NWiMLDS: Explore Series

# Machine Learning

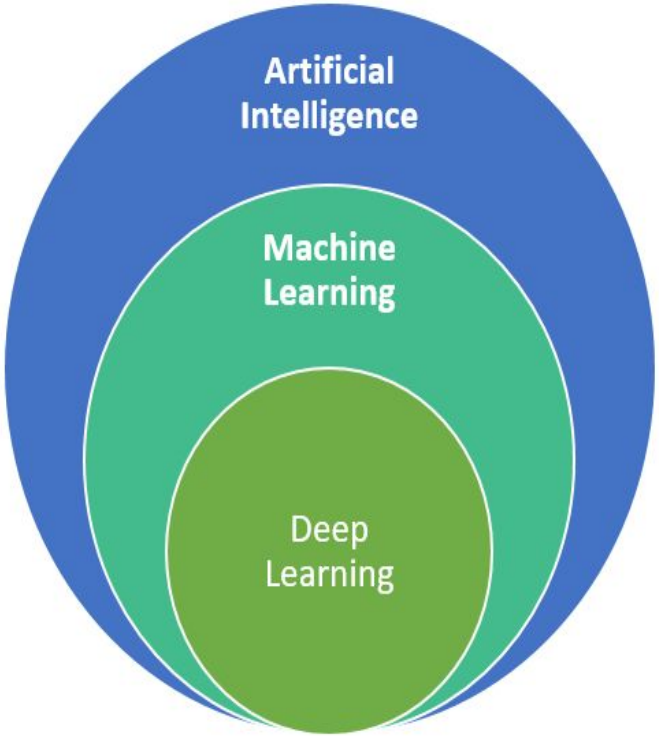
Zakia Mustafa

Lydia Gathoni

# Objectives

- Introduction
- Diving into Machine Learning
- Supervised Learning Algorithms
- Model Selection and Evaluation
- Evaluation Metrics
- Codelabs

# Introduction

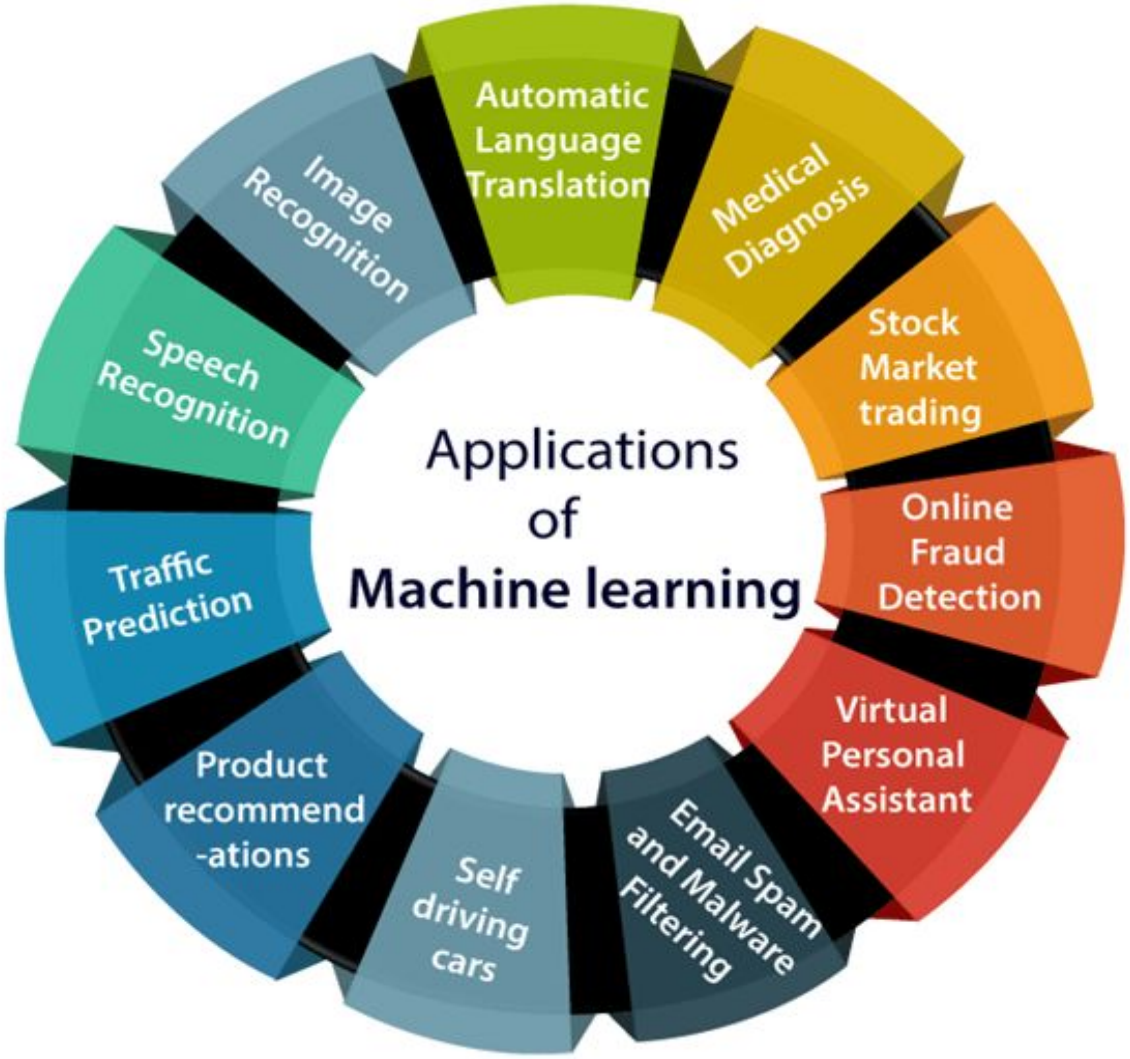



**What is Machine Learning:**

**Definition** -

Machine Learning (ML) is a subset of Artificial Intelligence where we develop algorithms that make a machine learn to do something without actually making computations about it. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.
—Arthur Samuel, 1959


