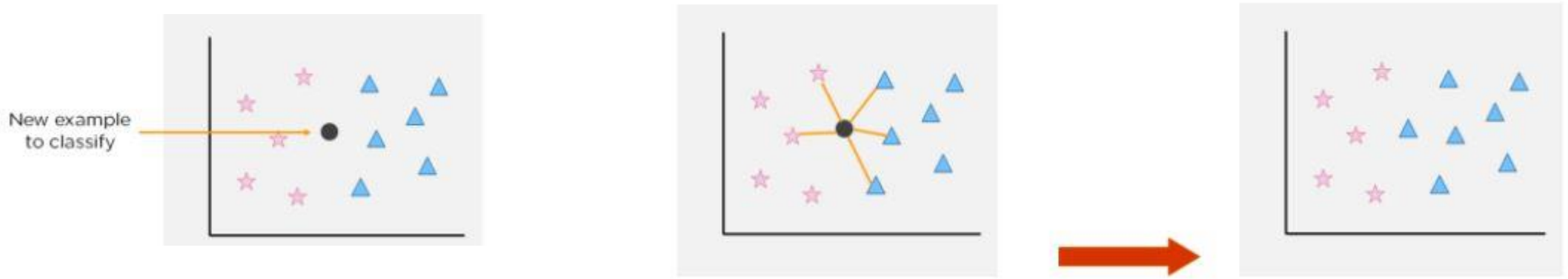
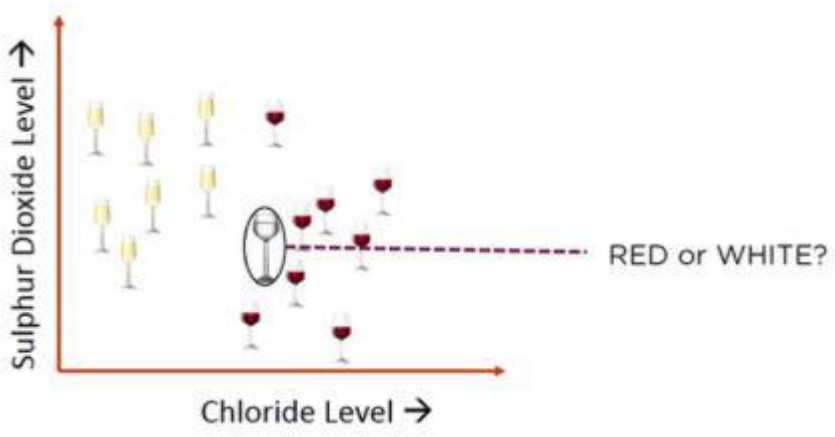
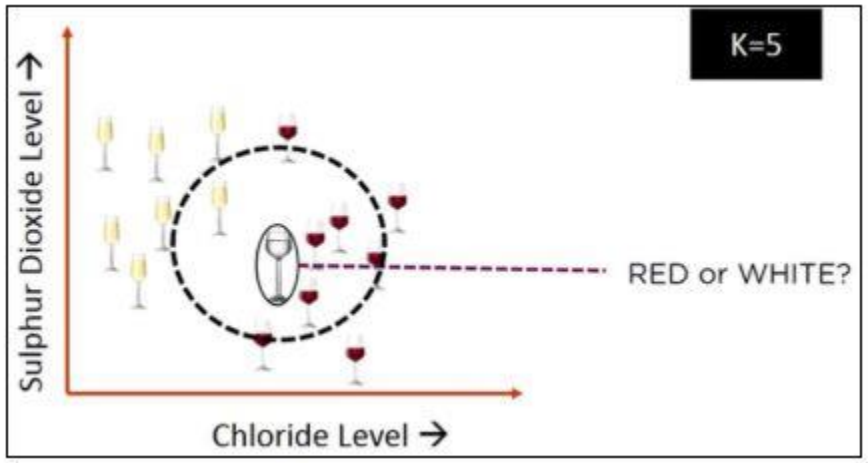


Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

KNN, cont.

