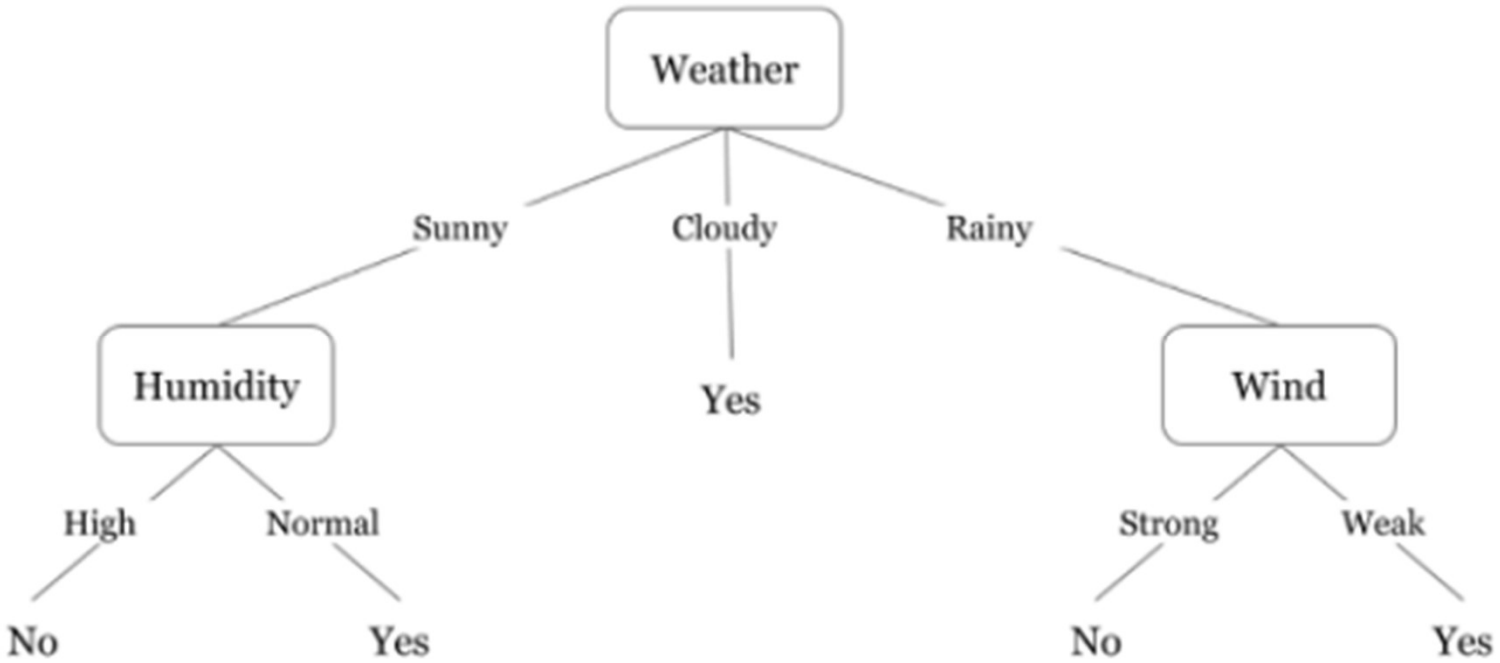
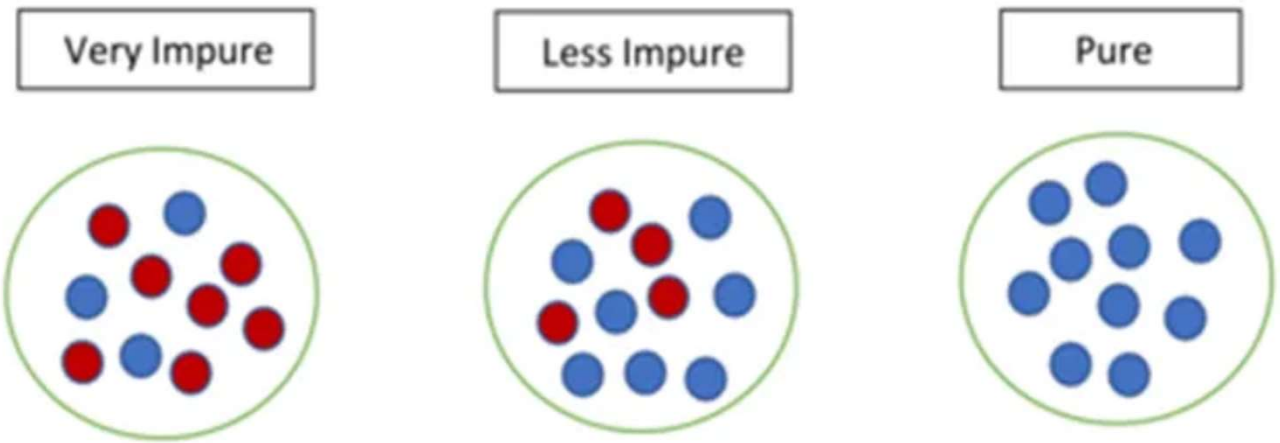
DTREE BAYES PCA SVM

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No


Weather
Sunny
Cloudy
Rainy
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Entropy

- Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. In a decision tree, the goal is to decrease the entropy of the dataset by creating more pure subsets of data. Since entropy is a measure of impurity, by decreasing the entropy, we are increasing the purity of the data.


Very Impure
Less Impure
Pure

$$E = -\sum_{i=1}^{N} p_i log_2 p_i$$

Pi is the probability of randomly selecting an example in class i. The logarithm of fractions gives a negative value, and hence a '-' sign is used in the entropy formula to negate these negative values.

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
12000	18	No

$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$

$H(d) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$

$H(d) = 0.97$

Salary	Age	Purchase
34000	31	No
15000	25	No
69000	57	Yes
25000	21	No
32000	28	No

$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$

$H(d) = -1/5 \log_2(1/5) - 4/5 \log_2(4/5)$

$H(d) = 0.72$

Salary	Age	Purchase
20000	21	No
10000	45	No
60000	27	No
15000	31	No
12000	18	No

$H(d) = -P_y \log_2(P_y) - P_n \log_2(P_n)$

$H(d) = -0/5 \log_2(0/5) - 5/5 \log_2(5/5)$

$H(d) = 0$

- When the entropy of a dataset is high, it means that the data is not pure, and the classes are not evenly distributed. On the other hand, when the entropy is low, it means that the data is pure, and the classes are evenly distributed.

Salary	Age	Purchase
20000	21	Yes
10000	45	No
60000	27	Yes
15000	31	No
30000	30	Maybe
12000	18	No
40000	40	Maybe
20000	20	Maybe

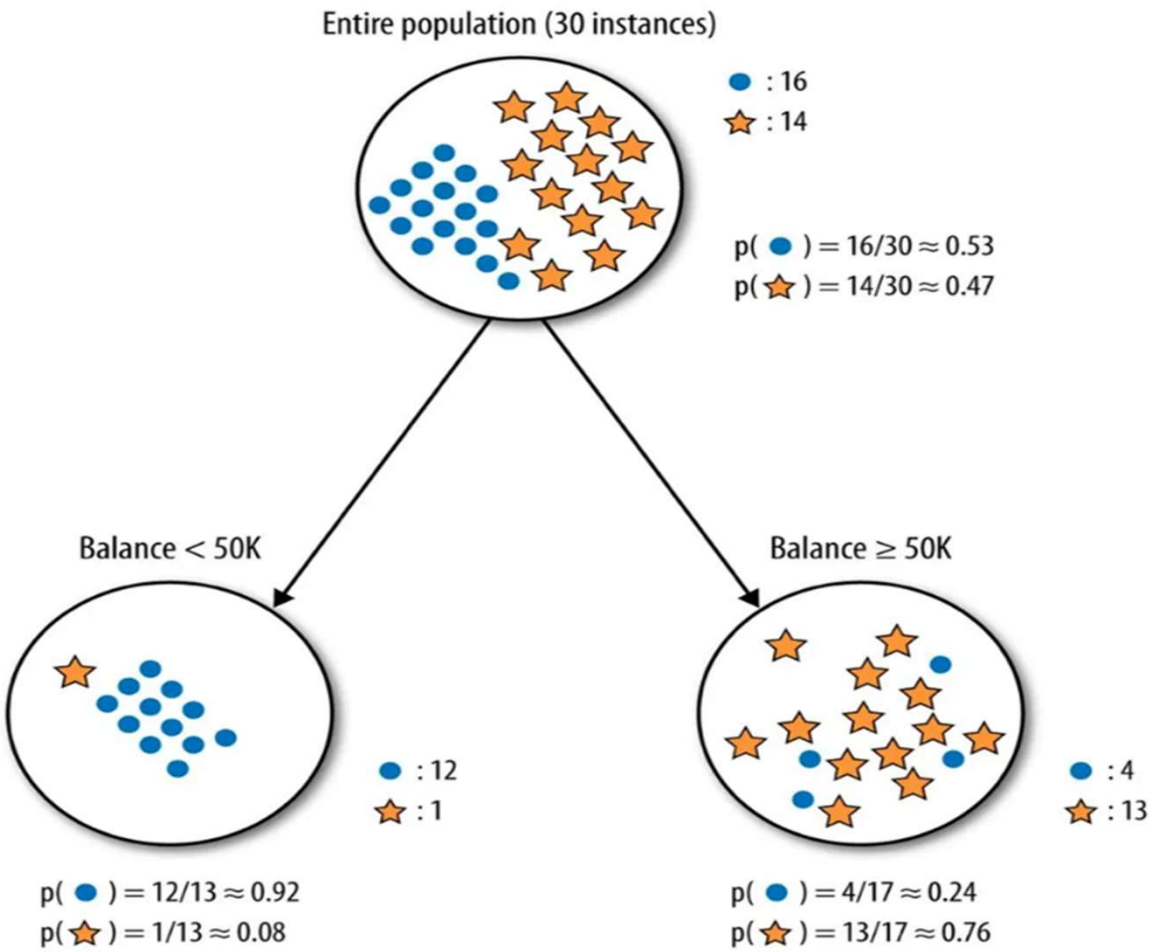
$$H(d) = - P_y \log_2(P_y) - P_n \log_2(P_n) - P_m \log_2(P_m)$$

$$H(d) = -2/8 \log_2(2/8) - 3/8 \log_2(3/8) - 3/8 \log_2(3/8)$$

$$H(d) = 1.56$$

Fig. Calculating entropy for a three class problem

Information gain = entropy (parent) – [weightes average] * entropy (children)


Entire population (30 instances)
: 16
: 14
p( ) = 16/30 ≈ 0.53
p( ) = 14/30 ≈ 0.47
Balance < 50K
: 12
: 1
p( ) = 12/13 ≈ 0.92
p( ) = 1/13 ≈ 0.08
Balance ≥ 50K
: 4
: 13
p( ) = 4/17 ≈ 0.24
p( ) = 13/17 ≈ 0.76

$$E(Parent) = -\frac{16}{30}\log_2\left(\frac{16}{30}\right) - \frac{14}{30}\log_2\left(\frac{14}{30}\right) \approx 0.99$$

Let's see how much uncertainty the tree can reduce by splitting on Balance.

$$E(Balance < 50K) = -\frac{12}{13}\log_2\left(\frac{12}{13}\right) - \frac{1}{13}\log_2\left(\frac{1}{13}\right) \approx 0.39$$

$$E(Balance > 50K) = -\frac{4}{17}\log_2\left(\frac{4}{17}\right) - \frac{13}{17}\log_2\left(\frac{13}{17}\right) \approx 0.79$$