Applications
of
Machine learning
Automatic Language Translation
Medical Diagnosis
Stock Market trading
Online Fraud Detection
Virtual Personal Assistant
Email Spam and Malware Filtering
Self driving cars
Product recommend -ations
Traffic Prediction
Speech Recognition
Image Recognition

# Key Terminologies

Labels - A **label** is the thing we're predicting,

Features - A **feature** is an input variable, could be one or many

Model - A model defines the relationship between features and label

Training set – Set of data to discover potentially predictive relationships, used to train

# Types of Machine Learning – At a Glance

## Supervised Learning

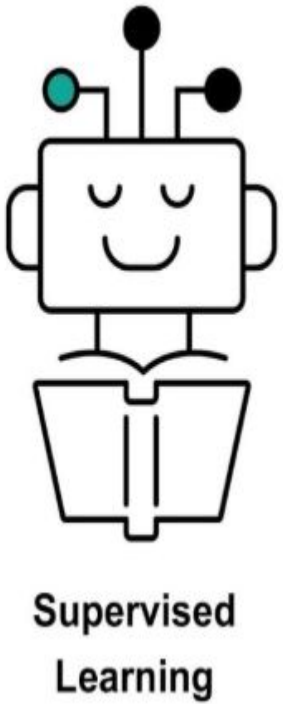
- Makes machine Learn explicitly
- Data with clearly defined output is given
- Direct feedback is given
- Predicts outcome/future
- Resolves classification and regression problems


## Unsupervised Learning

- Machine understands the data (Identifies patterns/structures)
- Evaluation is qualitative or indirect
- Does not predict/find anything specific


## Reinforcement Learning

- An approach to AI
- Reward based learning
- Learning form +ve & +ve reinforcement
- Machine Learns how to act in a certain environment
- To maximize rewards




- ➢ Algorithms are designed to learn by example
- ➢ Data consists of inputs paired with the correct output
- ➢ Algorithm will search for patterns in the data that will correlate with the desired output

**Types of Supervised Learning**

- **Regression:** Process of predicting a continuous value e.g. Co2 emissions of different cars.
- **Classification:** Process of predicting a discrete class label or category e.g. object recognition

**Popular Algorithms**

Linear Regression, Logistic Regression, Support Vector Machine, Decision Trees, KNN etc

