

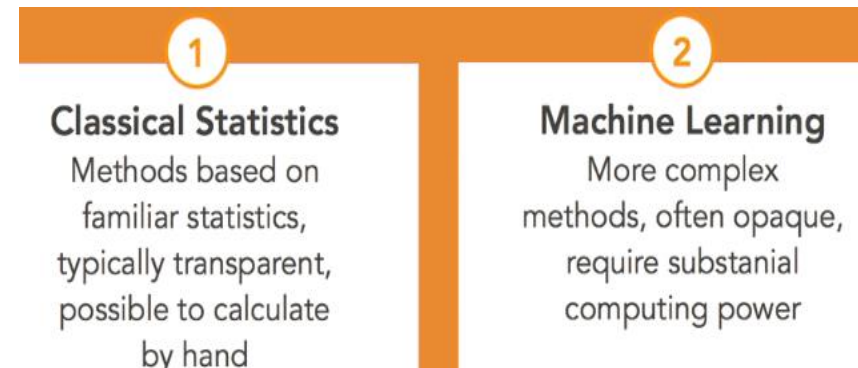
# Big Data Analysis with Machine Learning (I)



# Big data analysis techniques

## Big data analysis often blends

- traditional statistical data analysis approaches
  - **A/B testing** (compare two versions of an element to determine which one is superior based on a pre-defined metric)
  - **Correlation** (determine whether two variables are related to each other)
  - **Regression** (explore how a dependent variable is related to an independent variable)
- computational data analysis approach
  - **Data mining** (discover hidden or unknown patterns in datasets)
  - **Machine learning**



# Outline

- ML Concept
- ML steps
- ML algorithms
  - Supervised learning (classification)
- Big data ML
  - Apache Spark ML

# ML - Concept

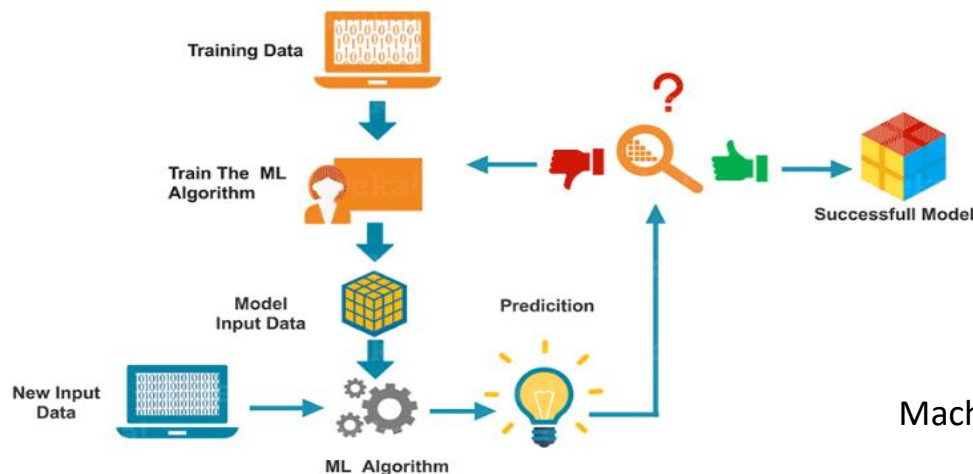
---

Machine learning is a field of computer science that gives computer systems the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed.

- Samuel, A. L. (1959): “*Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort*”
- Tom M. Mitchell (1997) provided a widely quoted, more formal definition of the algorithms studied in the machine learning field: “*A computer program is said to learn from experience  $E$  with respect to some class of tasks  $T$  and performance measure  $P$  if its performance at tasks in  $T$ , as measured by  $P$ , improves with experience  $E$* ”

# ML - Concept

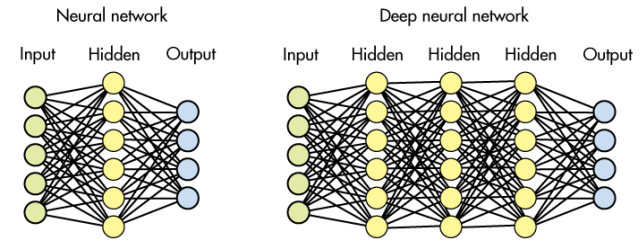
- Machine learning is to create an model / algorithm to predict answers (with maximum accuracy), via a process termed as **training**.
- Training data need to be collected in order to train a model.
- The trained model/algorithm can be used to predict answers for new data/examples



Machine learning process

# ML - Concept

## Deep learning



- Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning.
- Deep learning models are loosely related to information processing and communication patterns in a **biological nervous system**.
- Deep learning architectures such as **deep neural networks, deep belief networks and recurrent neural networks** have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics and drug design.



# ML - Concept

---

## Application

- Speech recognition ([demo](#))
- Autonomous car ([demo](#))
- Computer vision ([demo](#))
- Unmanned aerial vehicle ([demo](#))
- Board game
- Recommender system
- Face detection
- Object detection and recognition
- Image segmentation
- Multimedia event detection
- Economical and commercial usage
- ...



# ML - Steps

---

## 1. Gathering data

- Investigate the problem, and understand the domain.
- Identify the data sources required for the project
- Collect data, or datasets currently available and those that can be purchased or otherwise acquired
- Review the raw data: understanding the interdependencies among the data attributes, and become familiar with the content of the data, its quality, and its limitations.

# ML - Steps

---

## 2. Data preparation / data pre-processing

- Data Preprocessing is an important step in which mostly aims to improve raw data quality. It can be addressed by:
  - Data Integration (integrate various data sources)
  - Data exploration (Use summary statistics to spot issues in data, visualize data, etc)
  - Data cleaning (Fill missing data, smooth out noisy data, correct inconsistency in data, etc.)
  - Data transformation (transform data into different scales or domains, rescale/adjust feature values)
- Data now has to be split into two parts. The first part (i.e. **Training data**) that is used in training our model, will be the majority of the dataset and the second (i.e. **Testing data**) will be used for the evaluation of the trained model's performance.

# ML - Steps

---

## 3. Choosing a model

- Choosing a model among the many that researchers and data scientists have created over the years. Make the choice of the right one that should get the job done

## 4. Training

- The training data is used to incrementally improve the model's ability to predict.