- Example: predict if a glass of wine is red or white
- Different variables that are considered in this KNN algorithm include Sulphur dioxide and chloride levels
- K in KNN is a parameter that refers to the number of nearest neighbors in the majority voting process





- The majority votes from its fifth nearest neighbor and classifies the data point
- The glass of wine will be classified as red since four out of five neighbors are red

Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

KNN, cont.

- **Pros**:
  - Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly, which will not impact the accuracy of the algorithm.
  - KNN is very easy to implement. There are only two parameters required to implement KNN—the value of K and the distance function (e.g. Euclidean, Manhattan, etc.)

- **Cons**:
  - The KNN algorithm does not work well with large datasets. The cost of calculating the distance between the new point and each existing point is huge, which degrades performance.
  - Feature scaling (standardization and normalization) is required before applying the KNN algorithm to any dataset. Otherwise, KNN may generate wrong predictions

Model performance – confusion matrix

- The confusion matrix describes the model performance and gives us a matrix or table as an output
- The matrix is made up of the results of the forecasts in a condensed manner, together with the total number of right and wrong guesses.

		Predicted condition	
	Total population = P + N	Positive (PP)	Negative (PN)
Actual condition	Positive (P)	True positive (TP)	False negative (FN)
	Negative (N)	False positive (FP)	True negative (TN)