# Classification example

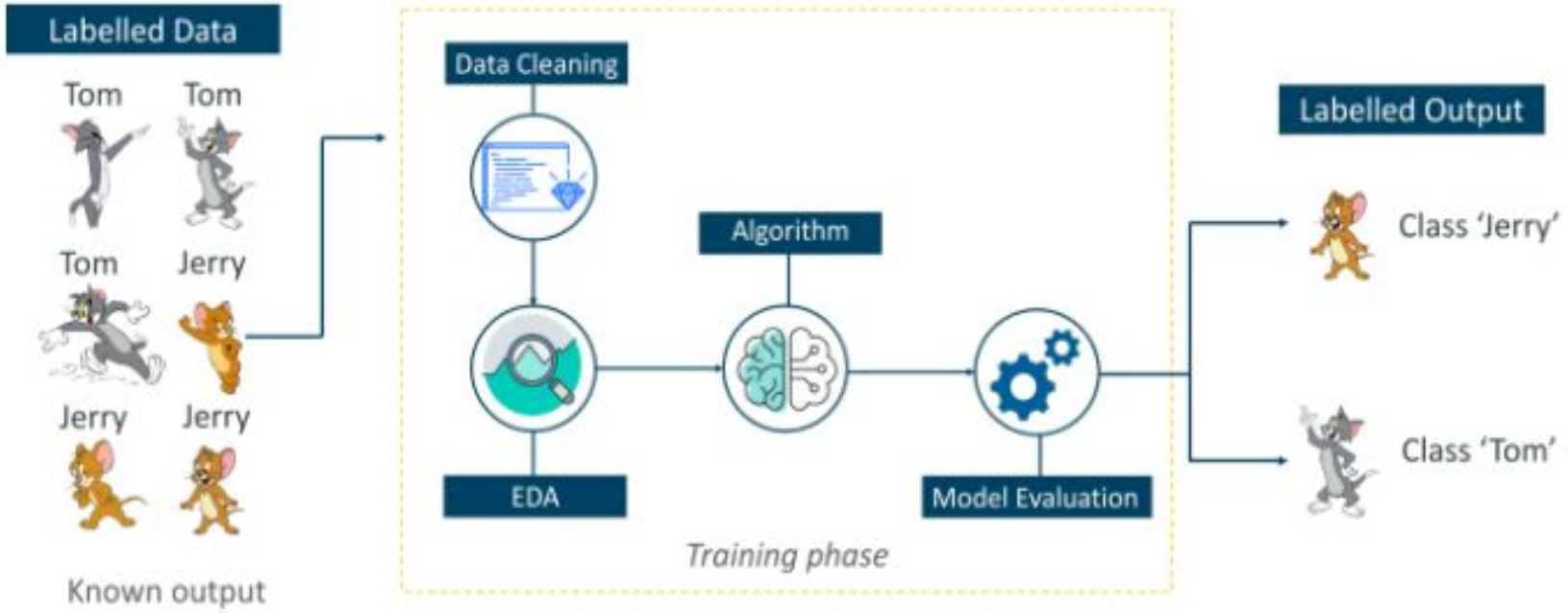
| Id | Cl.thickness | Cell.size | Cell.shape | Marg.adhesion | Epith.c.size | Bare.nuclei | Bl.cromatin | Normal.nucleoli | Mitoses | Class |
|---|---|---|---|---|---|---|---|---|---|---|
| 1000025 | 5 | 1 | 1 | 1 | 2 | 1 | 3 | 1 | 1 | benign |
| 1002945 | 5 | 4 | 4 | 5 | 7 | 10 | 3 | 2 | 1 | benign |
| 1015425 | 3 | 1 | 1 | 1 | 2 | 2 | 3 | 1 | 1 | benign |
| 1016277 | 6 | 8 | 8 | 1 | 3 | 4 | 3 | 7 | 1 | benign |
| 1017023 | 4 | 1 | 1 | 3 | 2 | 1 | 3 | 1 | 1 | benign |
| 1017122 | 8 | 10 | 10 | 8 | 7 | 10 | 9 | 7 | 1 | malignant |
| 1018099 | 1 | 1 | 1 | 1 | 2 | 10 | 3 | 1 | 1 | benign |
| 1018561 | 2 | 1 | 2 | 1 | 2 | 1 | 3 | 1 | 1 | benign |
| 1033078 | 2 | 1 | 1 | 1 | 2 | 1 | 1 | 1 | 5 | benign |
| 1033078 | 4 | 2 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | benign |
| 1035283 | 1 | 1 | 1 | 1 | 1 | 1 | 3 | 1 | 1 | benign |
| 1036172 | 2 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 1 | benign |
| 1041801 | 5 | 3 | 3 | 3 | 2 | 3 | 4 | 4 | 1 | malignant |

Discrete values

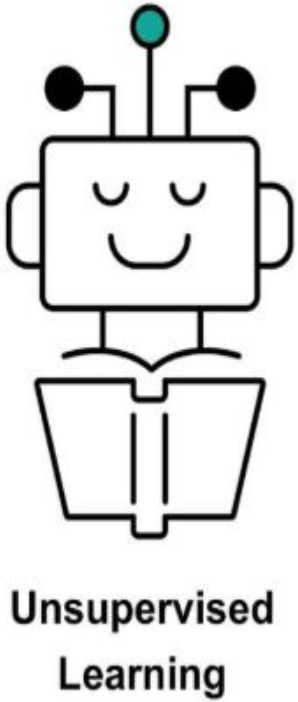
# Regression example

| | ENGINESIZE | CYLINDERS | FUELCONSUMPTION_COMB | CO2EMISSIONS |
|---|---|---|---|---|
| 0 | 2.0 | 4 | 8.5 | 196 |
| 1 | 2.4 | 4 | 9.6 | 221 |
| 2 | 1.5 | 4 | 5.9 | 136 |
| 3 | 3.5 | 6 | 11.1 | 255 |
| 4 | 3.5 | 6 | 10.6 | 244 |
| 5 | 3.5 | 6 | 10.0 | 230 |
| 6 | 3.5 | 6 | 10.1 | 232 |
| 7 | 3.7 | 6 | 11.1 | 255 |
| 8 | 3.7 | 6 | 11.6 | 267 |
| 9 | 2.4 | 4 | 9.2 | ? |

Continuous values




Example of Supervised Learning.