## 5. Evaluation

- Evaluation is to test the model against testing data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

# ML - Steps

---

## 6. Parameter turning

- Once evaluation is done, it's possible to see if you can further improve the training by tuning some parameters.
- There were a few parameters that are implicitly assumed when we did the training, and now is a good time to go back and test those assumptions and try other values.

## 7. Prediction

- Finally use the model to predict the outcome.



Google Cloud  
AI Adventures



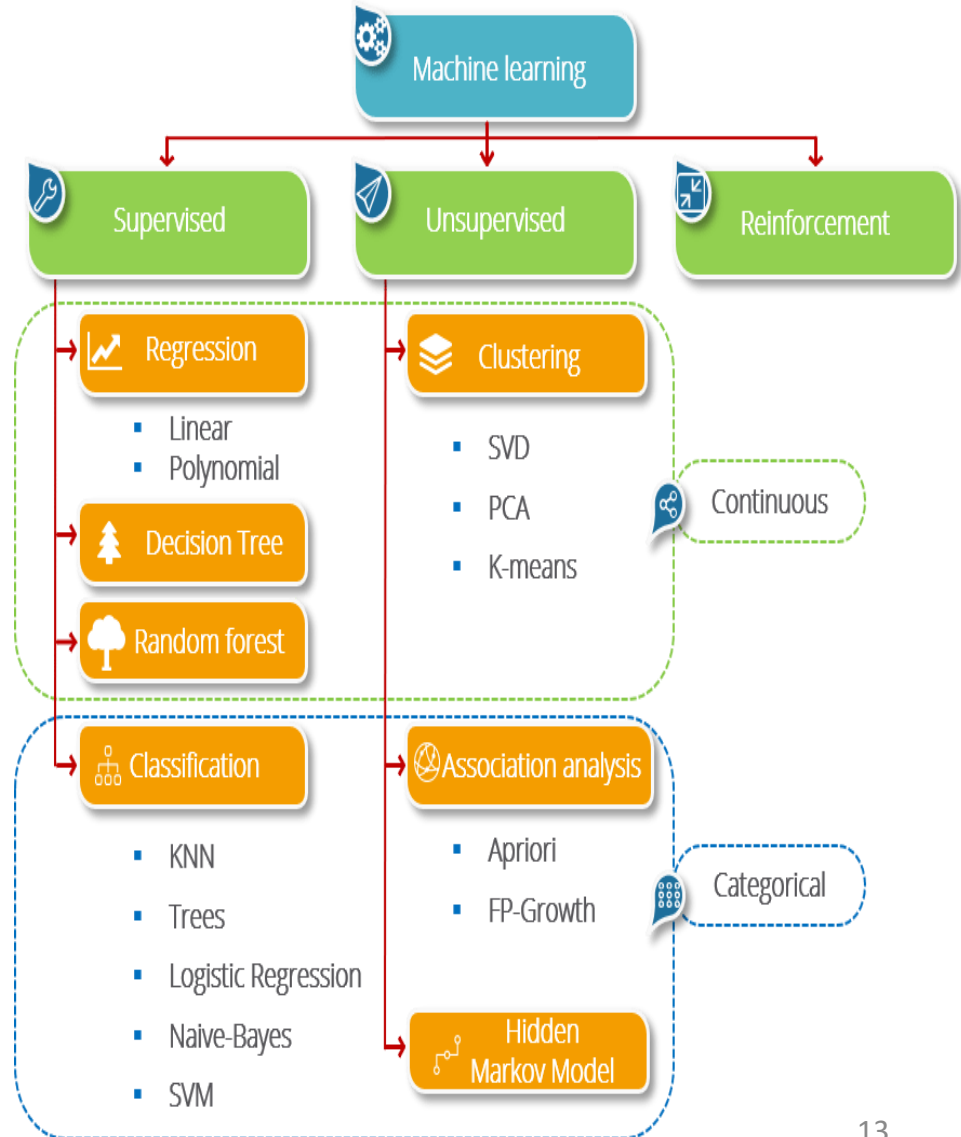
Presents:

7 Steps of Machine Learning

# ML - Algorithms

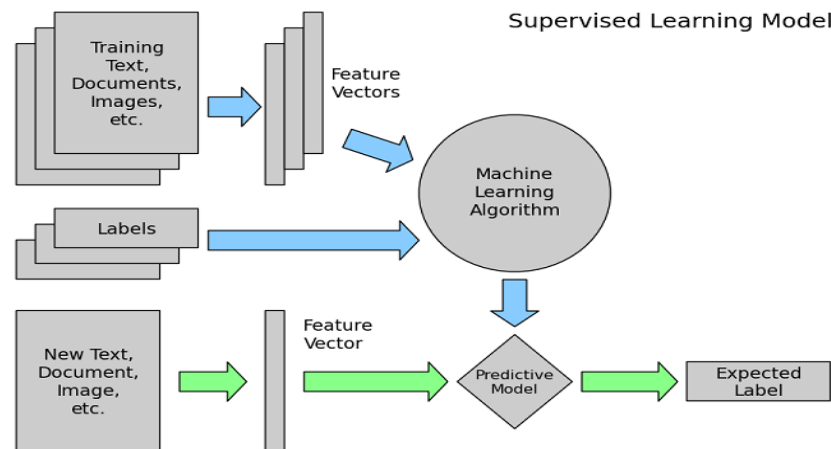
ML algorithms can be broadly classified into these categories:

- Supervised learning
- Unsupervised learning
- Reinforcement



# Supervised learning

- A supervised learning algorithm analyzes the **labelled training dataset** and produces an inferred function, which can be used for predicting new examples.
- Data for which you already know the target answer is called **labeled** dataset



# Supervised learning

---

## Supervised learning steps

- I. **Determine the type of training examples.** Firstly the user should decide what kind of data is to be used as a training set.
  
- II. **Gather a training dataset.** The **training dataset** (to train ML models/algorithm to find pattern) needs to be representative of the real-world use of the function. Thus, a set of input objects is gathered and corresponding outputs are also gathered, either from human experts or from measurements.

# Supervised learning

---

## Supervised learning steps

### III. Determine the input feature representation of the learned function.

- The accuracy of the learned function depends strongly on how the input object is represented.
- Typically, the input object is transformed into a **feature vector**, which contains a number of features that are descriptive of the object. The number of features should not be too large, because of the curse of dimensionality; but should contain enough information to accurately predict the output.