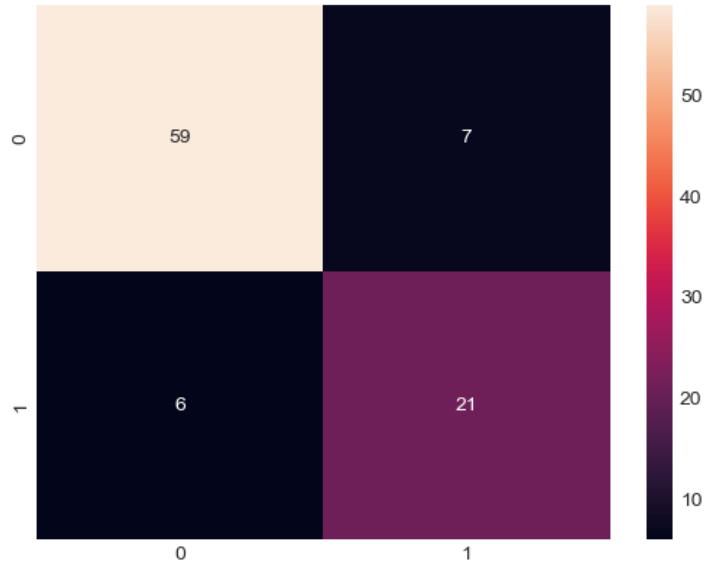


Image source: https://en.wikipedia.org/wiki/Confusion_matrix

Model performance – confusion matrix

Sources: [21][22][23][24][25][26][27][28][29] view · talk · edit

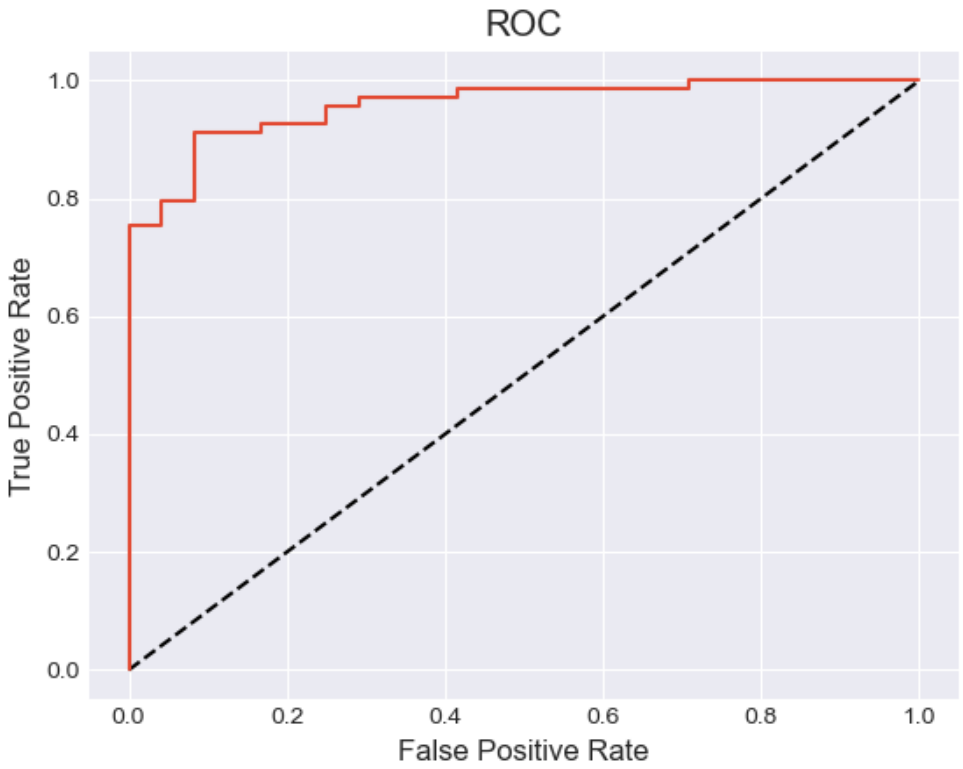
		Predicted condition			
	Total population = P + N	**Positive (PP)**	**Negative (PN)**	Informedness, bookmaker informedness (BM) = TPR + TNR − 1	Prevalence threshold (PT) = $\frac{\sqrt{TPR \times FPR} - FPR}{TPR - FPR}$
Actual condition	**Positive (P)**	**True positive (TP),** hit	**False negative (FN),** type II error, miss, underestimation	True positive rate (TPR), recall, sensitivity (SEN), probability of detection, hit rate, power = $\frac{TP}{P}$ = 1 − FNR	False negative rate (FNR), miss rate = $\frac{FN}{P}$ = 1 − TPR
	Negative (N)	**False positive (FP),** type I error, false alarm, overestimation	**True negative (TN),** correct rejection	False positive rate (FPR), probability of false alarm, fall-out = $\frac{FP}{N}$ = 1 − TNR	True negative rate (TNR), specificity (SPC), selectivity = $\frac{TN}{N}$ = 1 − FPR
	Prevalence = $\frac{P}{P+N}$	Positive predictive value (PPV), precision = $\frac{TP}{PP}$ = 1 − FDR	False omission rate (FOR) = $\frac{FN}{PN}$ = 1 − NPV	Positive likelihood ratio (LR+) = $\frac{TPR}{FPR}$	Negative likelihood ratio (LR−) = $\frac{FNR}{TNR}$
	Accuracy (ACC) = $\frac{TP+TN}{P+N}$	False discovery rate (FDR) = $\frac{FP}{PP}$ = 1 − PPV	Negative predictive value (NPV) = $\frac{TN}{PN}$ = 1 − FOR	Markedness (MK), deltaP (Δp) = PPV + NPV − 1	Diagnostic odds ratio (DOR) = $\frac{LR+}{LR-}$
	Balanced accuracy (BA) = $\frac{TPR+TNR}{2}$	F_1 score = $\frac{2PPV \times TPR}{PPV+TPR} = \frac{2TP}{2TP+FP+FN}$	Fowlkes–Mallows index (FM) = $\sqrt{PPV \times TPR}$	Matthews correlation coefficient (MCC) = $\sqrt{TPR \times TNR \times PPV \times NPV} - \sqrt{FNR \times FPR \times FOR \times FDR}$	Threat score (TS), critical success index (CSI), Jaccard index = $\frac{TP}{TP+FN+FP}$

used in imbalanced dataset.