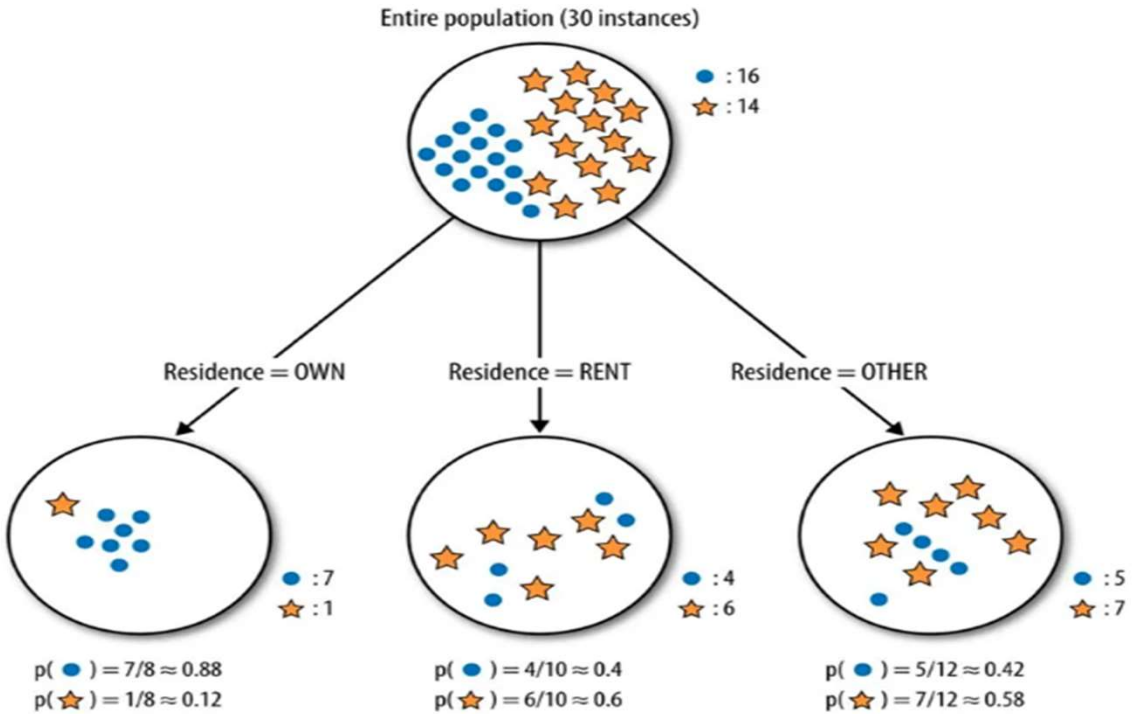
Weighted Average of entropy for each node:

$$\begin{aligned} E(Balance) &= \frac{13}{30} \times 0.39 + \frac{17}{30} \times 0.79 \\ &= 0.62 \end{aligned}$$

Information Gain:

$$\begin{aligned} IG(Parent, Balance) &= E(Parent) - E(Balance) \\ &= 0.99 - 0.62 \\ &= \mathbf{0.37} \end{aligned}$$

- Splitting on feature , “Balance” leads to an information gain of 0.37 on our target variable. Let’s do the same thing for the feature, “Residence” to see how it compares.



Splitting the tree on Residence gives us 3 child nodes.

$$E(\textit{Residence} = \textit{OWN}) = -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) \approx 0.54$$

$$E(\textit{Residence} = \textit{RENT}) = -\frac{4}{10}\log_2\left(\frac{4}{10}\right) - \frac{6}{10}\log_2\left(\frac{6}{10}\right) \approx 0.97$$

$$E(\textit{Residence} = \textit{OTHER}) = -\frac{5}{12}\log_2\left(\frac{5}{12}\right) - \frac{7}{12}\log_2\left(\frac{7}{12}\right) \approx 0.98$$

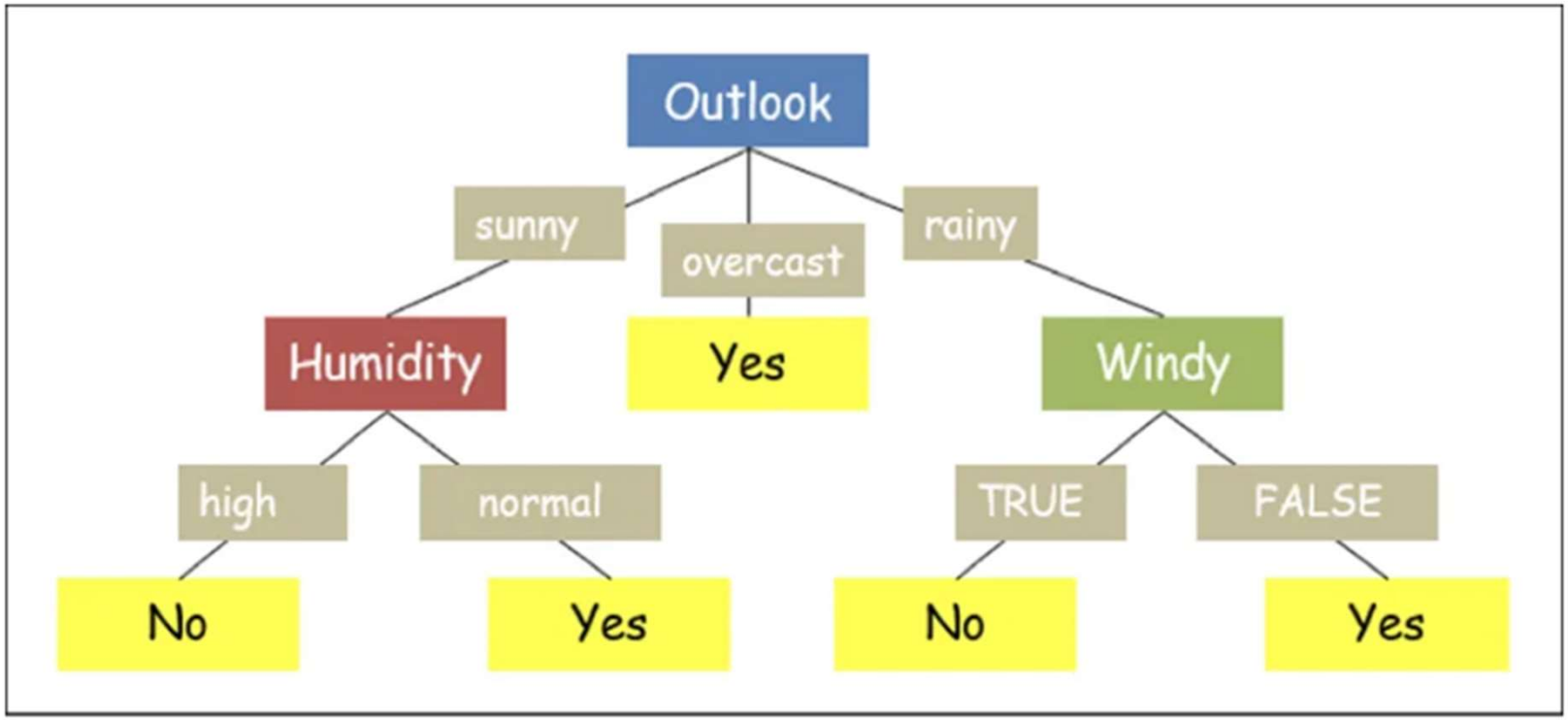
Weighted Average of entropies for each node:

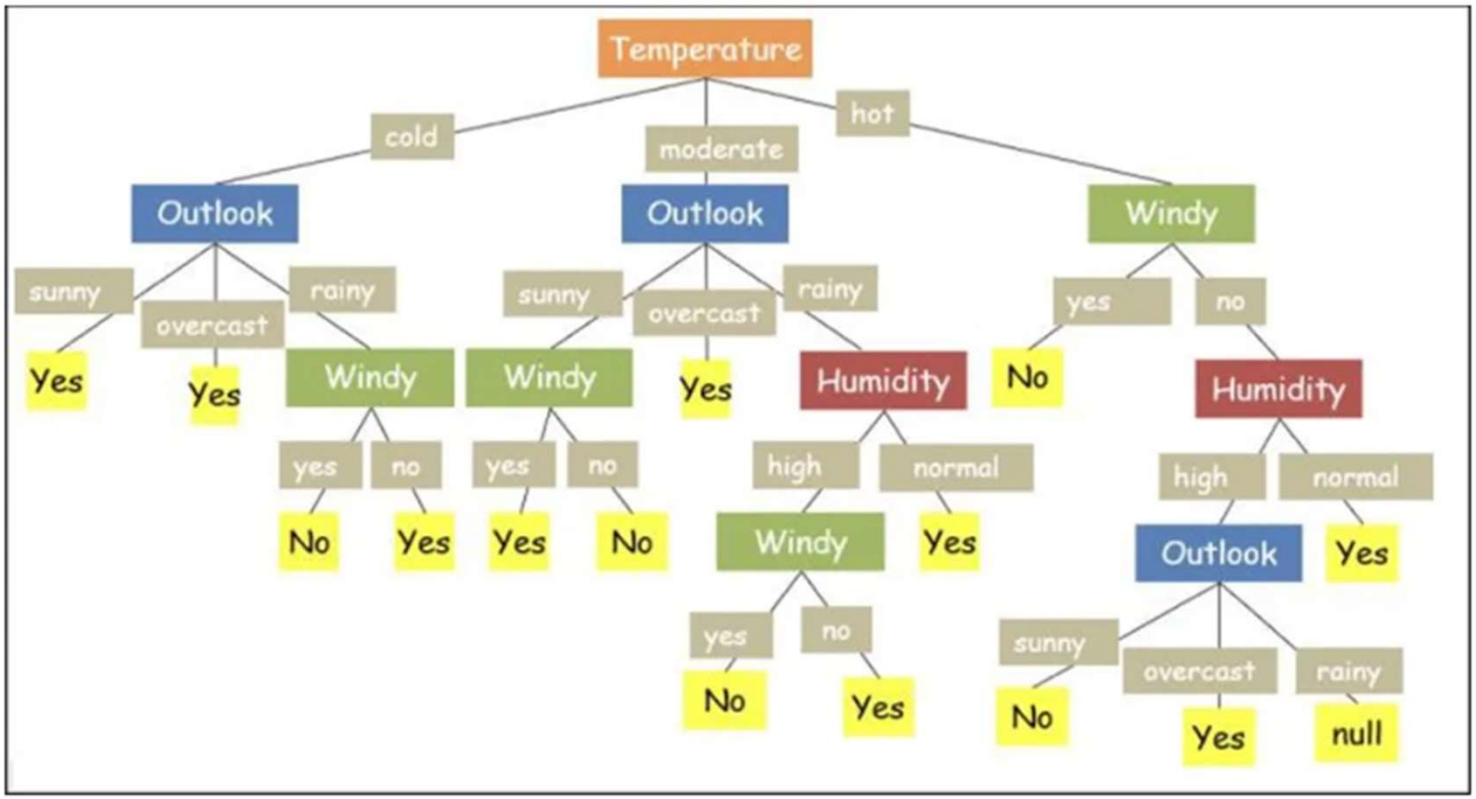
$$E(\textit{Residence}) = \frac{8}{30} \times 0.54 + \frac{10}{30} \times 0.97 + \frac{12}{30} \times 0.98 = 0.86$$

Information Gain:

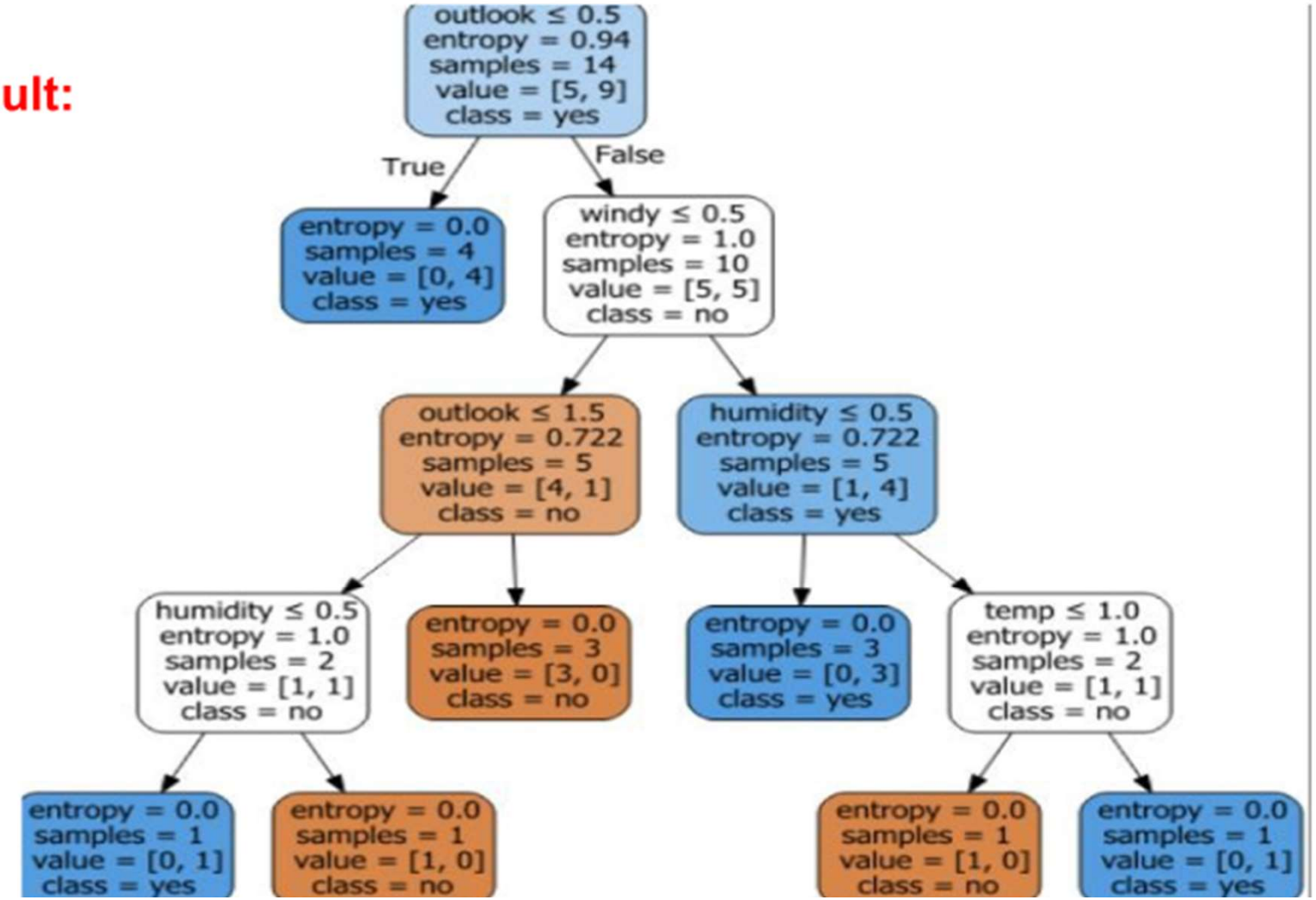
$$\begin{aligned} IG(\textit{Parent}, \textit{Residence}) &= E(\textit{Parent}) - E(\textit{Residence}) \\ &= 0.99 - 0.86 \\ &= 0.13 \end{aligned}$$

	outlook	temp	humidity	windy	play
2	sunny	hot	high	false	no
3	sunny	hot	high	true	no
4	overcast	hot	high	false	yes
5	rainy	mild	high	false	yes
6	rainy	cool	normal	false	yes
7	rainy	cool	normal	true	no
8	overcast	cool	normal	true	yes
9	sunny	mild	high	false	no
10	sunny	cool	normal	false	yes
11	rainy	mild	normal	false	yes
12	sunny	mild	normal	true	yes
13	overcast	mild	high	true	yes
14	overcast	hot	normal	false	yes
15	rainy	mild	high	true	no

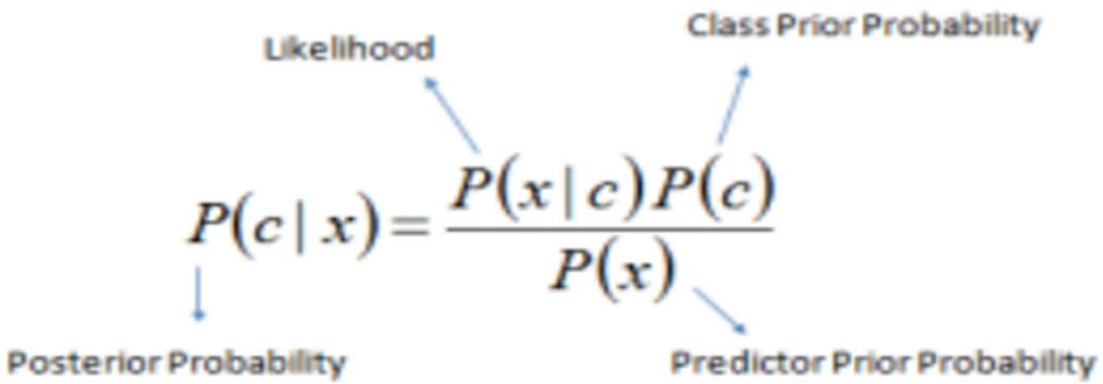

Outlook
sunny
overcast
rainy
Humidity
Yes
Windy
high
normal
TRUE
FALSE
No
Yes
No
Yes


Temperature
cold
moderate
hot
Outlook
Outlook
Windy
sunny
overcast
rainy
sunny
overcast
rainy
yes
no
Yes
Yes
Windy
Windy
Yes
Humidity
No
Humidity
yes
no
yes
no
high
normal
high
normal
No
Yes
Yes
No
Windy
Yes
Outlook
Yes
yes
no
sunny
overcast
rainy
No
Yes
No
Yes
null

Final Result:



Bayes theorem provides a way of computing posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



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

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

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

Frequency Table

		Stolen?	
		Yes	No
Color	Red	3	2
	Yellow	2	3

Likelihood Table

		Stolen?	
		P(Yes)	P(No)
Color	Red	3/5	2/5
	Yellow	2/5	3/5

Frequency and Likelihood tables of 'Color'

Frequency Table

		Stolen?	
		Yes	No
Type	Sports	4	2
	SUV	1	3

Likelihood Table

		Stolen?	
		P(Yes)	P(No)
Type	Sports	4/5	2/5
	SUV	1/5	3/5

$$P(Yes \mid X) = P(Red \mid Yes) * P(SUV \mid Yes) * P(Domestic \mid Yes) * P(Yes)$$

$$= 3/5 * 1/5 * 2/5 * 1$$
$$= 0.048$$

and, P(No | X):

$$P(No \mid X) = P(Red \mid No) * P(SUV \mid No) * P(Domestic \mid No) * P(No)$$

$$= 2/5 * 3/5 * 3/5 * 1$$
$$= 0.144$$

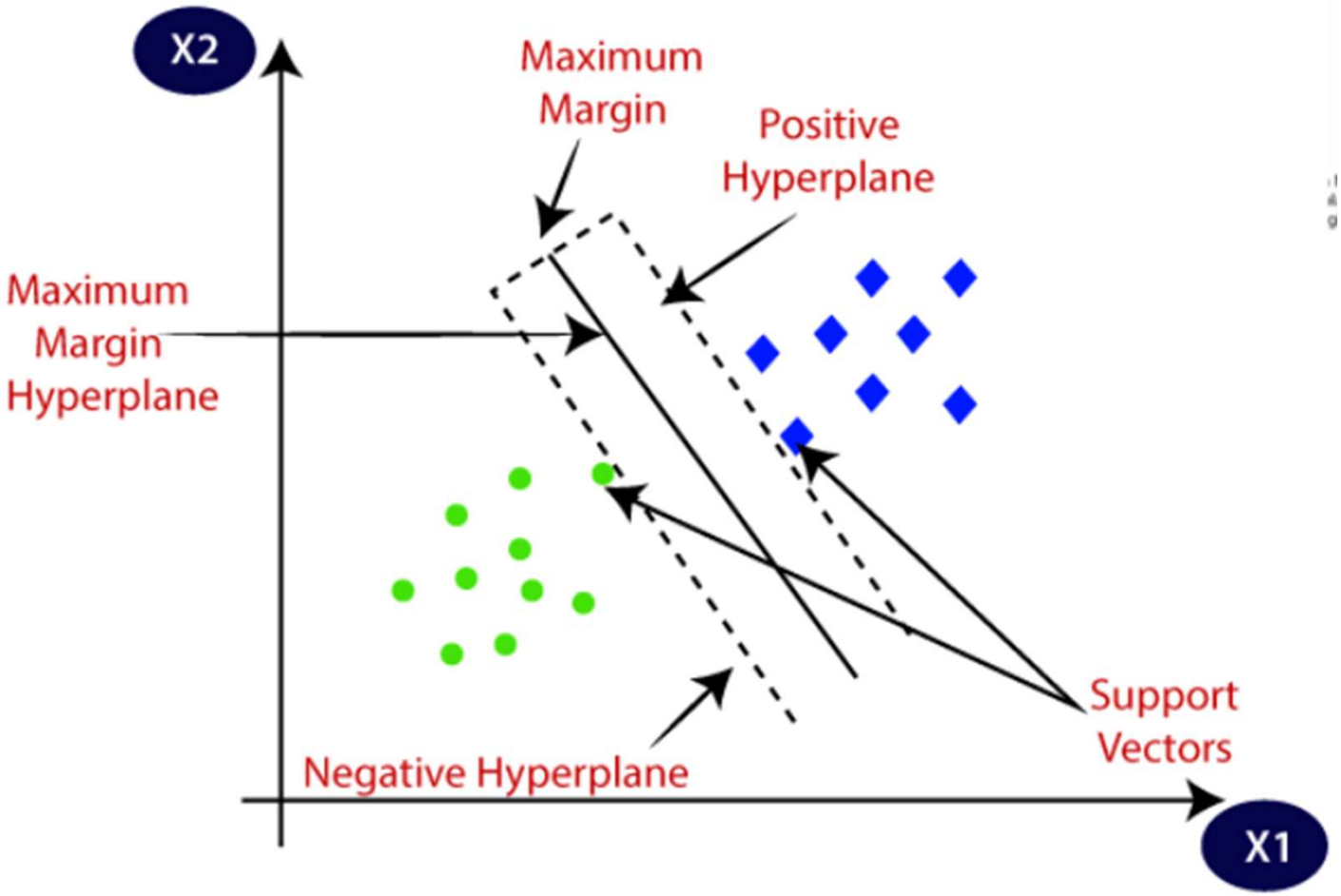
What is a Support Vector Machine(SVM)?

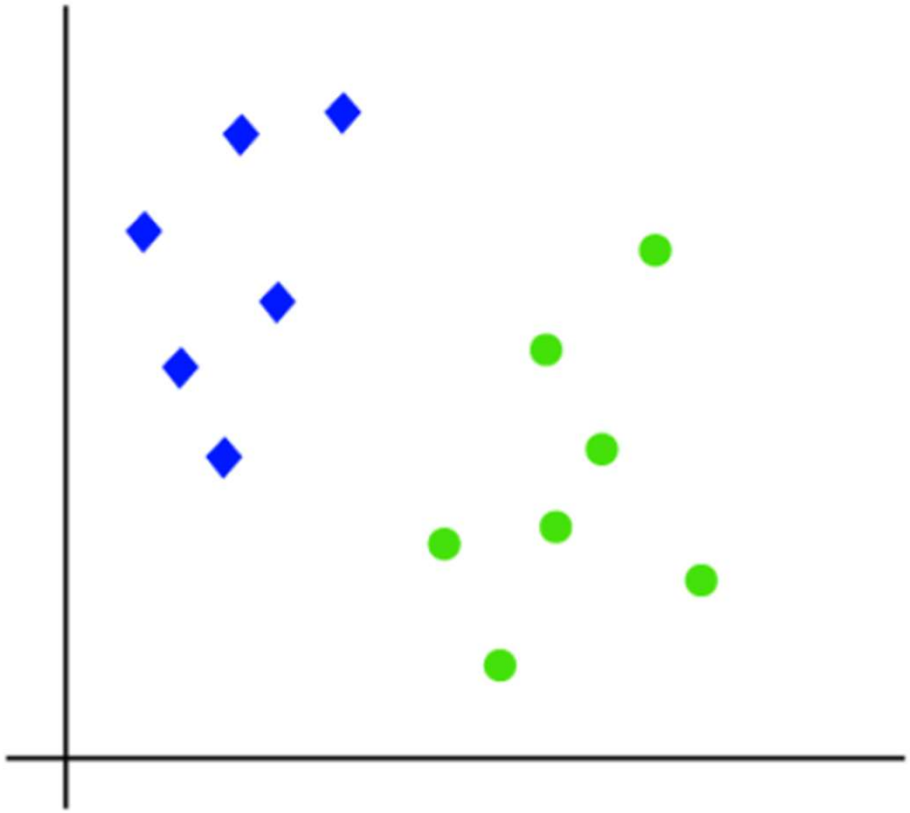
- It is a supervised machine learning problem where we try to find a hyperplane that best separates the two classes.

- Note: Don’t get confused between SVM and logistic regression.
- Both the algorithms try to find the best hyperplane, but the main difference is logistic regression is a probabilistic approach whereas support vector machine is based on statistical approaches.