# Supervised learning

## Supervised learning steps

- **Feature**: one attribute/property of the objects we are analyzing
- **Feature vector**: the set of the features an object has
- **Label**: target attribute / property of an object
- **Model/Algorithm**: the formula used to make the prediction

Feature vector

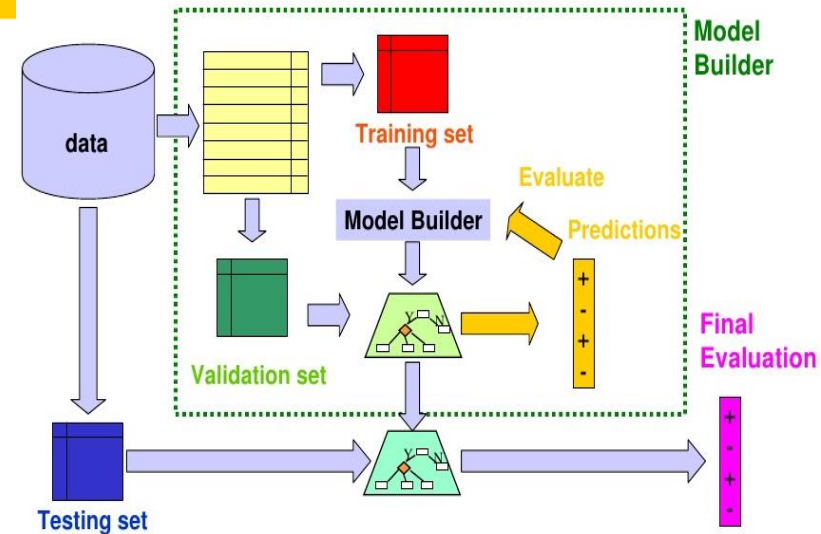
| Instance  | Shape   | Colour | Label      |
|---|---|--------|------------|
|    |    | red    | Apple      |
|    |    | green  | Apple      |
|    |    | green  | Pear       |
|   |   | yellow | Banana     |
|  |  | green  | Banana     |
|  |  | green  | Watermelon |







Training dataset/ Training examples ..

# Supervised learning

## Supervised learning steps:

- IV. Determine the structure of the learned function and corresponding learning algorithm
- V. **Complete the design.** Run the learning algorithm on the gathered training set. Some supervised learning algorithms require the user to determine certain control parameters. These parameters may be adjusted by optimizing performance on a subset (called a **validation dataset**) of the training dataset, or via cross-validation
- VI. **Evaluate the accuracy of the learned function.** After parameter adjustment and learning, the performance of the resulting function should be measured on a **testing dataset** (to evaluate the predictive quality of the trained model/algorithm) that is separate from the training set.



| Instance  | Shape   | Colour | Label |
|---|---|--------|-------|
|   |   | yellow | ?     |
|  |  | green  | ?     |
|  |  | green  | ?     |

Testing dataset/ Testing examples

# Supervised learning

---

## Supervised learning algorithms

- Supervised machine learning algorithms can be further divided into two subgroups: **Classification** and **Regression**.

**Classification**: identifying to which of a set of known categories (i.e. target labels are a finite set of **discrete categories**) a new instance belongs.

- For example, if an ML algorithm classifies set of fruits into “apple”, “pear”, “banana”, etc., then it is classification algorithm.
- Examples of Classification algorithms:
  - Decision tree
  - Support Vector machines (SVM)
  - K-nearest neighbors (KNN)
  - Naive Bayes classifier
  - Neural networks
  - ...

# Supervised learning

## Supervised learning algorithms

### Classification – Example Applications

#### Direct Marketing

- Goal: Reduce mailing cost by only targeting consumers likely to buy a new product.
- Approach:
  - Use the data for a similar product introduced before.
  - We know which customers decided to buy and which decided otherwise. This {buy, don't buy} decision forms the **feature vector**.
  - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
  - Use this information as input attributes to learn a classifier model.

#### Fraud Detection

- Goal: Predict fraudulent cases in credit card transactions
- Approach:
  - Use credit card transactions (When a customer buys, what he buys, how often he pays on time, etc.) and the info on its account-holder as attributes
  - Label past transactions as fraud or fair transactions. This forms the feature vector.
  - Learn a model for the class of the transactions.
  - Use this model to detect fraud by observing credit card transactions on an account.

# Supervised learning

## Supervised learning algorithms

- Supervised machine learning algorithms can be further divided into two subgroups: **Classification** and **Regression**.

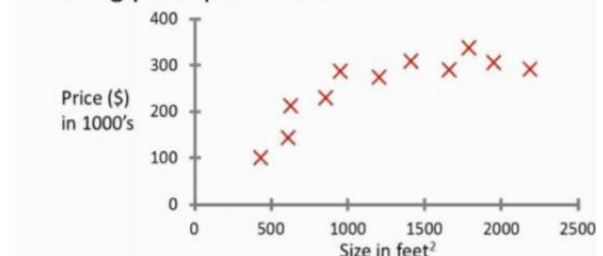
**Regression**: modelling and analyzing the relationship between a target variable (**continuous numeric values**) and one or more independent variables. They are used in primarily for forecasting.

### — Examples of Regression algorithm

- Linear regression
- Non-linear regression
- Logistic regression
- ...

| Instance | Area (Square meter) | Catchment area (0-10) | Property Price (£k) |
|----------|---------------------|-----------------------|---------------------|
| 1        | 100                 | 8                     | 1000                |
| 2        | 120                 | 9                     | 1300                |
| 3        | 60                  | 6                     | 800                 |
| ...      | ...                 | ...                   | ...                 |
| N        | 110                 | 6                     | ?                   |

Housing price prediction.



# Supervised learning algorithms

- Classification
  - K-NN
  - Naïve Bayes
  - Decision Tree
- Regression
  - General linear regression
  - Generalized linear regression

# Classification

---

**Classification:** identifying to which of a set of known categories (i.e. target labels are a finite set of **discrete categories**) a new instance belongs.

— Examples of Classification algorithms:

- Decision tree
- K-nearest neighbors (KNN)
- Naive Bayes classifier
- Support Vector Machines
- Neural networks
- ...