Image source: https://en.wikipedia.org/wiki/Confusion_matrix

Model performance – ROC curve

- AUC is for Area Under the Curve, and ROC refers to Receiver Operating Characteristics Curve
- It is a graph that displays the classification model's performance at various thresholds
- The AUC-ROC Curve is used to show how well the multi-class classification model performs
- The TPR and FPR are used to draw the ROC curve, with the True Positive Rate (TPR) on the Y-axis and the FPR (False Positive Rate) on the X-axis



3. SUPERVISED MACHINE LEARNING, cont.

OUTLINE

3.4 Cross-validation

3.5 Hyperparameter tunning

3.6 Support Vector Machines (SVM)

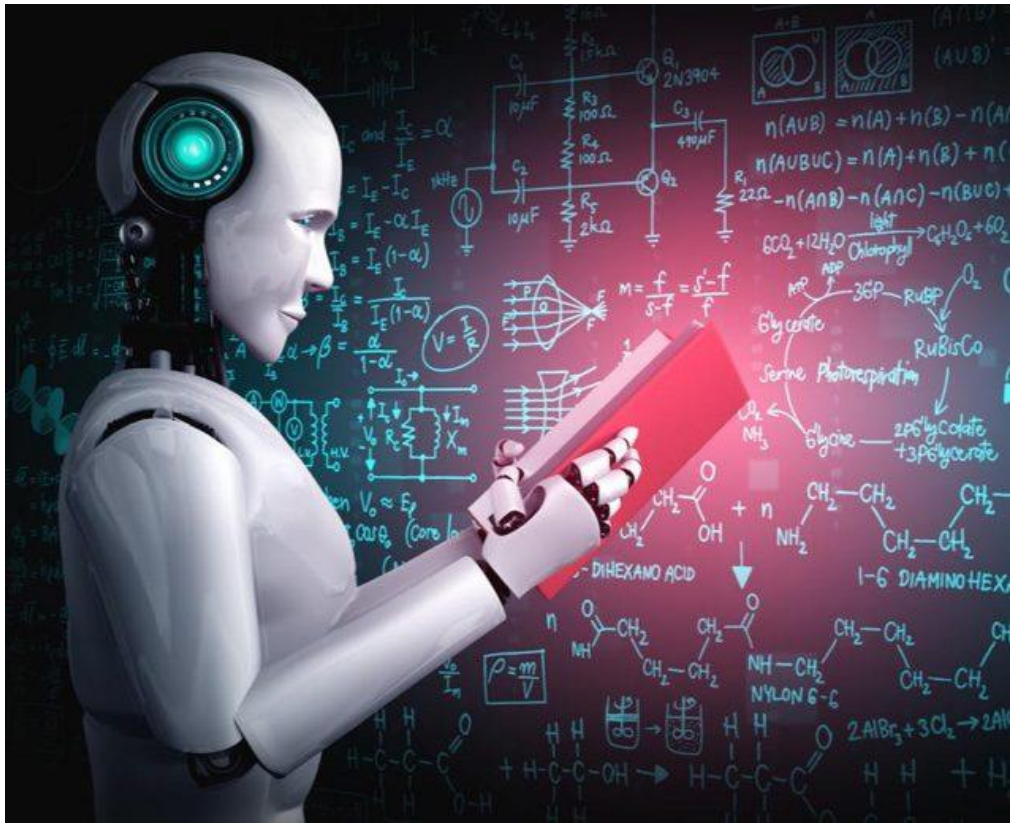


Image source: https://www.eweek.com/enterprise-apps/what-is-artificial-intelligence

3.4 CROSS-VALIDATION

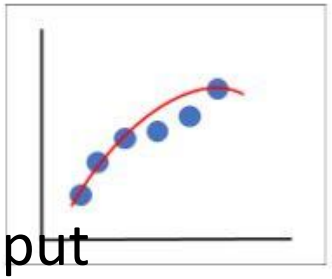
- How do we know if our model is functional? If we have trained it well?

- All of this can be determined by seeing how our model performs on **previously unseen data**, data that is completely new to it

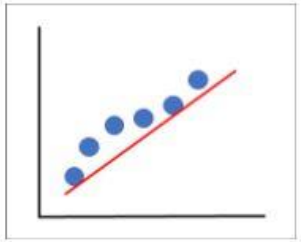
- We need to ensure that the **accuracy** of our model **remains constant**
  - → We need to **validate** our model

- Using **cross-validation** in machine learning, we can determine how our model is performing on previously unseen data and test its accuracy