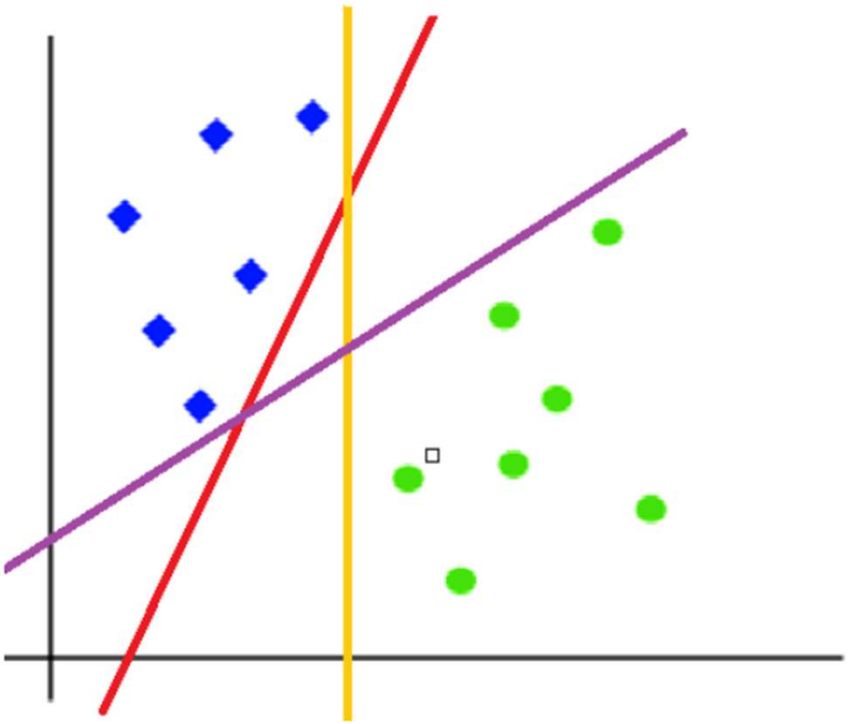
Types of Support Vector Machine (SVM) Algorithms

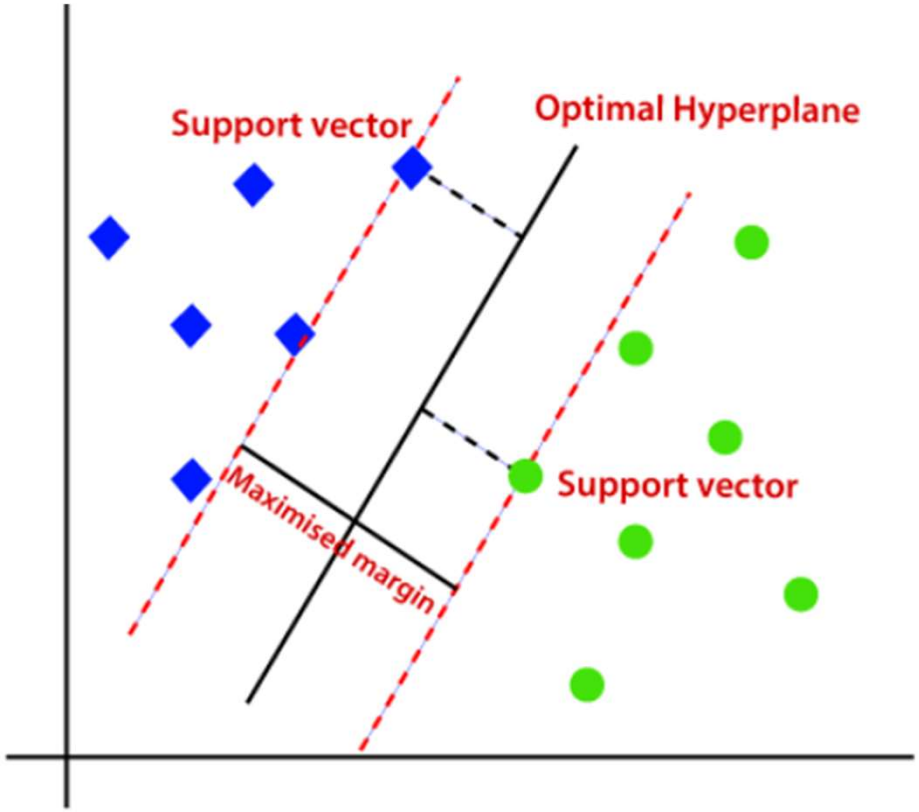
- Linear SVM: When the data is perfectly linearly separable only then we can use Linear SVM.
- Non-Linear SVM: When the data is not linearly separable then we can use Non-Linear SVM then we use some advanced techniques like kernel tricks to classify them.
- In most real-world applications we do not find linearly separable datapoints hence we use kernel trick to solve them.


X2
Maximum
Margin
Positive
Hyperplane
Maximum
Margin
Hyperplane
Support
Vectors
Negative Hyperplane
X1

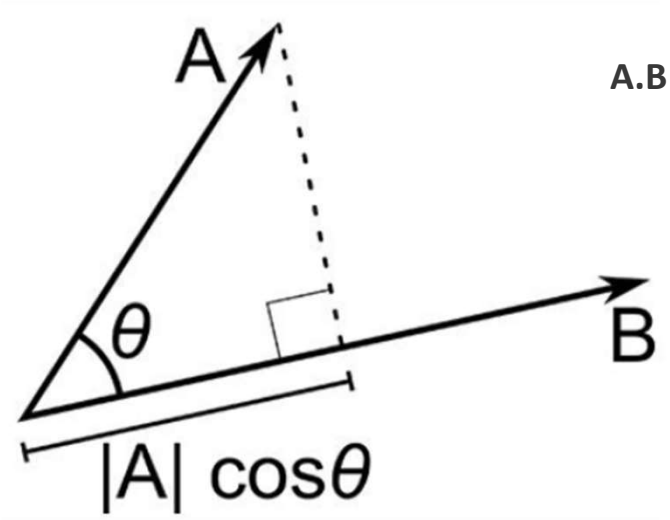


To classify these points, we can have many decision boundaries, but the question is which is the best and how do we find it? **NOTE:** Since we are plotting the data points in a 2-dimensional graph we call this decision boundary a **straight line** but if we have more dimensions, we call this decision boundary a "**hyperplane**"



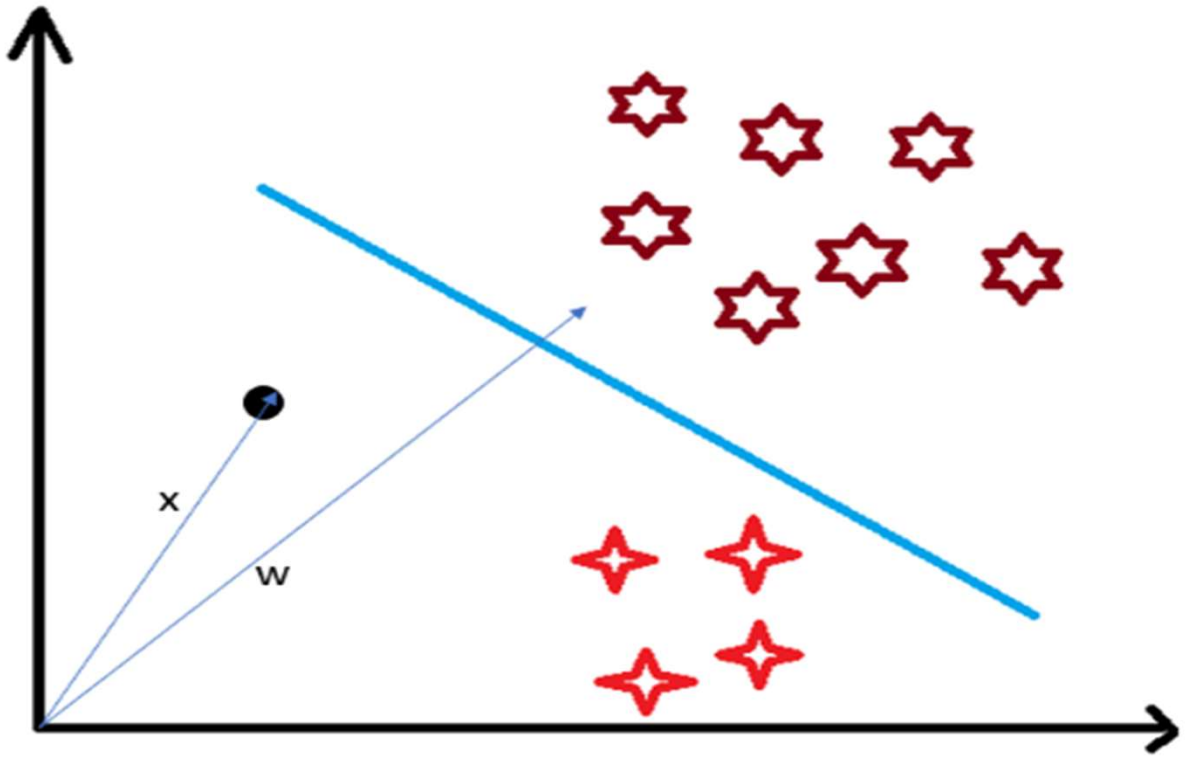

Support vector
Optimal Hyperplane
Maximised margin
Support vector

The dot product can be defined as the projection of one vector along with another, multiply by the product of another vector.

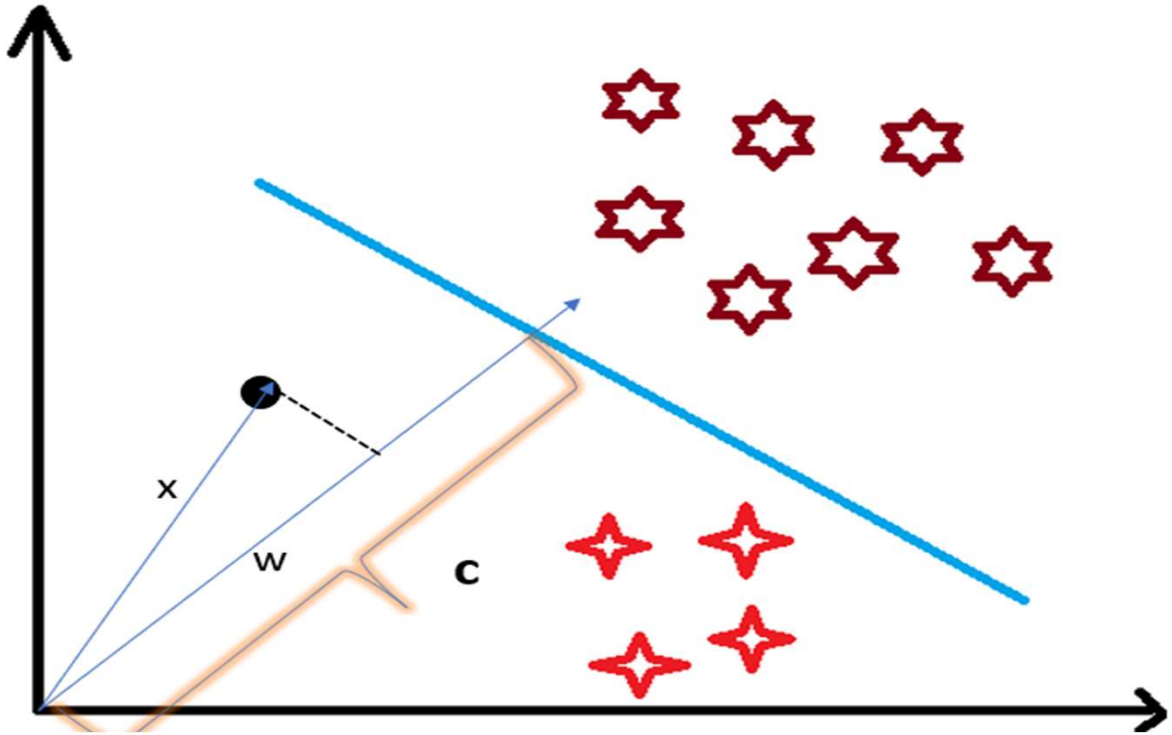


A.B = |A| cosθ * unit vector of B

Consider a random point X and we want to know whether it lies on the right side of the plane or the left side of the plane (positive or negative).