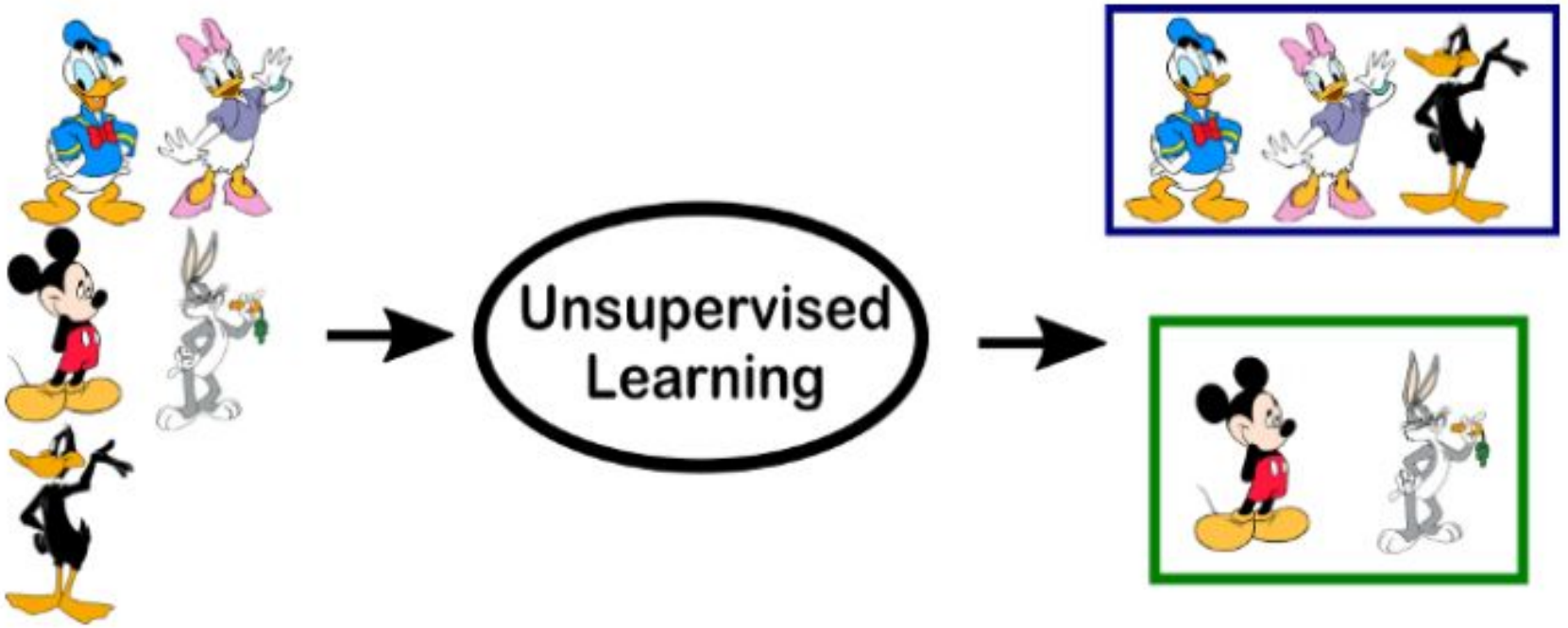

- No knowledge of the output label
- No knowledge of the output label nor the outcome or target to predict.
- Algorithms models the underlying or hidden structure in the data in order to learn more about the data

**Types of Unsupervised Learning**

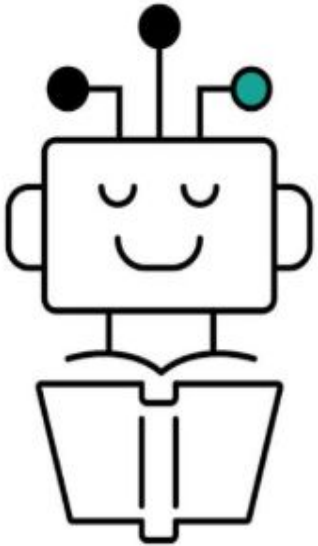
- **Clustering:** A clustering problem is where we group similar data according to a pattern in data, such as grouping customers by purchasing behavior.
- **Association:** An association rule learning problem is where we want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.