Model stability



- Any machine learning model needs to **consistently predict** the correct output across a variation of different input values → **stability**
- If a model does not change much when the input data is modified, it means that it has been trained **well** to **generalize** and find patterns in our data
- A model can **lose stability** in two ways:
  - **Underfitting**:
    - It occurs when the model does not fit properly to training data
    - It under-performs on both known and unseen data
  - **Overfitting**:
    - When the model trains well on training data and generalizes to it, but fails to perform on new, unseen data
    - It captures every little variation in training data and cannot perform on data that does not have the same variations



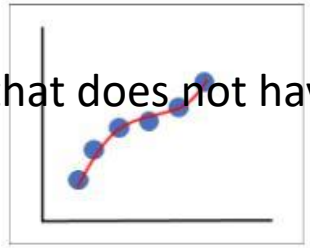


Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

Cross-validation

- While choosing machine learning models, we need to **compare different models**
- However, **data** is usually **limited**, our dataset might not have enough data points or may even have missing or wrong data
- Further, if we have fewer data, **training**, and **testing** on the **same portion of data** does not give us an accurate view of how our model performs
  - → This is where **cross-validation** comes into the picture
- Cross-Validation in machine learning is a technique that is used to **train and evaluate our model on a portion of our database**, before **re-portioning** our dataset and evaluating it on the new portions
  - → Our model is trained and tested on new data at every new step