To find this first we assume this point is a vector (X) and then we make a vector (w) which is perpendicular to the hyperplane. Let's say the distance of vector w from origin to decision boundary is 'c'. Now we take the projection of X vector on w.



We already know that projection of any vector or another vector is called dot-product. Hence, we take the dot product of x and w vectors. If the dot product is greater than ‘c’ then we can say that the point lies on the right side. If the dot product is less than ‘c’ then the point is on the left side and if the dot product is equal to ‘c’ then the point lies on the decision boundary.

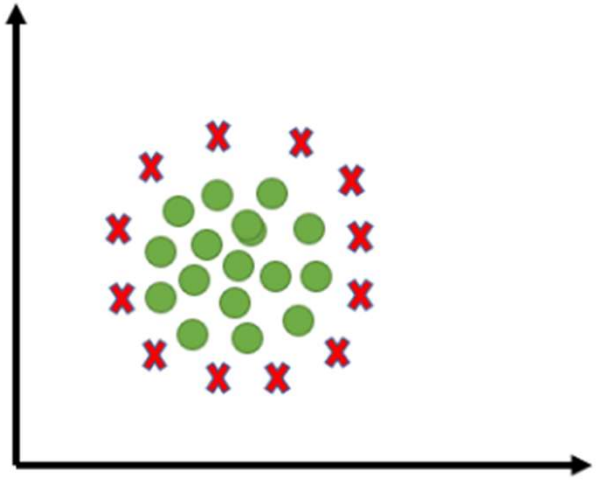
$$\overrightarrow{X}.\overrightarrow{w} = c \quad (the\ point\ lies\ on\ the\ decision\ boundary)$$

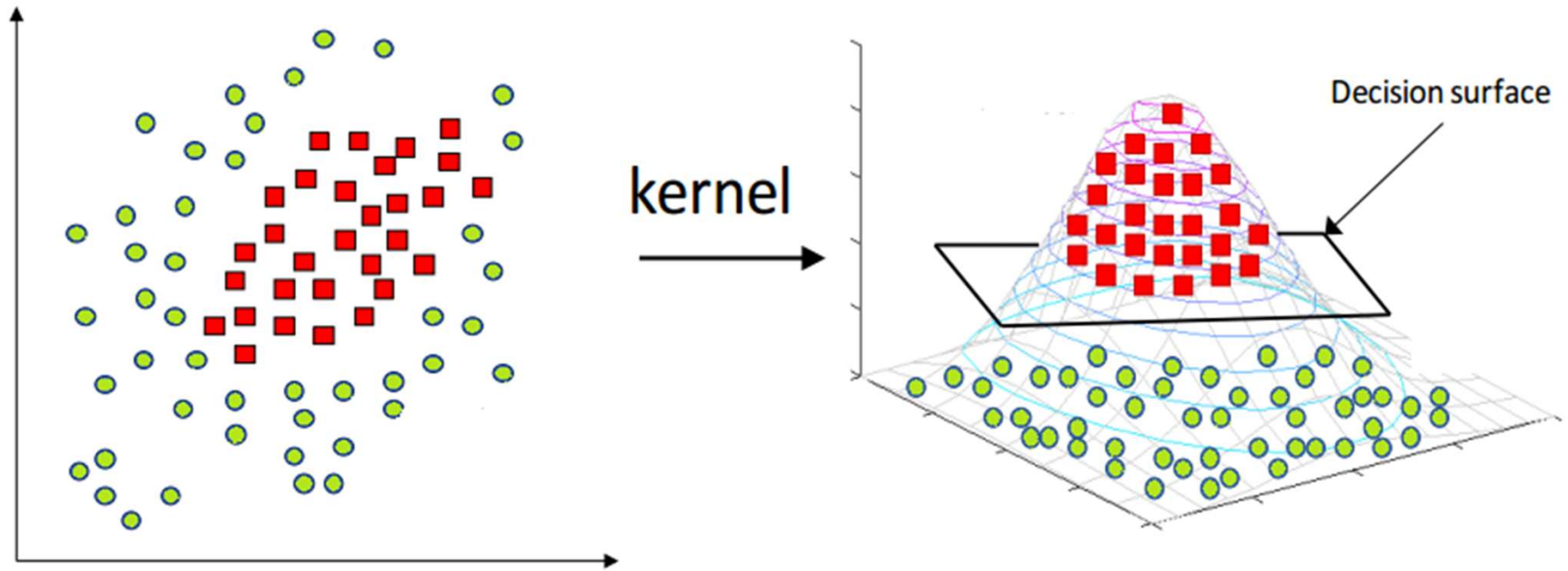
$$\overrightarrow{X}.\overrightarrow{w} > c\ (positive\ samples)$$

$$\overrightarrow{X}.\overrightarrow{w} < c\ (negative\ samples)$$

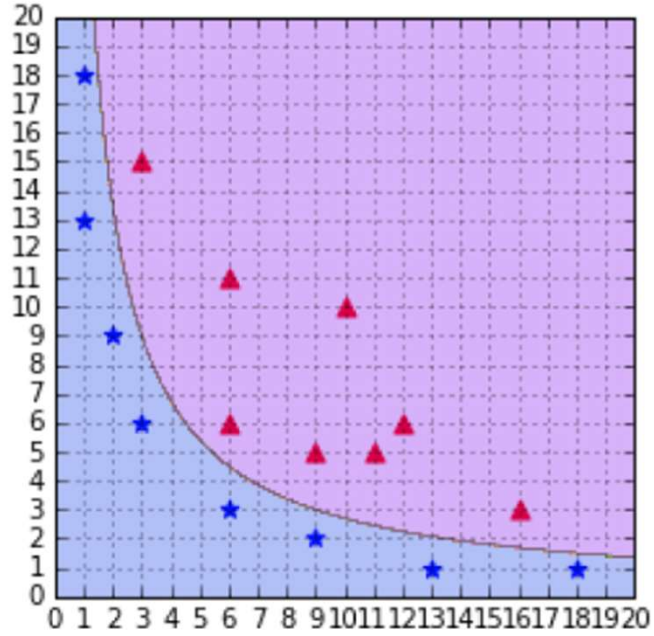
Kernels in Support Vector Machine

The most interesting feature of SVM is that it can even work with a non-linear dataset and for this, we use “Kernel Trick” which makes it easier to classifies the points. Suppose we have a dataset like this:



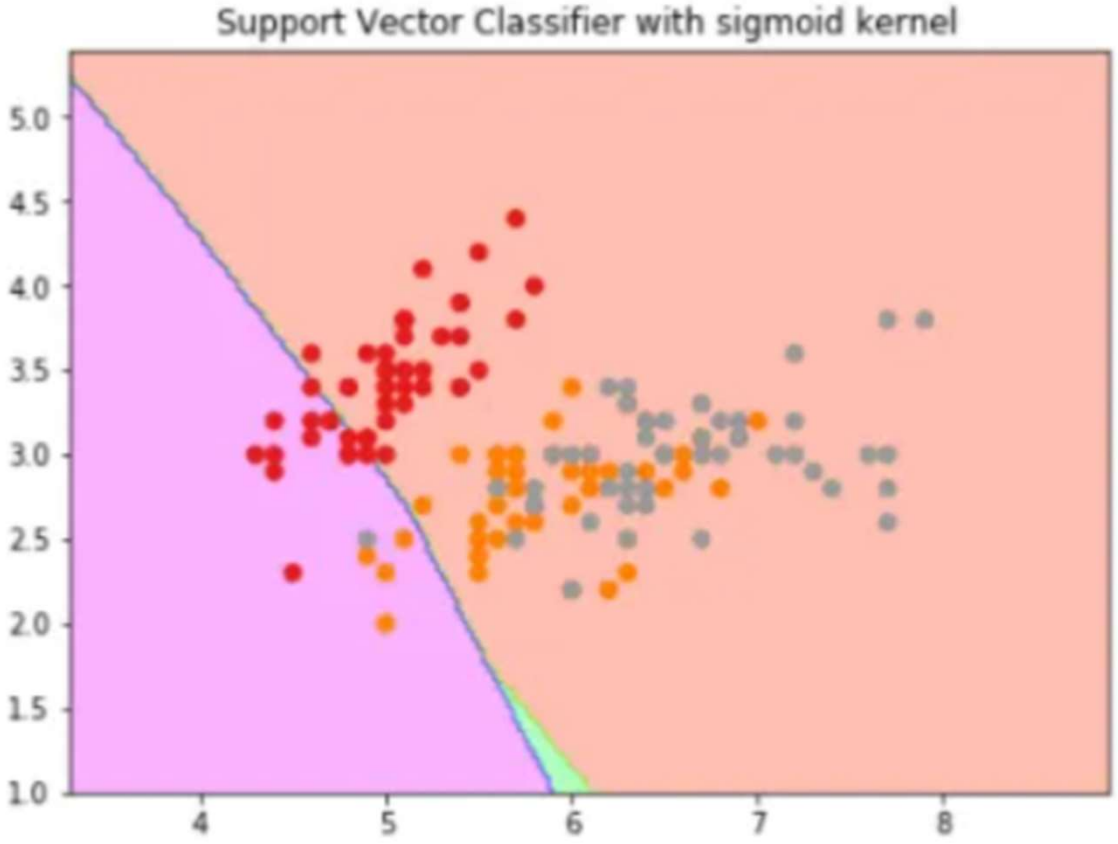

kernel
Decision surface

Polynomial Kernel



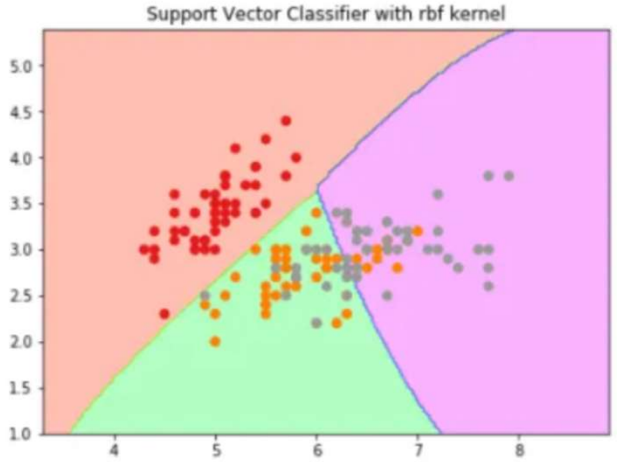
A SVM using a polynomial kernel is able to separate the data (degree=2)

Sigmoid Kernel



Rbf distancebased

1. ‘σ’ is the variance and our hyperparameter
2. $||X_1 - X_2||$ is the Euclidean Distance between two points X_1 and X_2

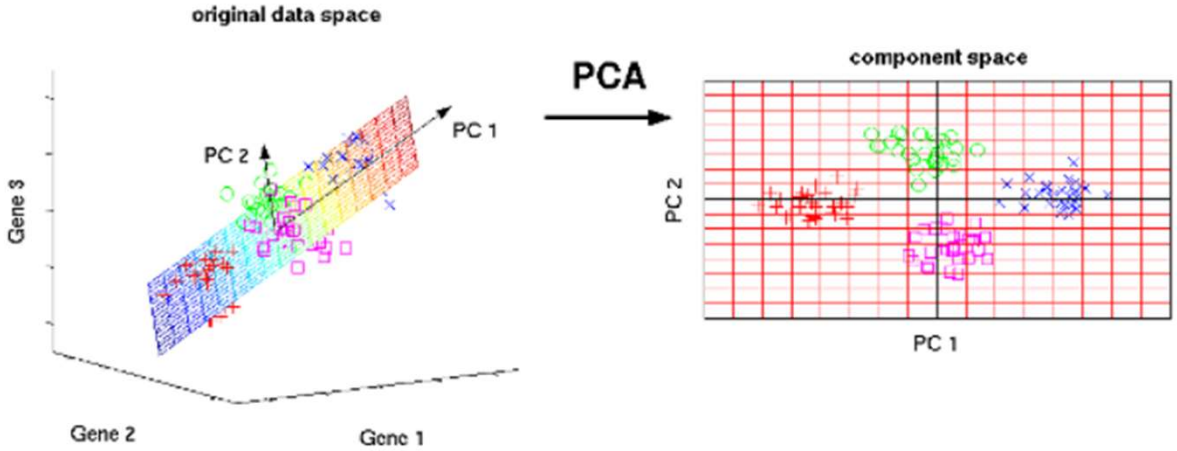


Principal Component Analysis (PCA)

- Picture this – you are working on a large-scale data science project. What happens when the given data set has too many variables? There are a few possible situations that you might come across. For instance, you find that most of the variables are correlated on analysis, and decide to run a model on the whole data.
- This returns poor accuracy, and you feel terrible and start thinking of some strategic method to find a few important variables. That's where Principal Component Analysis (PCA) is used.