**Popular Algorithms**

K-Means,  Dimensionality Reduction Techniques (PCA, SVD), Apriori etc

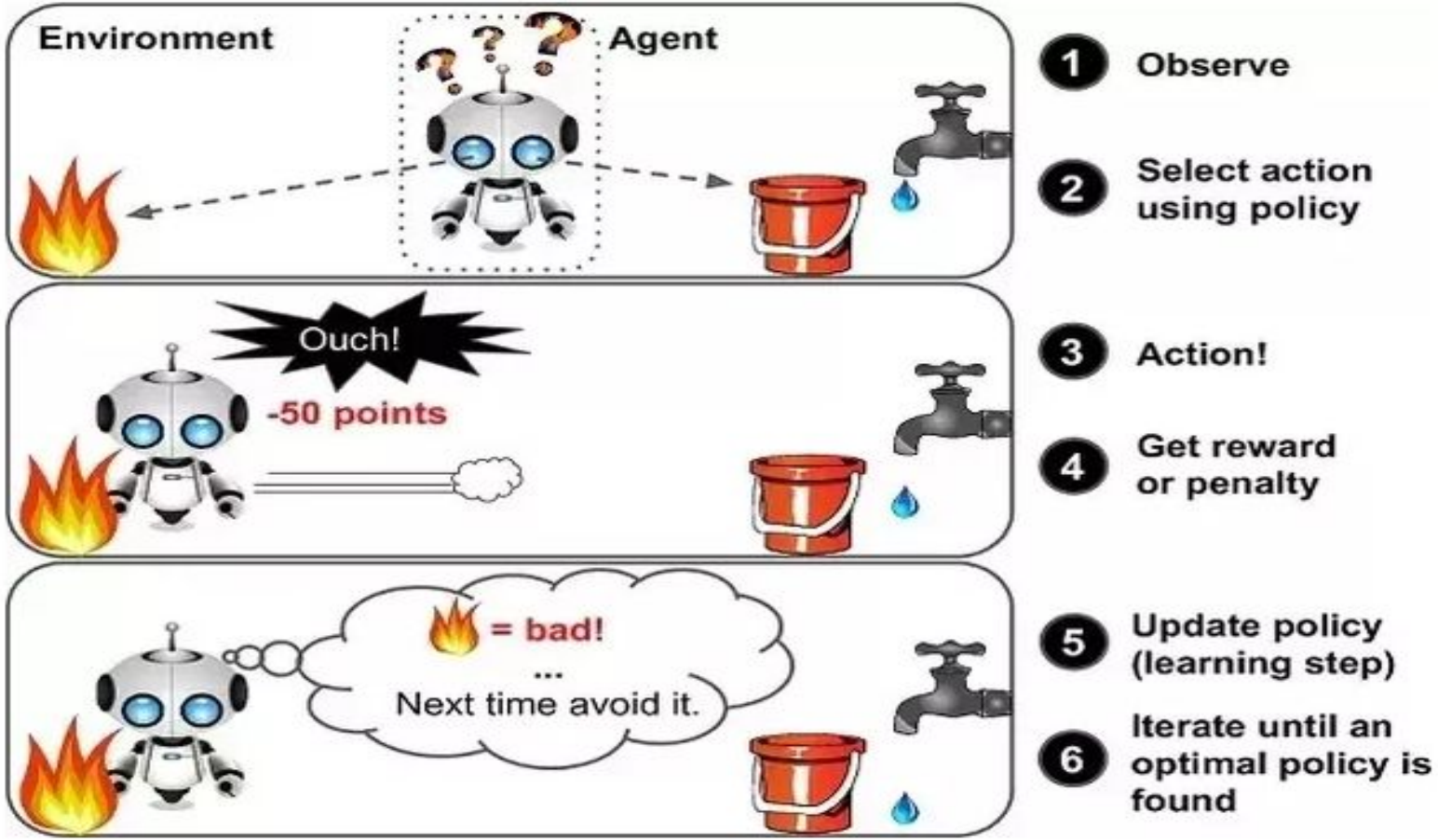



Example of Unsupervised Learning



**Reinforcement Learning**

- The algorithm learns by interacting with the environment.
- It adjusts itself based on feedback.
- The algorithm works on reward a system




Example of Reinforcement Learning

# Pipeline of a typical ML Problem


**Collect data.**

**Kaggle datasets**

**Zindi Africa**

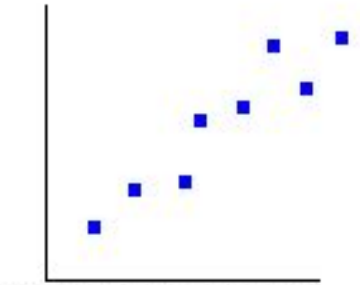
**Open Source Malaria**

UC Irvine Machine Learning Repository

Amazon AWS datasets

# Generalization, Overfitting and Underfitting

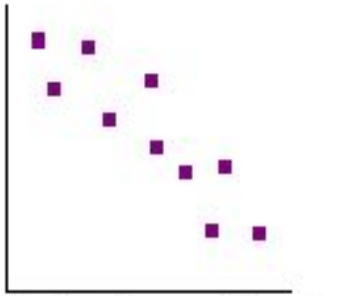
# 1. Linear Regression



Postive linear relationship

Allows us to predict some dependent variables (y) bases on a independent variable (x) by finding an optimal linear relationship between them.

It examines the linear relationship between two (Simple Linear Regression ) or more (Multiple Linear Regression) variables — a dependent variable and independent variable(s)



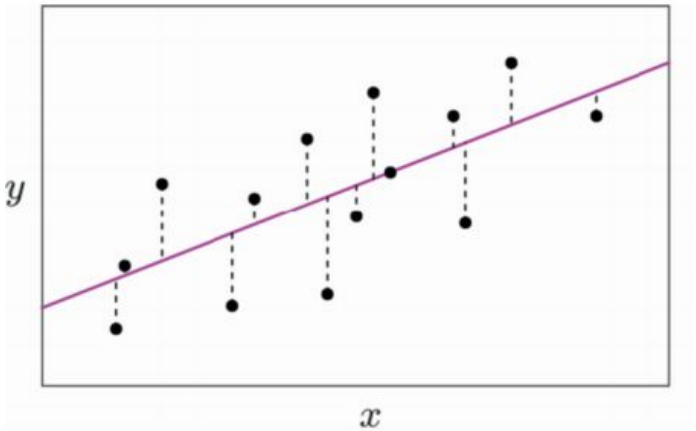
Negative linear relationship

- **Simple:**

  $y = b_0 + b_1 * x$

- **Multiple:**

  $y = b_0 + b_1 * x_1 + \ldots + b_n * x_n$



# Assumptions

- **Linearity.** Linear regression assumes that the relationship between your input and output is linear. It does not support anything else.
- **Remove Noise.** Linear regression assumes that your input and output variables are not noisy.
- **Remove Collinearity.** Linear regression will over-fit your data when you have highly correlated input variables.
- **Gaussian Distributions.** Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution.
- **Rescale Inputs:** Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.