Cross-validation

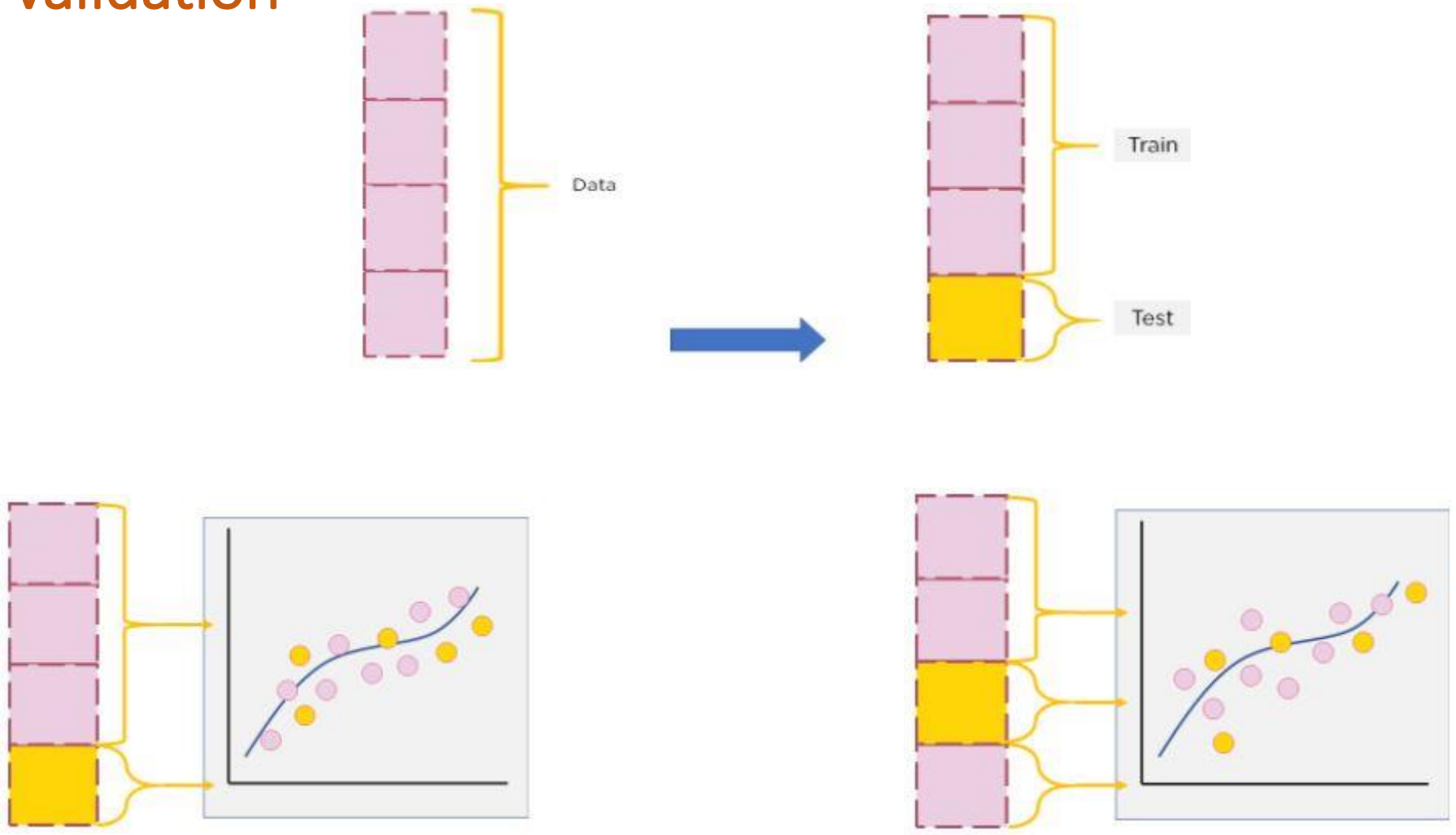


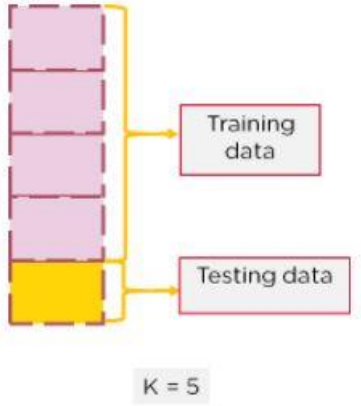
Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

Cross-Validation Models

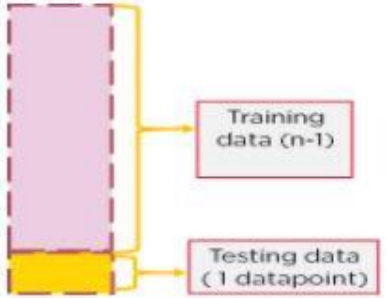
- **K-fold** cross-validation:
  - K refers to the number of portions the dataset is divided into

- **Leave one out** cross-validation (LOOCV):
  - We select one data point as the test data;
  - The rest of the dataset will be used for training and the single data point will be used to predict after training

- **Stratified K-fold** cross-validation:
  - This method is useful when there are minority classes present in our data
  - In some cases, while partitioning the data, some testing sets will include instances of minority classes while others will not
  - The data is split so that each portion has the same percentage of all the different classes that exist in the dataset



Final Measure = Average(Measure 1, Measure2,....Measure K)



Final Measure = Average(Measure 1, Measure2,....Measure N)

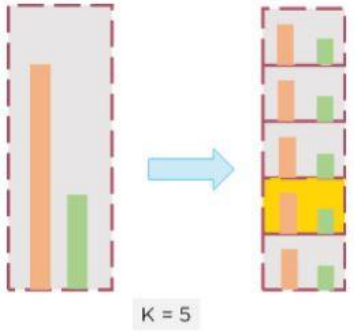


Image source: https://www.simplilearn.com/tutorials/machine-learning-tutorial

3.5 HYPER-PARAMETER TUNNING

- Hyper-parameters are parameters that are not directly learnt within estimators
  - Examples: C, kernel and gamma for Support Vector Classifier, alpha for Lasso, etc.
- It is possible and recommended to search the hyper-parameter space for the best cross validation score
- Any parameter provided when constructing an estimator may be optimized in this manner
- Two generic approaches to parameter search are provided in scikit-learn:
  - **GridSearchCV** exhaustively considers all parameter combinations
  - **RandomizedSearchCV** can sample a given number of candidates from a parameter space with a specified distribution