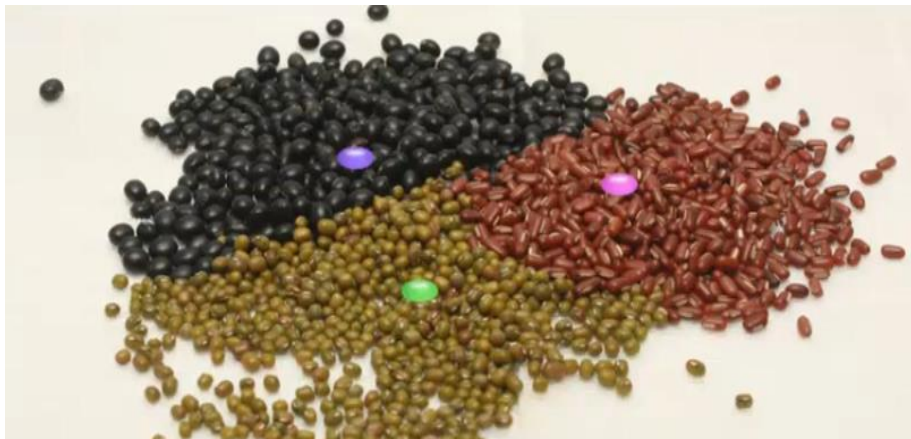
Classification goals

# Classification

---

## K-Nearest Neighbors (k-NN)

- Cover & Hart (1967) firstly proposed this algorithm.
- A type of **Instance-based learning** (compares new problem instances with instances seen in training) or **Lazy learning** (generalization of the training data is delayed until a query is made to the system)



To decide the class of a new instance by looking at the nearest neighbours



# Classification

## K-Nearest Neighbors (k-NN)

### Algorithm

- The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

| Movie Title     | # of Fights | # of Kisses | Movie Type |
|-----------------|-------------|-------------|------------|
| California man  | 3           | 104         | Romance    |
| He is not...    | 2           | 100         | Romance    |
| Beatiful woman  | 1           | 81          | Romance    |
| Kevin Longblade | 101         | 10          | Action     |
| Roho...         | 99          | 5           | ...        |
| ...             | ...         | ...         | ...        |
| xxx             | 18          | 90          | ?          |



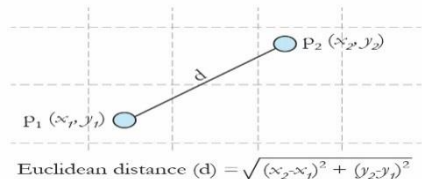
| Point(X, Y) | X   | Y   | Label   |
|-------------|-----|-----|---------|
| A           | 3   | 104 | Romance |
| B           | 2   | 100 | Romance |
| C           | 1   | 81  | Romance |
| D           | 101 | 10  | Action  |
| E           | 99  | 5   | ...     |
| ...         | ... | ... | ...     |
| N           | 18  | 90  | ?       |

# Classification

## K-Nearest Neighbors (k-NN)

### Algorithm

- In the classification phase,  $k$  is a user-defined constant (i.e.. # of nearest neighbors), and an unlabeled instance (testing data) is classified by assigning the label which is most frequent (majority-voting) among the  $k$  training samples **nearest** to that instance.
- A commonly used distance metric is Euclidean distance



$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.$$

$\text{Dist}(A, N) = 20$     $\text{Dist}(B, N) = 18$     $\text{Dist}(C, N) = 19$     $\text{Dist}(D, N) = 115$

- If  $K = 3$ , then the nearest 3 points are A, B, C (all “Romance”). So N is “Romance”
- If  $K =$  other number, and N is classified to the class that the majority of nearest neighbours belong to

# Classification

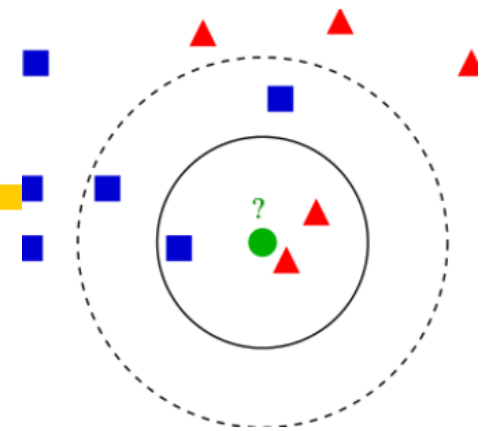
## K-Nearest Neighbors (k-NN)

### Advantages

- Simple, easy to implement
- Larger values of  $k$  reduces effect of the noise on the classification

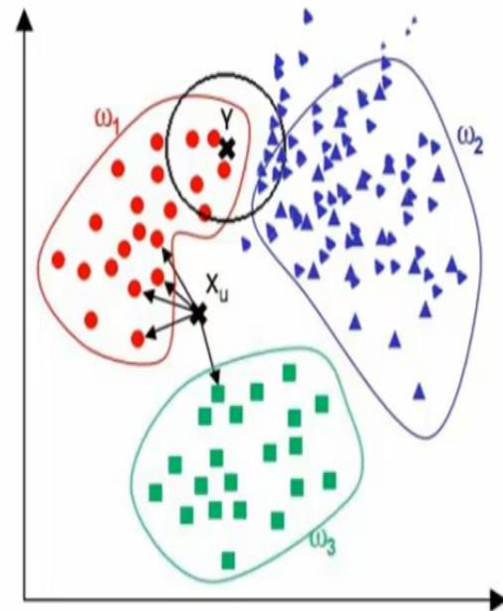
### Drawbacks

- Large memory to store all the data
- High complexity
- Problem occurs when class distribution is skewed, i.e. examples of a more frequent class tend to dominate the prediction of new example



The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles.

- If  $k = 3$  (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle.
- If  $k = 5$  (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).