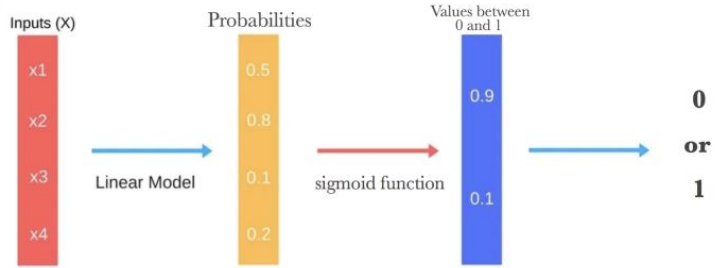
# In Python

```
# Separating the target from our data
X = features
y = target variable
#  Split the data into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

# apply linear regression
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()
regressor.fit() #The fit method will look for our optimal line.

#More practice on the hands-on labs
```

# 2. Logistic Regression

Used for classification tasks

Predict a categorical outcome with discrete probability outcome**.**




Probabilities are transformed into binary values in order to actually make a prediction.
This is the task of the logistic function, also called the sigmoid function.

## Types of logistic regression

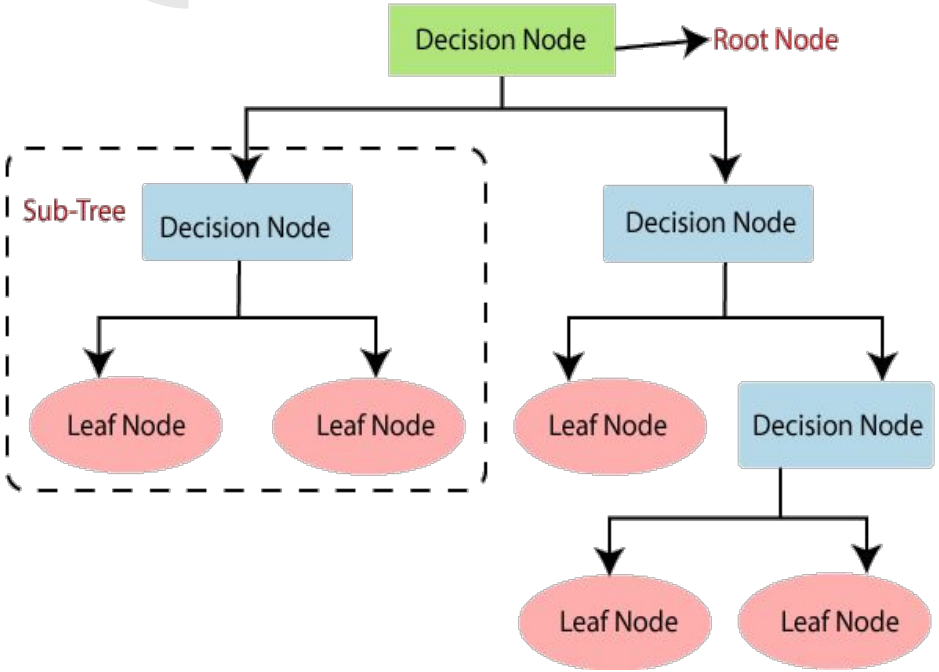
- Binomial (Pass/Fail)
- Multinomial (Cats, Dogs, Sheep)
- Ordinal (Low, Medium, High)

# Python Code

```
# Separating the target from our data
X = features
y = target variable
#  Split the data into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

# Apply logistic regression
from sklearn.linear_model import LogisticRegression
LogReg = LogisticRegression()
LogReg.fit()

#More practice on the hands-on labs
```

# 3. Decision Trees

Can be used for both classification and Regression problems




Internal decision nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

More info: https://youtu.be/DCZ3tsQIoGU

# How does the Decision Tree algorithm Work?

The algorithm starts from the root node of the tree.

- **Step-1:** Begin the tree with the root node, says A, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the A into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3.

Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

# Attribute Selection Measures

Selecting the best attribute for the root node and for sub-nodes. There is a technique which is called as Attribute selection measure or ASM.

There are two popular techniques for ASM, which are:

- **Information Gain -** Calculates how much information a feature provides us about a class

  According to the value of information gain, we split the node and build the decision tree

- **Gini Index -** a metric to measure how often a randomly chosen element would be incorrectly identified

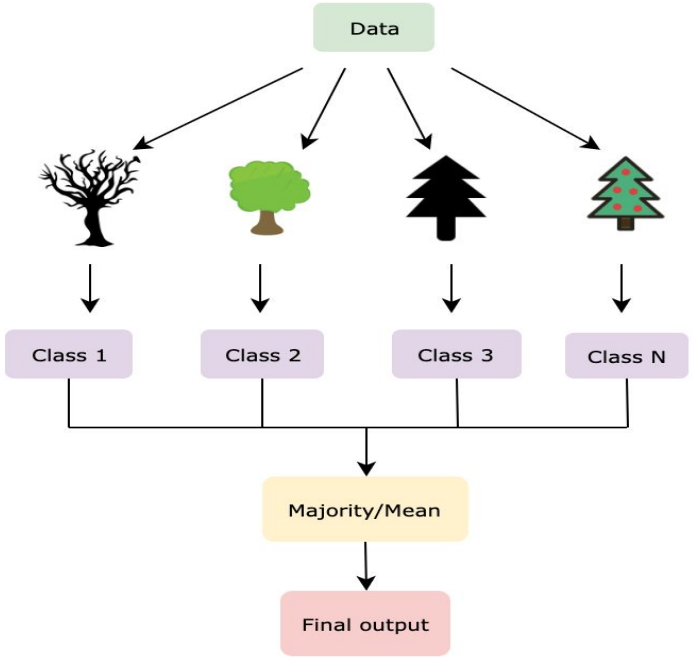
# Python Code

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
# Split the independent and dependent variables
y = target
X = features
# Train using 70% of the data.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# Create a decision tree classifier
tree = DecisionTreeRegressor()
# Train it on our training set.
tree = tree.fit(X_train, y_train)
# Predict based on the model we've trained
y_pred = tree.predict(X_test)
```

# 4. Random Forest

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

Used for both Classification and Regression problems




It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

# Assumptions

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

# Code..