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```

Image source: https://scikit-learn.org/stable/modules/grid_search.html

3.6 SUPPORT VECTOR MACHINES (SVM)

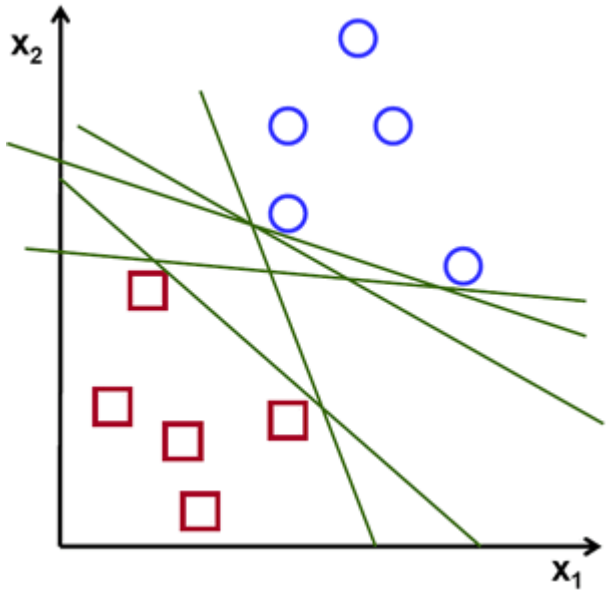
- Support vector machines (SVMs) are a set of supervised learning methods used for **classification**, **regression** and **outliers detection**

- The **advantages** of SVMs are:
  - Effective in high dimensional spaces
  - Still effective in cases where number of dimensions is greater than the number of samples
  - Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient

- The **disadvantages** of SVMs include:
  - If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions
  - SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation

What is SVM?

- The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points
- Objective is to find a plane that has the maximum margin, i.e the maximum distance between data points of both classes



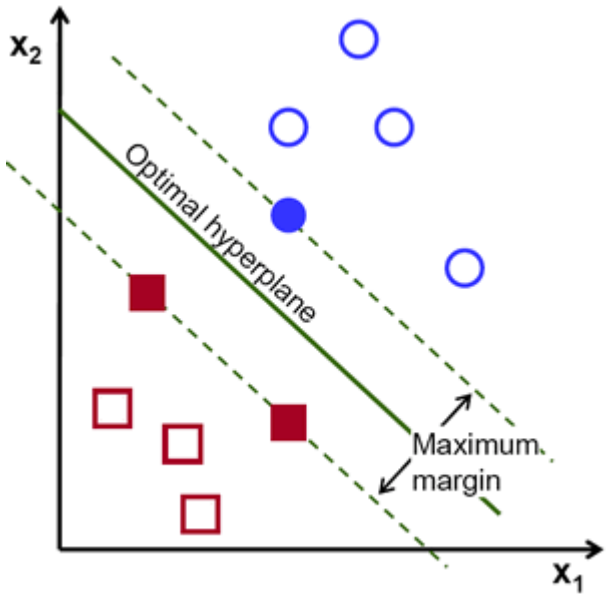
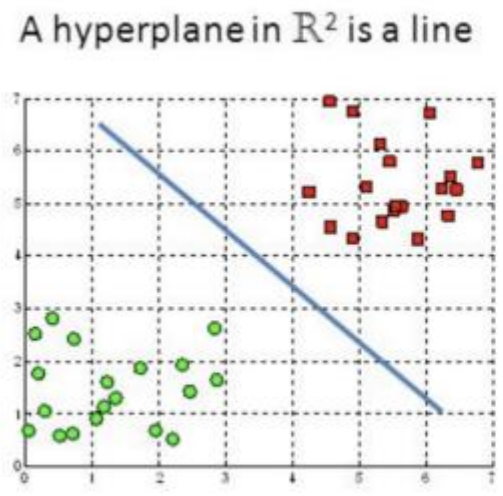


Image source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Hyperplanes and support vectors

- Hyperplanes are decision boundaries that help classify the data points
- Data points falling on either side of the hyperplane can be attributed to different classes
- Also, the dimension of the hyperplane depends upon the number of features