# Classification

---

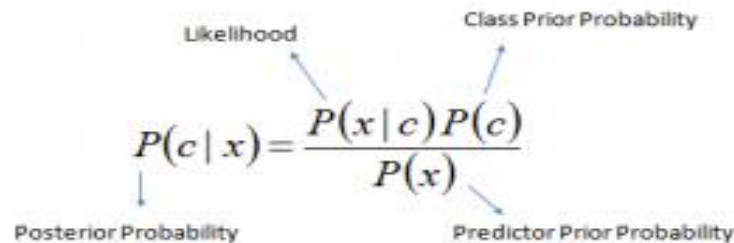
## K-Nearest Neighbors (k-NN)



# Classification

## Naïve Bayes

- Naive Bayes (NB) has been studied extensively since the 1950s.
- Naive Bayes is a classification technique based on Bayes' theorem (conditional probability)



The diagram shows the Bayes' theorem formula  $P(c|x) = \frac{P(x|c)P(c)}{P(x)}$  with arrows pointing from labels to the terms in the formula. 'Likelihood' points to  $P(x|c)$ , 'Class Prior Probability' points to  $P(c)$ , 'Posterior Probability' points to  $P(c|x)$ , and 'Predictor Prior Probability' points to  $P(x)$ .

$$P(c | \mathbf{X}) = P(x_1 | c) \times P(x_2 | c) \times \dots \times P(x_n | c) \times P(c)$$

- $P(c|x)$  is the posterior probability of *class (target)* given *feature (attribute)*.
- $P(c)$  is the prior probability of *class*.
- $P(x|c)$  is the likelihood which is the probability of *feature(attribute)* given *class*.
- $P(x)$  is the prior probability of *feature(attribute)*

# Classification

---

## Naïve Bayes

### Algorithm

- NB algorithm constructs tables of probabilities that are used to estimate the likelihood that new examples belong to various classes.
- NB algorithm has an assumption that all of the features in a dataset are equally important and unrelated to each other (i.e. each attribute is conditionally independent of every other attribute given a class label)
- The input features are generally categorical.
- NB classifier is often used for text classification.

# Classification

## Naïve Bayes

### Algorithm

| 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  |       |      |

1. Convert the data set to frequency table
2. Create Likelihood table by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.
3. Use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

$$\begin{aligned} P(\text{Play} \mid \text{Sunny}) &= P(\text{Sunny} \mid \text{Play}) * P(\text{Play}) / P(\text{Sunny}) \\ &= (3/9) * (9/14) * / (5/14) = 0.60 \end{aligned}$$

The probability that players will play given that weather is sunny is 0.60

# Classification

---

## Naïve Bayes

### Advantages

- Simple, fast and effective
- Does well with noisy and missing data
- Requires relatively few examples for training, but also works well with very large numbers of examples

### Drawbacks

- Relies on an often-faulty assumption of equally important and independent features.
- Not ideal for datasets with large numbers of numeric features
- Estimated probabilities are less reliable than the predicated classes.



# Classification

## Naïve Bayes



Supervised Learning in R: Classification

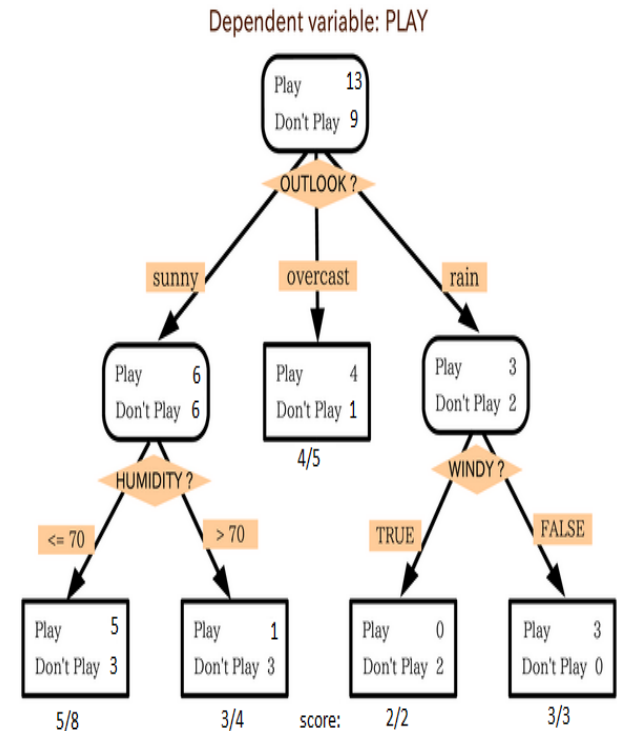
SUPERVISED LEARNING IN R: CLASSIFICATION

## Understanding Bayesian methods

# Classification

## Decision Tree

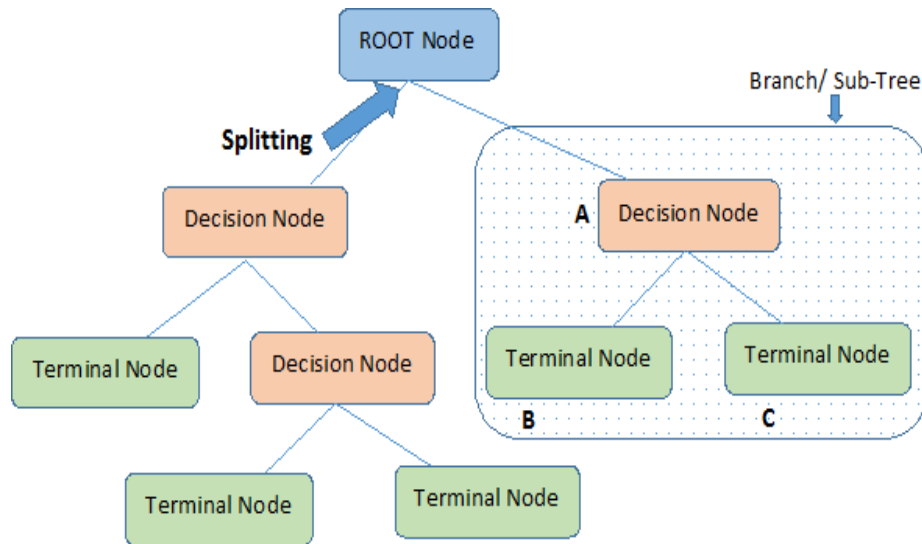
- Decision tree is a type of supervised learning algorithm that is mostly used in classification problems.
- Decision trees are built using a heuristic called **recursive partitioning**. This approach is generally known as **divide and conquer** because it splits the population or sample into two or more smaller homogeneous sets (or sub-populations) based on features in input variables.
  - Beginning at the root node, which represents the entire dataset, the algorithm chooses a feature that is the most predictive of the target class.
  - The examples are then partitioned into groups of distinct values of this feature; this decision forms the first set of tree branches.
  - The algorithm continues to divide-and-conquer the nodes, choosing the best candidate feature each time until a stopping criterion is reached.