- **Linear Transformation**: PCA performs a linear transformation of data, seeking directions of maximum variance.
- **Feature Selection**: Principal components are ranked by the variance they explain, allowing for effective feature selection.
- **Data Compression**: PCA can compress data while preserving most of the original information.
- **Clustering and Classification**: It finds applications in clustering and classification tasks by reducing noise and highlighting underlying structure.
- **Advantages**: PCA offers linearity, computational efficiency, and scalability for large datasets.
- **Limitations**: It assumes data normality and linearity and may lead to information loss.

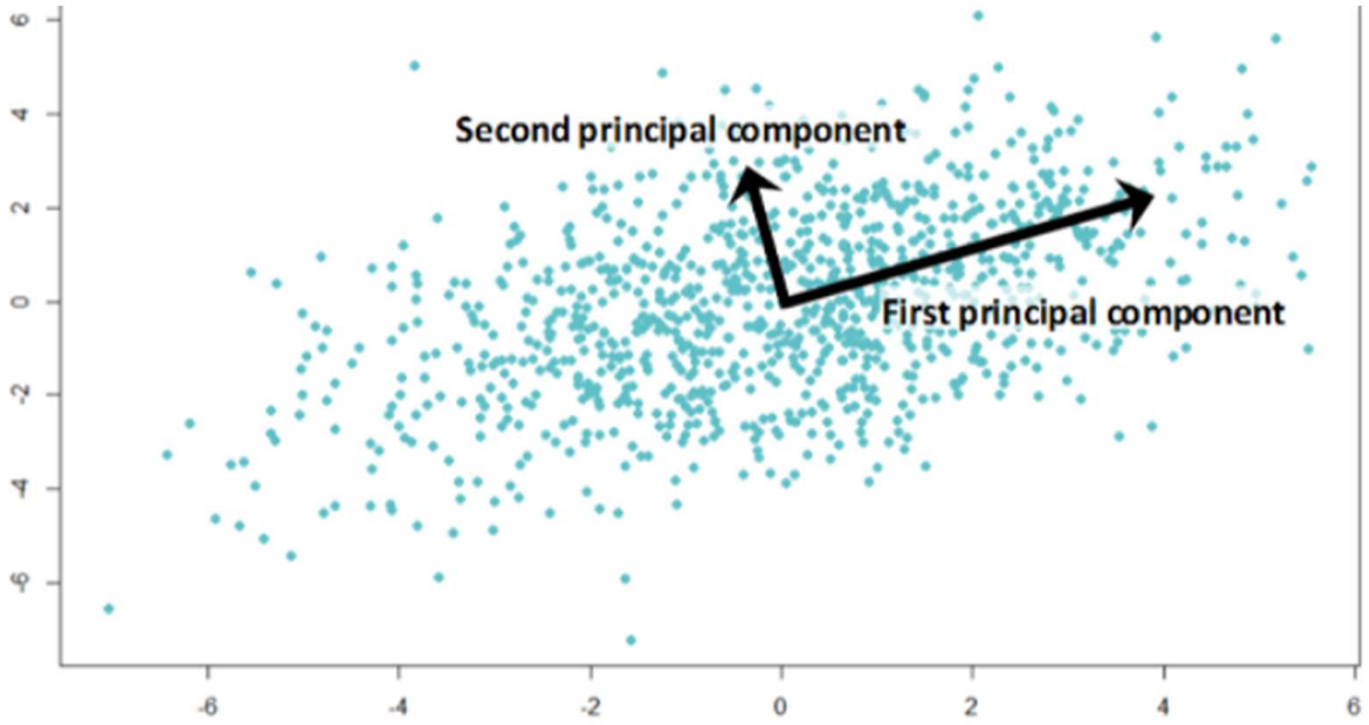
The image below shows the transformation of high-dimensional data (3 dimension) to low-dimensional data (2 dimension) using PCA. Not to forget, each resultant dimension is a linear combination of *p* features



- The first principal component is a linear combination of original predictor variables that captures the data set's maximum variance.

- The second principal component is also a linear combination of original predictors, which captures the remaining variance in the data set and is uncorrelated with Z^1.
- In other words, the correlation between first and second components should be zero. It can be represented as:

- $Z^2 = \Phi^{12}X^1 + \Phi^{22}X^2 + \Phi^{32}X^3 + ... + \Phi p2Xp$


Second principal component
First principal component
6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6

How Principal Component Analysis (PCA) Work ?

1. **Standardize the Data**

   If the features of your dataset are on different scales, it's essential to standardize them (subtract the mean and divide by the standard deviation).

2. **Compute the Covariance Matrix**

   Calculate the covariance matrix for the standardized dataset.

3. **Compute Eigenvectors and Eigenvalues**

   Find the eigenvectors and eigenvalues of the covariance matrix. The eigenvectors represent the directions of maximum variance, and the corresponding eigenvalues indicate the magnitude of variance along those directions.

4. **Sort Eigenvectors by Eigenvalues**

   Sort the eigenvectors based on their corresponding eigenvalues in descending order.

5. **Choose Principal Components**

   Select the top k eigenvectors (principal components) where k is the desired dimensionality of the reduced dataset.

6. **Transform the Data**

   Multiply the original standardized data by the selected principal components to obtain the new, lower-dimensional representation of the data.

- Principal Component Analysis (PCA) Examples
- Image Compression: It reduces image dimensionality for efficient storage without losing critical information.
- Genomic Data Analysis: PCA identifies patterns in gene expression data, aiding in disease research.
- Financial Data Analysis: It analyzes covariance in asset returns for portfolio optimization.
- Spectral Analysis: PCA helps in signal processing to identify dominant spectral features.
- Customer Segmentation: It clusters customers based on behavior for targeted marketing.

- Why is Normalization of Variables Necessary in Principal Component Analysis (PCA)?
- The principal components are supplied with a normalized version of the original predictors. This is because the original predictors may have different scales. For example: Imagine a data set with variables measuring units as gallons, kilometers, light years, etc. The scale of variances in these variables will obviously be large.

- Performing PCA on un-normalized variables will lead to exponentially large loadings for variables with high variance. In turn, this will lead to the dependence of a principal component on the variable with high variance. This is undesirable.

RANDOM FOREST

- Random Forest is a widely-used machine learning algorithm which combines the output of multiple decision trees to reach a single result.

- **Customer churn prediction**
- **Fraud detection**
- **Stock price prediction**
- **Medical diagnosis**

- Step 1: In the Random forest model, a subset of data points and a subset of features is selected for constructing each decision tree.
- Step 2: Individual decision trees are constructed for each sample.
- Step 3: Each decision tree will generate an output.
- Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression, respectively.

- **n_estimators:** Number of trees the algorithm builds before averaging the predictions.
- **max_features:** Maximum number of features random forest considers splitting a node.
- **mini_sample_leaf:** Determines the minimum number of leaves required to split an internal node.
- **criterion:** How to split the node in each tree? (Entropy/Gini impurity/Log Loss)
- **max_leaf_nodes:** Maximum leaf nodes in each tree

- It can be used in classification and regression problems.
- It solves the problem of overfitting as output is based on majority voting or averaging.
- It performs well even if the data contains null/missing values.
- Each decision tree created is independent of the other; thus, it shows the property of parallelization.
- It is highly stable as the average answers given by a large number of trees are taken.
- It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.
- It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.
- We don’t have to segregate data into train and test as there will always be 30% of the data, which is not seen by the decision tree made out of bootstrap.

K MEANS

```
Initialize k means with random values
--> For a given number of iterations:

    --> Iterate through items:

        --> Find the mean closest to the item by calculating
        the euclidean distance of the item with each of the means

        --> Assign item to mean

        --> Update mean by shifting it to the average of the items in that cluster
```

Hyper parameter tuning are influencing the below factors while designing your model

- Linear Model
  - What degree of polynomial features should use?
- Decision Tree
  - What is the maximum allowed depth?
- What is the minimum number of samples required at a leaf node in the decision tree?
  - Random forest
- How many trees we should include?
  - Neural Network
  - How many neurons we should keep in a layer?

- In the ML world, there are many Hyperparameter optimization techniques are available.
- Manual Search
- Random Search
- Grid Search
- Halving
  - Grid Search
  - Randomized Search
- Automated Hyperparameter tuning
  - Bayesian Optimization
  - Genetic Algorithms

Ensemble Learning Techniques

- **Gradient Boosting Machines (GBM):**

- sequentially builds a group of decision trees and corrects the residual errors made by previous trees, enhancing its predictive accuracy

- **Extreme Gradient Boosting (XGBoost)**:

- XGBoost features tree pruning, regularization, and parallel processing, which makes it a preferred choice for data scientists seeking robust and accurate predictive models.

- **CatBoost:**

- It is designed to handle features categorically that eliminates the need for extensive pre-processing.

- high predictive accuracy, fast training, and automatic handling of overfitting.

- **Stacking:**

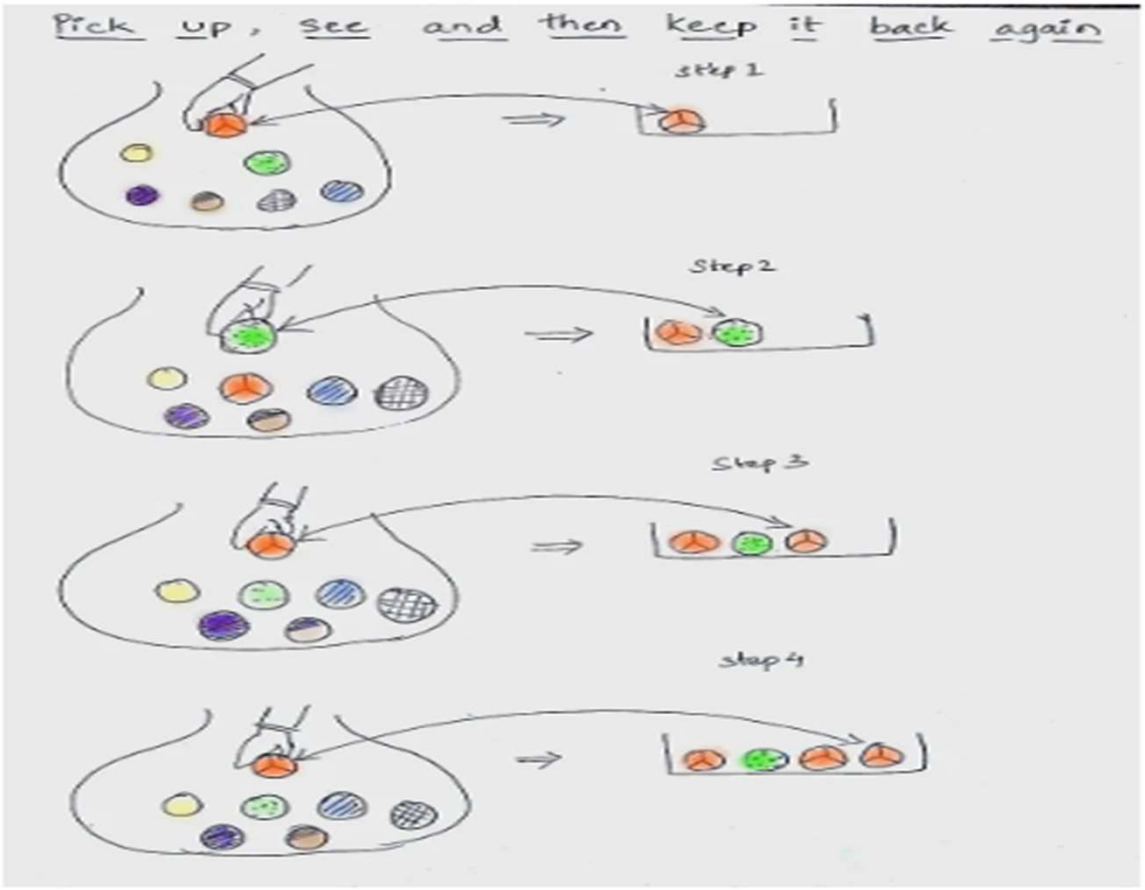
- It combines the output of multiple base models by training a combiner(an algorithm that takes predictions of base models) and generate more accurate prediction.