```
from sklearn.model_selection import train_test_split
y = target
X = features

# Train using 80% of the data.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
# Import the model
from sklearn.ensemble import RandomForestRegressor
forest = RandomForestRegressor(n_estimators = 100, random_state=42, min_samples_split =
20, max_depth=5)
forest = forest.fit(X_train, y_train)

# Predict based on the model we've trained
y_pred = forest.predict(X_test)
```

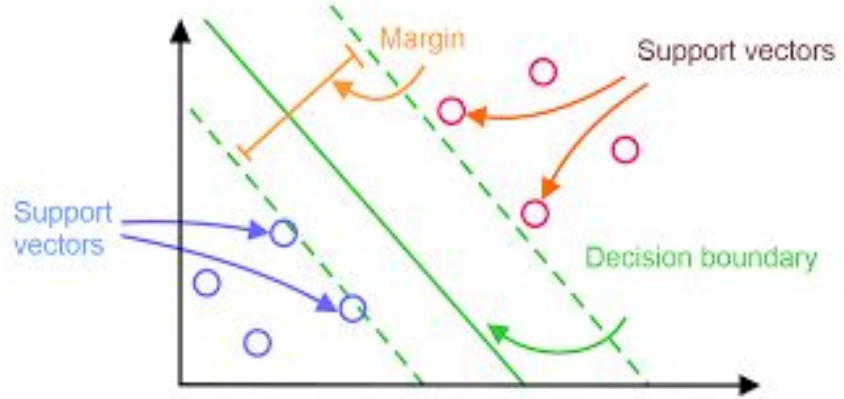
# 5. Support Vector Machines

# Support Vector Machines

Used for classification and regression problems

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.

To separate the two classes of data points, there are many possible hyperplanes(decision boundary) that could be chosen.



Terms you'll encounter while implementing SVM

- **Kernel:** In SVM, this is a function that quantifies the similarities of two observations. In other words, it is a function that maps a lower dimension data to a higher dimension data.
- **Regularization:** Python Scikit-learn library has a regularization parameter denoted by C.  C is the penalty parameter that represents misclassification or the error term. The misclassification or error term tells the SVM optimization how much error is bearable. A small value of C creates a small margin hyperplane and a larger value of C creates a large margin hyperplane.

# Python Code

```
# Separating the target from our data
 X = features
y = target variable
#  Split the data into train and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 0)

# Let's now build the svm model
model= SVC(kernel = 'linear')
# Train the model using the training set
model.fit(X_train,y_train)

# Predict the response for the test set
y_pred
```

# 6. Naive Bayes

The **Naive Bayes Classifier** is a statistical classification technique based on the Bayes Theorem. It has high accuracy and speed on large datasets.

This type of classifier takes into account the assumption that the effect of a particular feature in a class is independent of other features.

The assumption can be expressed as:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

- P(h): The prior probability of h - the probability of hypothesis h being true (regardless of the data).
- P(D): The prior probability - the probability of the data (regardless of the hypothesis).
- P(h|D): The posterior probability - the probability of hypothesis h given the data D.
- P(D|h): The posterior probability - the probability of data d given that the hypothesis h was true.

**Three variations of the Naive Bayes algorithm depending on the distribution of the data**

**1. Gaussian:** This type of classifier assumes the distribution of features to be Gaussian or normal. This is the most common type of Naive Bayes algorithm that we will come across.

```
from sklearn.naive_bayes import GaussianNB
Model = GaussianNB().fit(X, y)
```

**2. Multinomial:** This type of classifier is used when the data is distributed multinomially, i.e., multiple occurrences matter a lot.

```
from sklearn.naive_bayes import MultinomialNB
model = MultinomialNB().fit(X, y)
```

**3. Bernoulli:** This type of classifier is used when the features in the data set are binary-valued.