Population is classified into four different groups based on multiple features to identify 'if they will play or not'. To split the population into different groups.

# Classification

## Decision Tree



**Note:-** A is parent node of B and C.

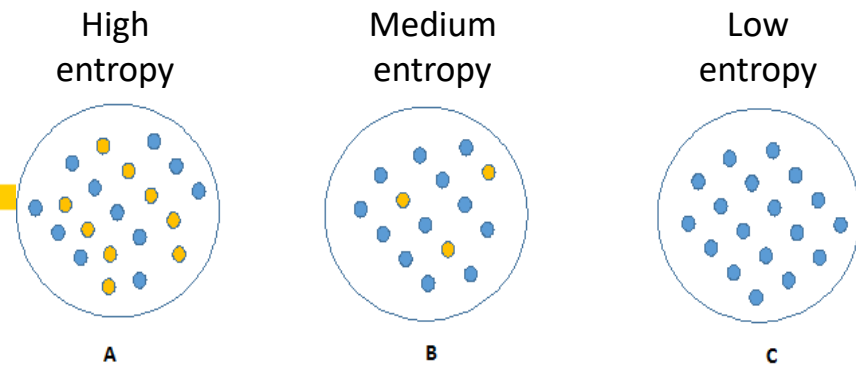
- **Root Node:** It represents entire population and this further gets divided into two or more homogeneous sets.
- **Splitting:** It is a process of dividing a node into two or more sub-nodes.
- **Decision Node:** When a sub-node splits into further sub-nodes, then it is called decision node.
- **Leaf/ Terminal Node:** Nodes do not split is called Leaf or Terminal node.
- **Pruning:** When we remove sub-nodes of a decision node, this process is called pruning. You can say opposite process of splitting.
- **Branch / Sub-Tree:** A sub section of entire tree is called branch or sub-tree.
- **Parent and Child Node:** A node, which is divided into sub-nodes is called parent node of sub-nodes where as sub-nodes are the child of parent node.

- Decision trees can become far more complex with many nodes, branches and leaves.
- There are numerous implementations of decision tress, but one of the most well-known is **C5.0** algorithm which is the industry standard for producing decision tress.

# Classification

## Decision Tree

### Entropy



- The concept of Information **Entropy** was introduced by Claude Shannon in his 1948 paper "A Mathematical Theory of Communication".
- Entropy is a measure of the amount of information produced by a stochastic source of data, which indicates level of uncertainty. A larger entropy value indicates a higher level of uncertainty or diversity, implying lower purity or homogeneity.

$$\text{Entropy}(S) = \sum_{i=1}^c -p_i \log_2(p_i)$$

For a given dataset  $S$ , the term  $c$  refers to the number of different classes, and  $p_i$  refer to the proportion/probability of values falling into level  $i$ .

Entropy has minimum value of 0 indicating a completely pure/homogenous dataset, while 1 indicates the maximum amount of disorder/impurity.

# Classification

Training dataset for  
decision tree

## Decision Tree

### Entropy

In the training dataset, two  
classes (Yes/No)

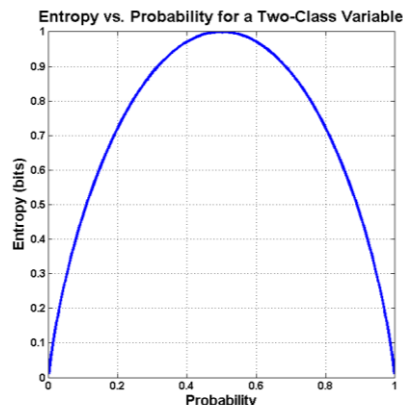
$$p_{\text{Yes}} = 9/14, p_{\text{No}} = 5/14$$

$$\text{Entropy (S)} = \sum_{i=1}^c -p_i \log_2(p_i)$$

$$= -p_1 \log_2(p_1) - p_2 \log_2(p_2)$$

$$= -9/14 * \log_2(9/14) - 5/14 * \log_2(5/14)$$

$$= 0.940 \text{ bits}$$



| ID | Age    | Income | Student | Credit-rating | Class:<br>buy_laptop |
|----|--------|--------|---------|---------------|----------------------|
| 1  | Youth  | High   | No      | Fair          | No                   |
| 2  | Youth  | High   | No      | Excellent     | No                   |
| 3  | Middle | High   | No      | Fair          | Yes                  |
| 4  | Senior | Medium | No      | Fair          | Yes                  |
| 5  | Senior | Low    | Yes     | Fair          | Yes                  |
| 6  | Senior | Low    | Yes     | Excellent     | No                   |
| 7  | Middle | Low    | Yes     | Excellent     | Yes                  |
| 8  | Youth  | Medium | No      | Fair          | No                   |
| 9  | Youth  | Low    | Yes     | Fair          | Yes                  |
| 10 | Senior | Medium | Yes     | Fair          | Yes                  |
| 11 | Youth  | Medium | Yes     | Excellent     | Yes                  |
| 12 | Middle | Medium | No      | Excellent     | Yes                  |
| 13 | Middle | High   | Yes     | Fair          | Yes                  |
| 14 | Senior | Medium | No      | Excellent     | No                   |

# Classification

---

## Decision Tree

### Tree splitting

Decision trees employ algorithms to decide which feature will be used to split a node in two or more sub-nodes. The decision of making strategic splits heavily affects a tree's accuracy.

- Decision tree splits the nodes on all available features and then selects the most significant feature to split the node which results in most homogeneous sub-nodes (i.e. as distinct sub-groups as possible).
- The most commonly used splitting algorithms include: **Information Gain**, Gini Index, Chi-Square, etc.