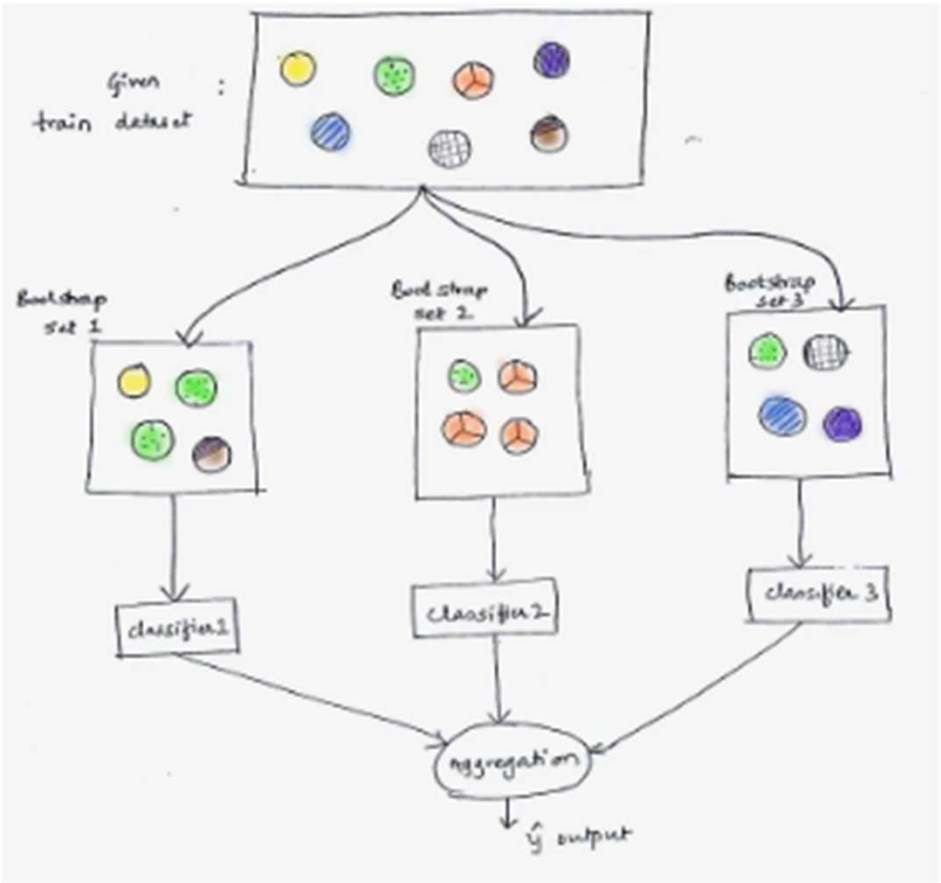
- **Bagging Algorithm**
- Bagging is a supervised learning technique that can be used for both regression and classification tasks.

Some pointers about bagging technique:

1. Aggregation function is typically the statistical mode for classification and average for regression.
2. Aggregation helps reduce both bias and variance.
3. Predictors can all be trained in parallel unleashing the potential of a GPU.
4. Bagging allows training instances to be sampled several times for the same classifier. *(example: "orange" ball is picked up 3 times for classifier 2 as in the above figure)*

Let's try to understand this with a visual cue shown below:

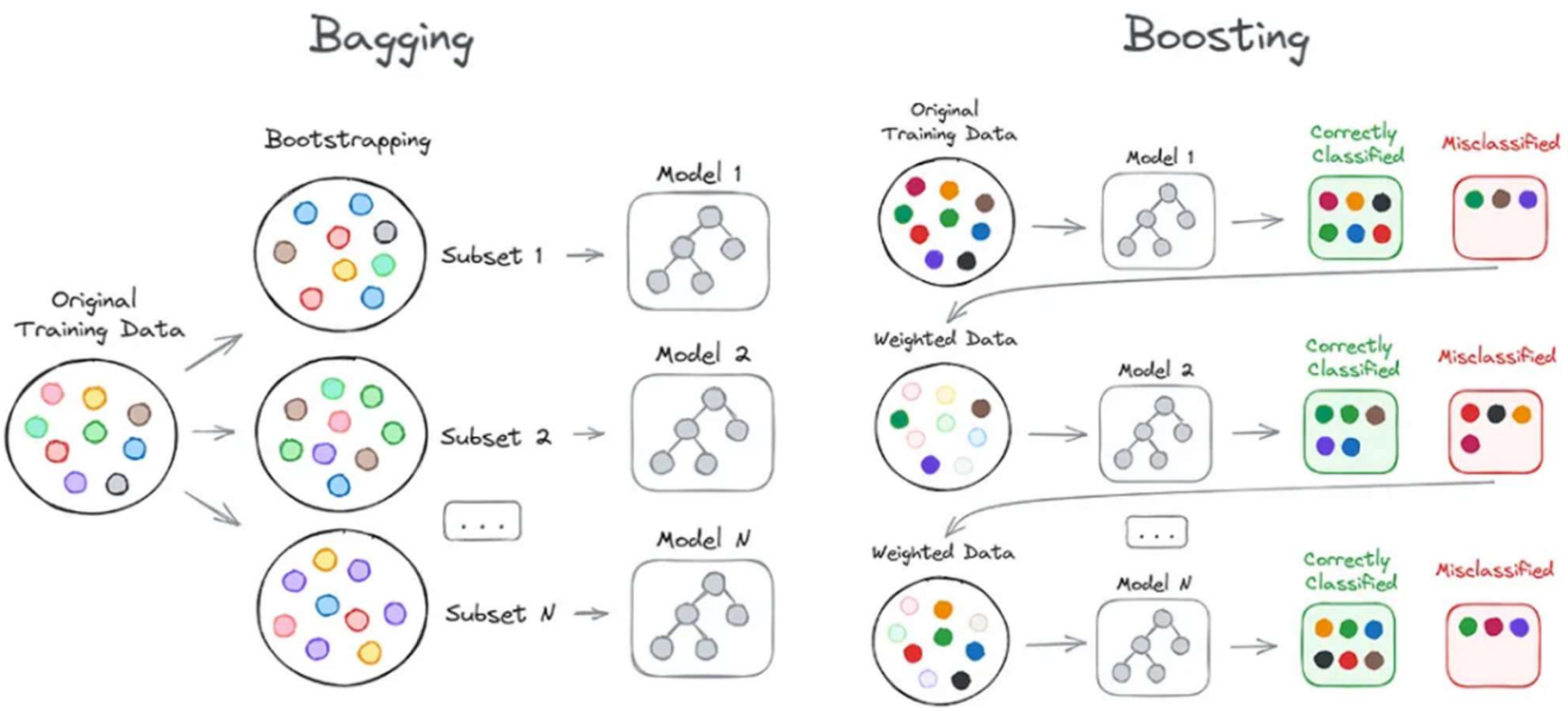




Bagging Ensemble Predictor.

```python
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Load the Iris dataset
data = load_iris()
X = data.data
y = data.target
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Create a base classifier (e.g., Decision Tree)
base_classifier = DecisionTreeClassifier()
bagging_classifier = BaggingClassifier(base_classifier, n_estimators=10, random_state=42)
bagging_classifier.fit(X_train, y_train)
y_pred = bagging_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

- **Boosting Algorithm**
- Boosting is an ensemble technique that combines multiple weak learners to create a strong learner.
- The ensemble of weak models are trained in series such that each model that comes next, tries to correct errors of the previous model until the entire training dataset is predicted correctly. One of the most well-known boosting algorithms is AdaBoost (Adaptive Boosting).


Bagging
Bootstrapping
Original
Training Data
Subset 1
Subset 2
. . .
Subset N
Model 1
Model 2
Model N
Boosting
Original
Training Data
Model 1
Correctly
Classified
Misclassified
Weighted Data
Model 2
Correctly
Classified
Misclassified
. . .
Weighted Data
Model N
Correctly
Classified
Misclassified

```python
# Import necessary libraries and modules
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
# Load the dataset
data = load_iris()
X = data.data
y = data.target
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
base_classifier = DecisionTreeClassifier(max_depth=1)  # Weak learner
# Create an AdaBoost Classifier with Decision Tree as the base classifier
adaboost_classifier = AdaBoostClassifier(base_classifier, n_estimators=50, learning_rate=1.0, random_sta
adaboost_classifier.fit(X_train, y_train)
# Make predictions
y_pred = adaboost_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

- **Classification and Regression:** Ensemble techniques make problems like classification and regression versatile in various domains, including finance, healthcare, marketing, and more.
- **Anomaly Detection:** Ensembles can be used to detect anomalies in datasets by combining multiple anomaly detection algorithms, thus making it more robust.
- **Portfolio Optimization:** Ensembles can be employed to optimize investment portfolios by collecting predictions from various models to make better investment decisions.
- **Customer Churn Prediction:** In business and marketing analytics, by combining the results of various models capturing different aspects of customer behaviour, ensembles can be used to predict customer churn.
- **Medical Diagnostics:** In healthcare, ensembles can be used to make more accurate predictions of diseases based on various medical data sources and diagnostic models.
- **Credit Scoring:** Ensembles can be used to improve the accuracy of credit scoring models by combining the outputs of various credit risk assessment models.
- **Climate Prediction:** Ensembles of climate models help in making more accurate and reliable predictions for weather forecasting, climate change projections, and related environmental studies.
- **Time Series Forecasting**: Ensemble learning combines multiple time series forecasting models to enhance accuracy and reliability, adapting to changing temporal patterns.

Types of Data Professionals

Data Engineer

Data Scientists
ML Ops

ML Engineer

Data Analysts
ML Ops

Data Scientist

Data Analyst

25 Most important Mathematical Definitions in Data Science

blog.DailyDoseofDS.com

1) Gradient Descent

$$\theta_{j+1} = \theta_j - \alpha \nabla J(\theta_j)$$

2) Normal distribution

$$f(x|\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

3) Z-score

$$z = \frac{x - \mu}{\sigma}$$

4) Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

5) Correlation

$$\text{Correlation} = \frac{\text{Cov}(X, Y)}{\text{Std}(X) \cdot \text{Std}(Y)}$$

6) Cosine Similarity

$$\text{similarity} = \frac{A \cdot B}{\|A\| \|B\|}$$

7) Naive Bayes

$$P(y|x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i|y)}{P(x_1, \ldots, x_n)}$$

8) MLE

$$\text{argmax}_\theta \prod_{i=1}^{n} P(x_i|\theta)$$

9) OLS

$$\hat{\beta} = (X^T X)^{-1} X^T y$$

10) F1 Score

$$\frac{2 \cdot P \cdot R}{P + R}$$

11) ReLU

$$\max(0, x)$$

12) Softmax

$$P(y = j|x) = \frac{e^{x^T w_j}}{\sum_{k=1}^{K} e^{x^T w_k}}$$

13) R2 score

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

14) MSE

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

15) MSE + L2 Reg

$$\text{MSE}_{\text{regularized}} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

16) Eigen vectors

$$Av = \lambda v$$

17) Entropy

$$\text{Entropy} = -\sum_i p_i \log_2(p_i)$$

18) KMeans

$$\text{argmin}_S \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2$$

19) KL Divergence

$$D_{\text{KL}}(P\|Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

20) Log-loss

$$-\frac{1}{N} \sum_{i=1}^{N} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)\right)$$