```
from sklearn.naive_bayes import BernoulliNB
Model = BernoulliNB().fit(X, y)
```

# Summary on when to use which model

# Model Evaluation and Hyperparameter Tuning

# Cross Validation

This is the automatic optimization of the hyperparameters of a machine learning model.

These hyperparameters are configuration variables that are external to the model and whose values cannot be estimated from data.

A hyperparameter as a parameter whose value is set before the learning process begins with the goal of searching across various hyperparameter configurations to find a configuration that results in the best performance.

Examples of model hyperparameters;

1. The C and sigma hyperparameters for support vector machines.
2. Number of leaves or depth of a tree.
3. The k in k-nearest neighbors.

**Why are hyperparameters important?**

- They help define higher-level concepts about the model,  such as its complexity and ability to learn.
- With the right parameters, they can eliminate the chances of overfitting and underfitting

**How do we perform hyperparameter tuning?**

There are several approaches to hyperparameter tuning:

**1. Manual:** Select hyperparameters based on intuition/experience/guessing

**2. Grid Search:** Try out a bunch of hyperparameters from a given set of hyperparameters, and see what works best.

**3. Random search:** Try out a bunch of random hyperparameters from a uniform distribution over some hyperparameter space, and see what works best.

**4. Ensemble method: combining the models that perform best**

# Evaluation Metrics for Machine Learning Models

# 1. Model Accuracy:

Model accuracy in terms of classification models can be defined as the ratio of correctly classified samples to the total number of samples

$$\textbf{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

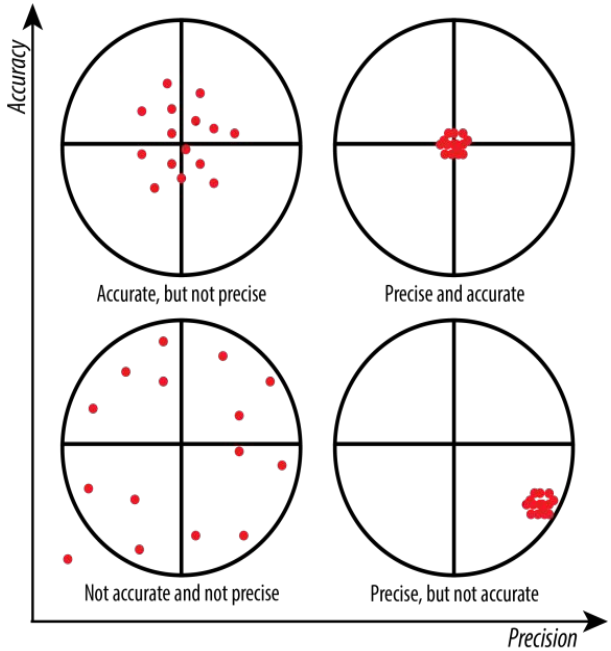
Or for binary classification models, the accuracy can be defined as:

$$\textbf{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

| | | Predicted | |
|---|---|---|---|
| | | Positive | Negative |
| Actual | Positive | TP | FN |
| | Negative | FP | TN |

**Confusion Matrix**

# 2. Precision and Recall




In a classification task, the **precision** for a class is the number of true positives divided by the total number of elements labeled as belonging to the positive class

$$\text{Precision} = \frac{TP}{(TP+FP)}$$

**Recall** is defined as the number of true positives divided by the total number of elements that actually belong to the positive class.

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

# 3. F1 Score

The F1 score is the harmonic mean of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

F1 Score might be a better measure to use if we need to seek a balance between Precision and Recall AND there is an uneven class distribution.

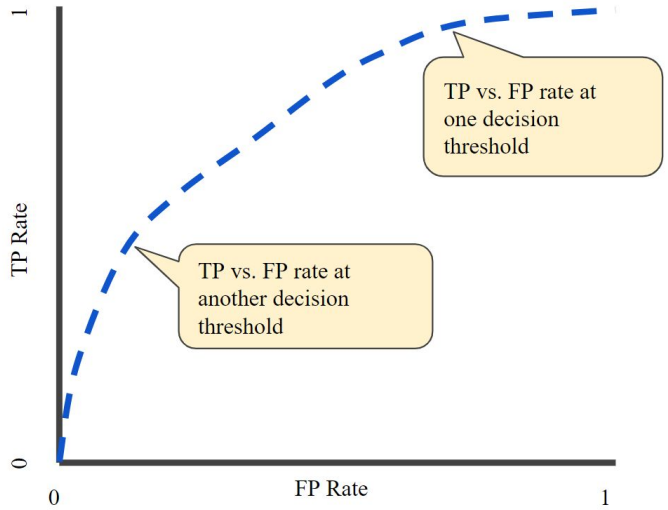
# 4. ROC Curve

A **receiver operating characteristic** curve

The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold

$$\text{True Positive Rate (TPR)} = \frac{TP}{(TP+FN)}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{(FP+TN)}$$



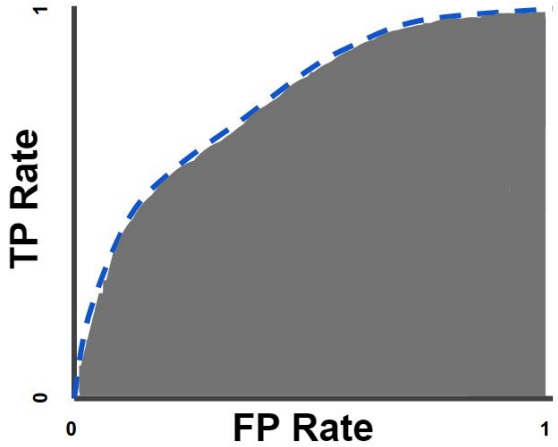
# 5. AUC

AUC stands for "Area under the ROC Curve."

It measures the entire two-dimensional area underneath the entire ROC curve

AUC provides an aggregate measure of performance across all possible classification thresholds

It ranges between 0 and 1



# HANDS-ON LABS

# Suggested Readings

1. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow - O'Reilley
2. Machine Learning Refined - Jeremy Watts
3.

# References


1. https://heartbeat.fritz.ai/
2. https://towardsdatascience.com/
3. https://www.kdnuggets.com/
4. Machine Learning Refined - Jeremy Watts
5. https://www.geeksforgeeks.org/
6.