# Classification

## Decision Tree

### Tree splitting – Information Gain

- Use entropy to calculate the change in homogeneity resulting from a split on each possible feature.
- The Information Gain for a feature  $F$  is calculated as the difference between entropy in the node before the split ( $S_1$ ) and the sub-nodes resulting from the split ( $S_2$ ):

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

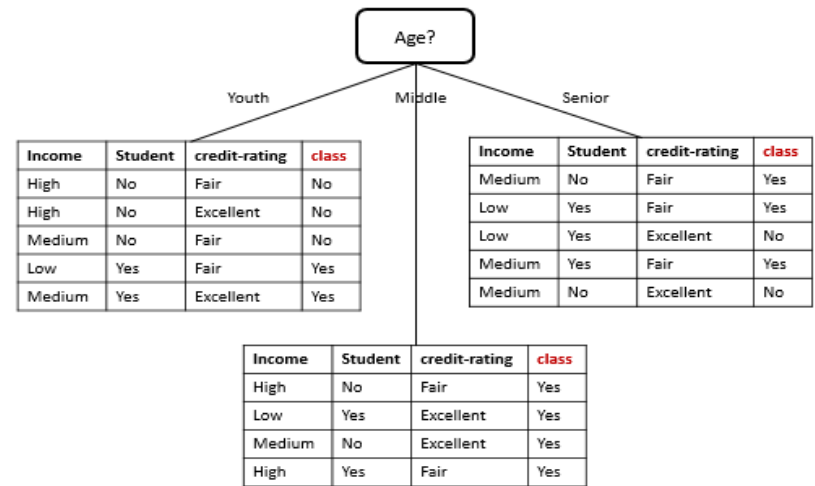
- The entropy resulting from a split (i.e.  $\text{Entropy}(S_2)$ ) is the sum of entropy of each of the  $n$  sub-nodes weighted by the proportion of examples falling into that sub-node ( $W_i$ )

$$\text{Entropy}(S_2) = \sum_{i=1}^n w_i \text{Entropy}(P_i)$$

# Classification

## Decision Tree

### Information Gain



- If split on the feature of 'age', then the entropy after splitting is calculated:

$$\begin{aligned}\text{Entropy}(\text{Split-on-Age}) &= \frac{5}{14} * (-\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5}) + \\ &\quad \frac{4}{14} * (-\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4}) + \\ &\quad \frac{5}{14} * (-\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5}) \\ &= 0.694 \text{ bits}\end{aligned}$$

$$\begin{aligned}\text{InfoGain}(\text{Age}) &= \text{Entropy}(\text{Before-split}) - \text{Entropy}(\text{Split-on-Age}) \\ &= 0.940 - 0.694 = 0.246 \text{ bit}\end{aligned}$$

Similarly to get  $\text{InfoGain}(\text{Income}) = 0.029$ ,  $\text{InfoGain}(\text{Student}) = 0.151$ ,  
 $\text{InfoGain}(\text{Credit\_rating}) = 0.048$

- The higher the information gain, the better a feature is at creating homogeneous sub-nodes after a split on that feature.



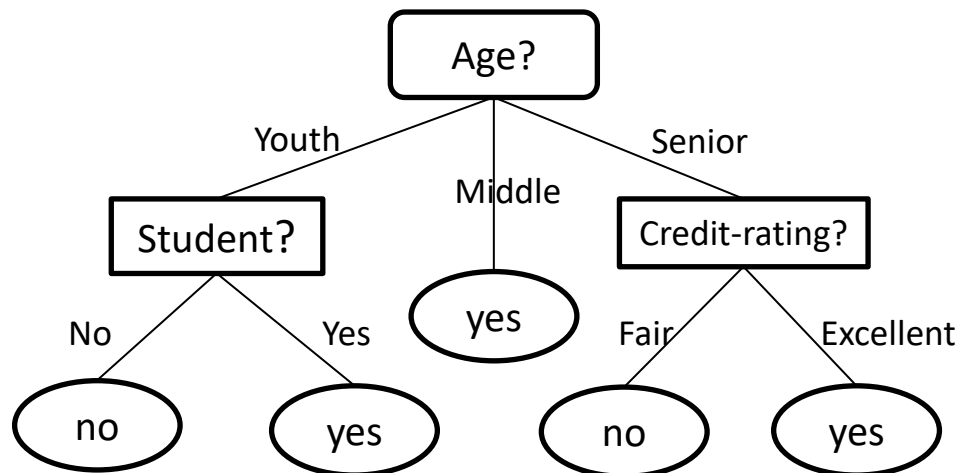
# Classification

## Decision Tree

### Information Gain

Stopping criteria for building the decision tree

- All (or nearly all) of the examples at the node have the same class
- There are no remaining features to distinguish among examples
- The tree has grown to a predefined size limit



# Classification

## Decision Tree

| Strengths  | Weaknesses   |
|--|--|
| <ul style="list-style-type: none"><li>• An all-purpose classifier that does well on most problems</li><li>• Highly-automatic learning process can handle numeric or nominal features, missing data</li><li>• Uses only the most important features</li><li>• Can be used on data with relatively few training examples or a very large number</li><li>• Results in a model that can be interpreted without a mathematical background (for relatively small trees)</li><li>• More efficient than other complex models</li></ul> | <ul style="list-style-type: none"><li>• Decision tree models are often biased toward splits on features having a large number of levels</li><li>• It is easy to overfit or underfit the model</li><li>• Can have trouble modeling some relationships due to reliance on axisparallel splits</li><li>• Small changes in training data can result in large changes to decision logic</li><li>• Large trees can be difficult to interpret and the decisions they make may seem counterintuitive</li></ul> |

# Classification

---

## Decision Tree



DataCamp



Machine Learning with Tree-Based Models in R

MACHINE LEARNING WITH TREE-BASED MODELS IN R

## **Introduction to classification trees**

# Classification

---

- Classification algorithms summary



***k*-Nearest  
Neighbors (*k*-NN)**



**Naive Bayes**



**Decision trees**



**Random forests**

Classification algorithms



# Big Data ML



Spark MLlib is a distributed ML framework that contains many algorithms and utilities.

ML algorithms include:

- Classification: logistic regression, naive Bayes,...
- Regression: generalized linear regression, survival regression,...
- Decision trees, random forests, and gradient-boosted trees
- Recommendation: alternating least squares (ALS)
- Clustering: K-means, Gaussian mixtures (GMMs),...
- Topic modeling: latent Dirichlet allocation (LDA)
- Frequent itemsets, association rules, and sequential pattern mining

ML workflow utilities include:

- Feature transformations: standardization, normalization, hashing,...
- ML Pipeline construction
- Model evaluation and hyper-parameter tuning
- ML persistence: saving and loading models and Pipelines

Other utilities include:

- Distributed linear algebra: SVD, PCA,...
- Statistics: summary statistics, hypothesis testing,...