21) SVM

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i(w \cdot x_i - b))$$

22) Linear regression

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n + \epsilon$$

23) SVD

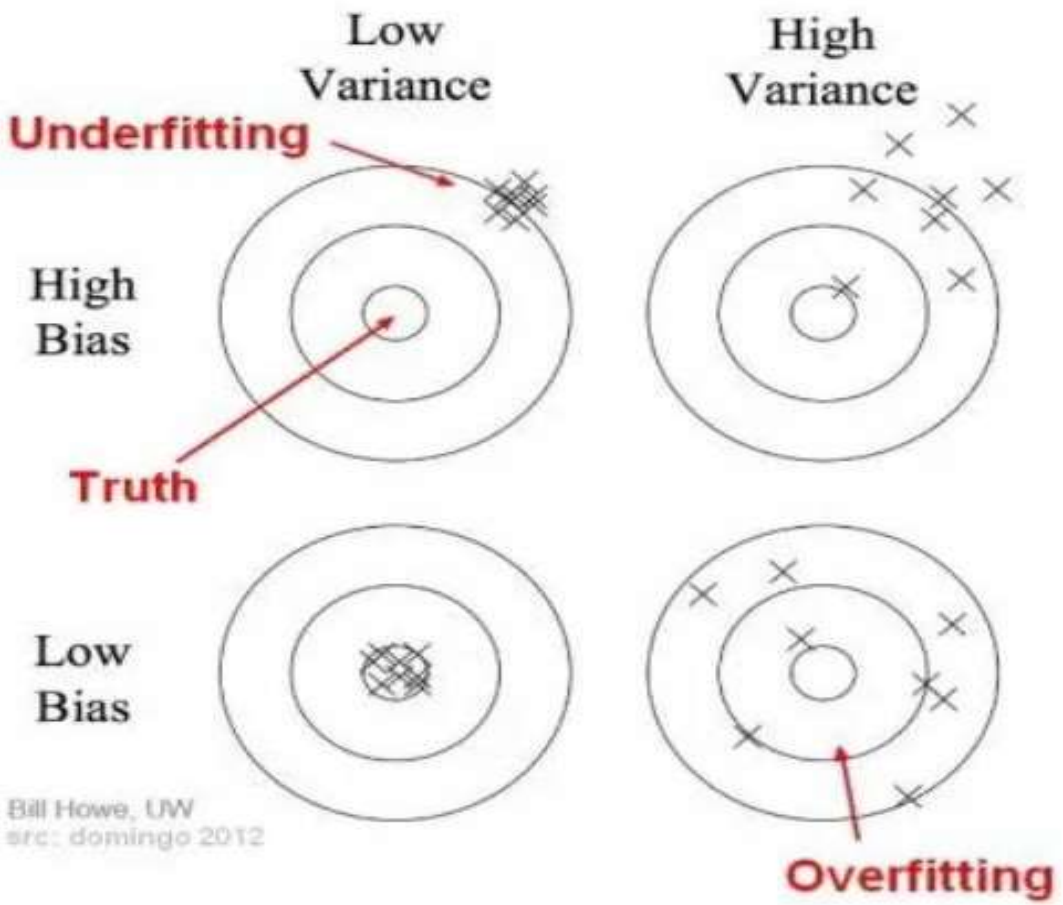
$$A = U \Sigma V^T$$

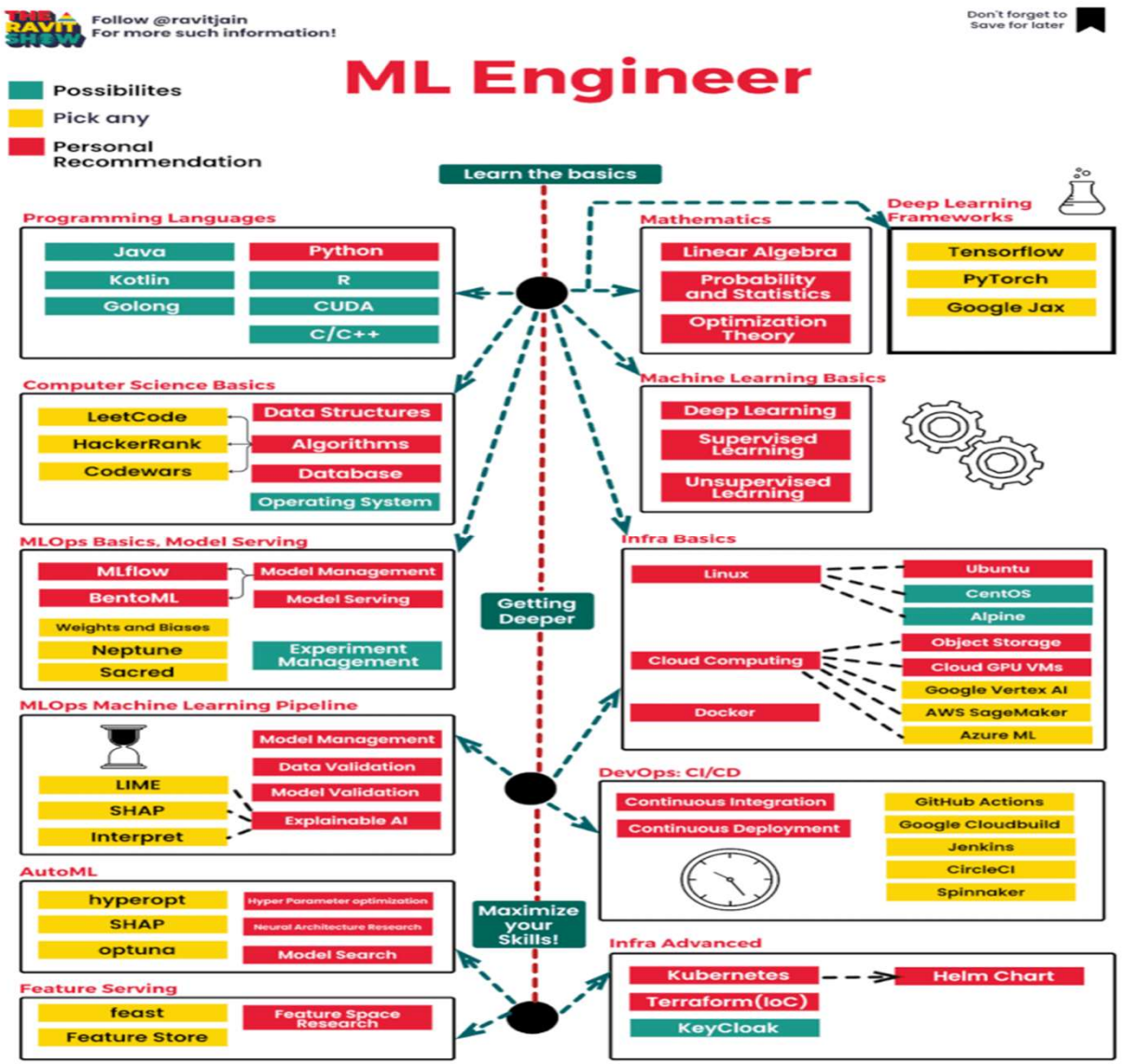
24) Lagrange multiplier

$$\max f(x) \; ; \; g(x) = 0$$
$$L(x, \lambda) = f(x) - \lambda * g(x)$$

25) What will you add?

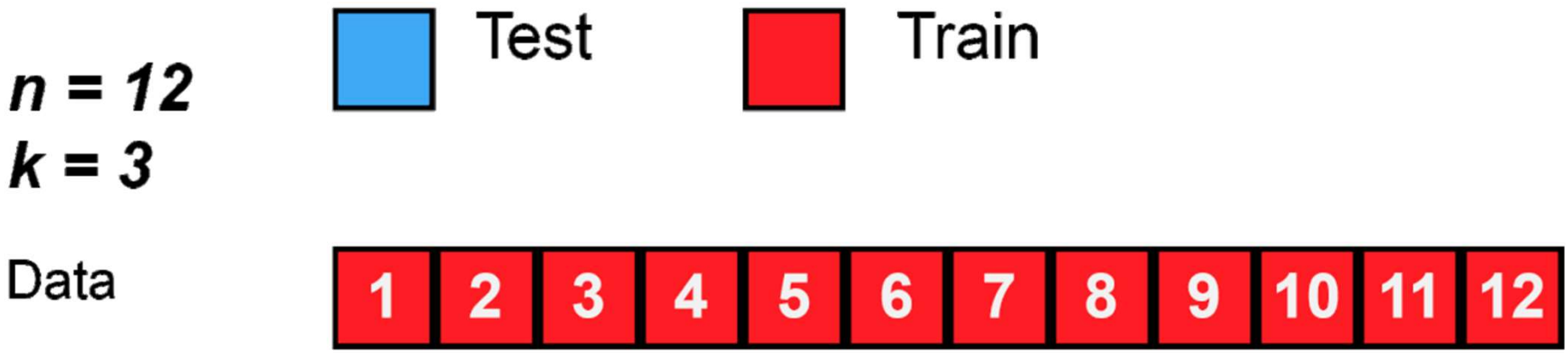
• • •

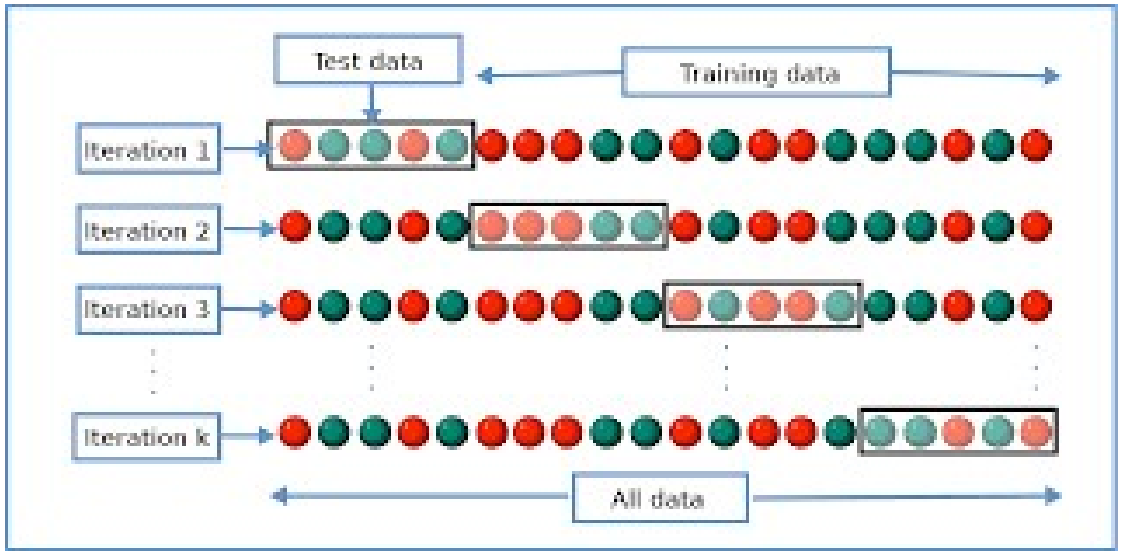

Low
Variance
High
Variance
Underfitting
High
Bias
Truth
Low
Bias
Bill Howe, UW
src: domingo 2012
Overfitting


Follow @ravitjain
For more such information!
Don't forget to Save for later
ML Engineer
Possibilites
Pick any
Personal Recommendation
Learn the basics
Programming Languages
Java
Python
Kotlin
R
Golong
CUDA
C/C++
Mathematics
Linear Algebra
Probability and Statistics
Optimization Theory
Deep Learning Frameworks
Tensorflow
PyTorch
Google Jax
Computer Science Basics
LeetCode
HackerRank
Codewars
Data Structures
Algorithms
Database
Operating System
Machine Learning Basics
Deep Learning
Supervised Learning
Unsupervised Learning
MLOps Basics, Model Serving
MLflow
BentoML
Weights and Biases
Neptune
Sacred
Model Management
Model Serving
Experiment Management
Getting Deeper
Infra Basics
Linux
Ubuntu
CentOS
Alpine
Cloud Computing
Object Storage
Cloud GPU VMs
Google Vertex AI
AWS SageMaker
Azure ML
Docker
MLOps Machine Learning Pipeline
LIME
SHAP
Interpret
Model Management
Data Validation
Model Validation
Explainable AI
DevOps: CI/CD
Continuous Integration
Continuous Deployment
GitHub Actions
Google Cloudbuild
Jenkins
CircleCI
Spinnaker
AutoML
hyperopt
SHAP
optuna
Hyper Parameter optimization
Neural Architecture Research
Model Search
Maximize your Skills!
Infra Advanced
Kubernetes
Helm Chart
Terraform(IoC)
KeyCloak
Feature Serving
feast
Feature Store
Feature Space Research

- **What is Cross-Validation?**
- Cross validation is a technique used in machine learning to evaluate the performance of a model on unseen data.

- It involves dividing the available data into multiple folds or subsets, using one of these folds as a validation set, and training the model on the remaining folds.


Test
Train
n = 12
k = 3
Data
1
2
3
4
5
6
7
8
9
10
11
12


Test data
Training data
Iteration 1
Iteration 2
Iteration 3
Iteration k
All data

The Apriori Algorithm -- Example