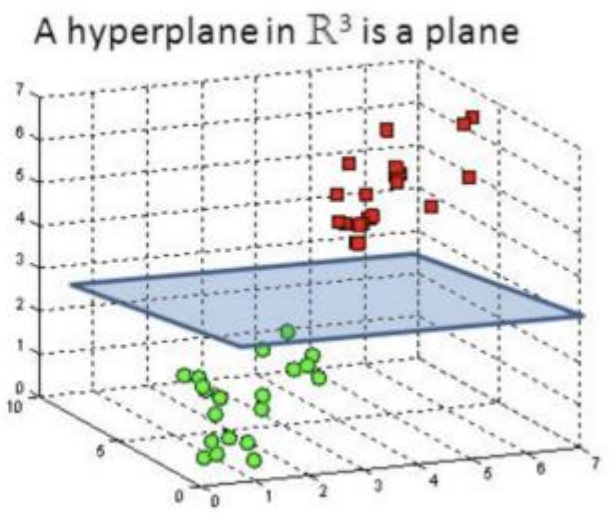


Image source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Support vectors

- Support vectors are **data points** that are **closer to the hyperplane** and influence the position and orientation of the hyperplane
- Using these support vectors, we maximize the margin of the classifier
- Deleting the support vectors will change the position of the hyperplane
- These are the points that help us build our SVM

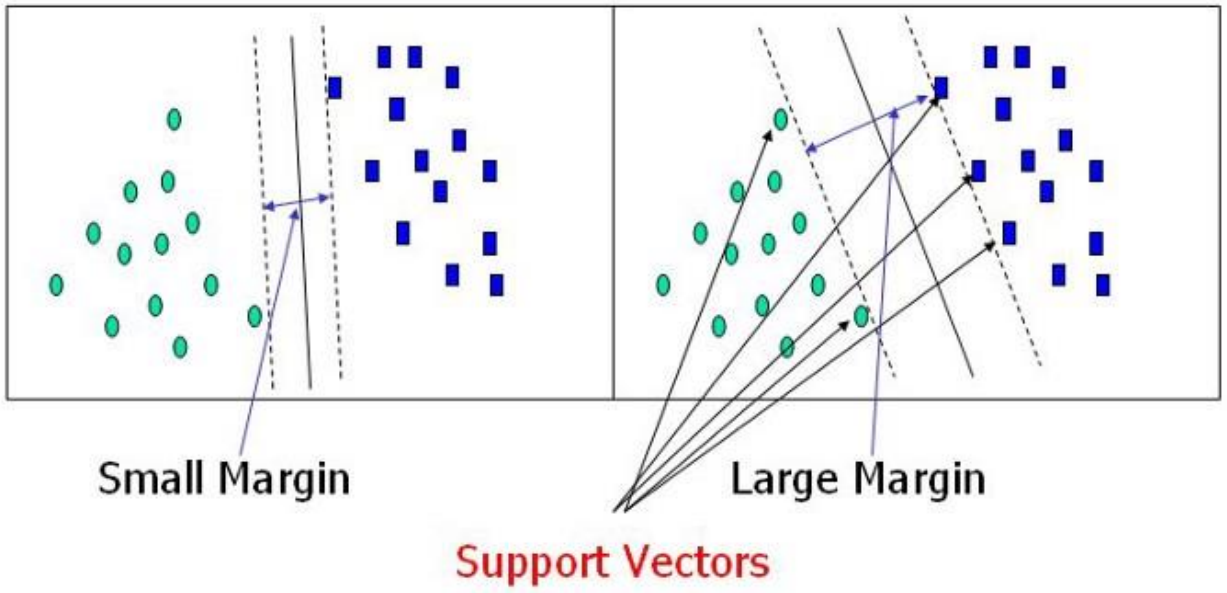


Image source: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Large margin intuition

- In **logistic regression**
  - We take the output of the linear function and squash the value within the range of [0,1] using the sigmoid function
  - If the squashed value is greater than a threshold value (0.5) we assign it a label 1, else we assign it a label 0

- In **SVM**
  - We take the output of the linear function and if that output is greater than 1, we identify it with one class and if the output is -1, we identify it with another class
  - Since the threshold values are changed to 1 and -1 in SVM, we obtain this reinforcement range of values ([-1,1]) which acts as margin
  - We are looking to maximize the margin between the data points and the hyperplane