# Big Data ML

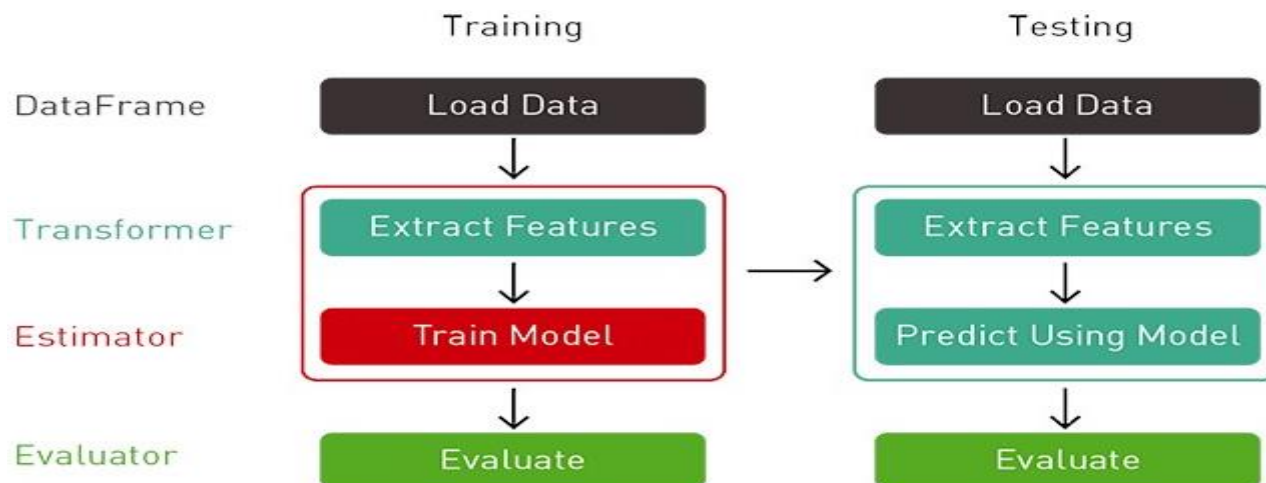


Spark MLlib is an umbrella term to refer to both machine learning library packages in Apache Spark.

- [spark.mllib](#) is the original machine learning API, based on the RDD API (which has been in maintenance mode since Spark 2.0, deprecated)
- [spark.ml](#) is the newer ML API from Spark 2.0, based on Data Frames:
  - The Spark SQL and the Dataset/DataFrame APIs provide ease of use, space efficiency, and performance gains. Having ML APIs built on top of DataFrames provides the scalability of partitioned data processing with the ease of SQL for data manipulation.
  - Spark ML provides a uniform set of high-level APIs, built on top of DataFrames with the goal of making machine learning scalable and easy.
  - We will focus on using the spark.ml package for machine learning in Apache Spark

# Big Data ML

- Spark MLlib is heavily based on scikit-learn's (single-node ML) ideas on **pipelines**.
  - The concept of pipelines is common across many ML frameworks as a way to organize a series of operations to apply to your data. In Spark MLlib, the Pipeline API provides a high-level API built on top of DataFrames to organize your machine learning workflow.

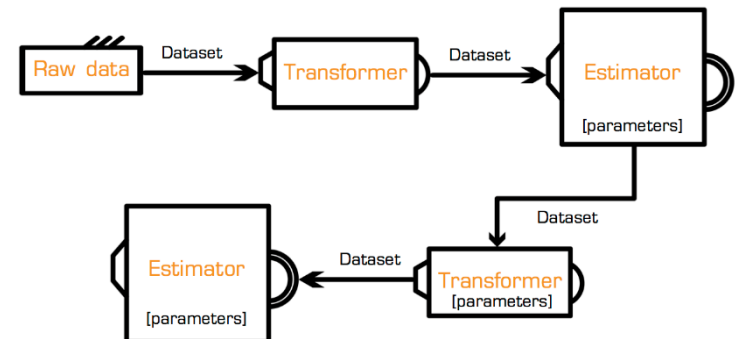
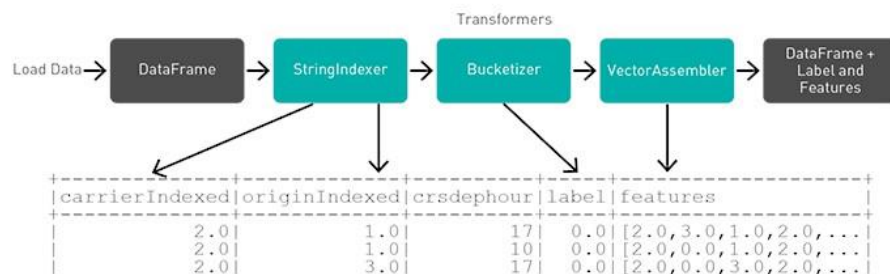


- The basic concepts/terminologies to build an end-to-end ML pipeline with Mllib:
  - **DataFrame:** Spark ML API uses DataFrame from Spark SQL as an ML dataset. A DataFrame could have different columns storing text, feature vectors, true labels, and predictions.
  - **Transformer:** Accepts a DataFrame as input, and returns a new DataFrame with one or more columns appended to it. A transformer applies rule-based transformations to either prepare data for model training or generate predictions using a trained Mllib model.
  - **Estimator:** An Estimator is an algorithm which can be fit on a DataFrame to produce a Transformer. E.g., a learning algorithm is an Estimator which trains on a DataFrame and produces a model
  - **Evaluator:** evaluation metrics for evaluating the performance of a machine learning model. It aims to estimate the generalization accuracy of a model on the future (unseen/out-of-sample) data.



# Big Data ML

- The basic concepts/terminologies to build an end-to-end ML pipeline with Mllib:
  - **Pipeline:** Organizes a series of transformers and estimators into a single model. While pipelines themselves are estimators, the output of `pipeline.fit()` returns a `PipelineModel`, a transformer.



Spark ML pipeline

# Big Data ML



An example of classification algorithms using Spark MLlib –  
**Decision Tree**

## Data Preparation

MACHINE LEARNING WITH PYSARK



## Decision Tree

MACHINE LEARNING WITH PYSARK



Please see the Python code in the tutorial on Blackboard. More examples of classifications using Spark MLlib can be found at: [Machine Learning Library \(MLlib\) Guide](#)