MACHINE LEARNING OVERVIEW

GETTING THE MOST OUT OF YOUR DATA

Purushottam Darshankar

AGENDA

- ✓ Introduction to Machine learning and its application
- ✓ Overview of Specific methods
  - Supervised
    - Regression
    - Classification
  - Unsupervised
    - Clustering / Segmentation
    - Anomaly Detection
    - Dimensionality Reduction
  - Reinforced
- ✓ Best Practices of Machine Learning

MACHINE LEARNING

Wikipedia:

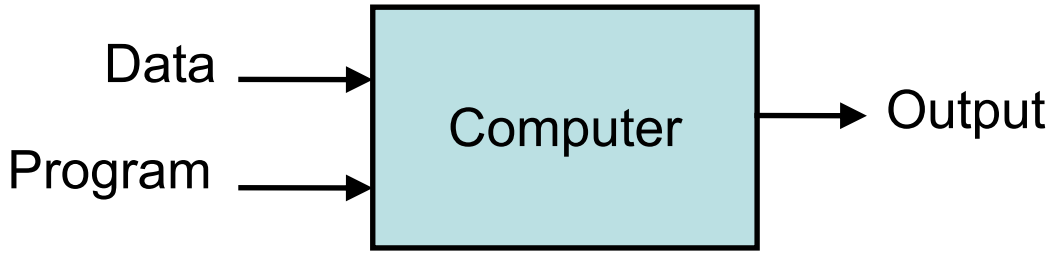
“Machine learning is a scientific discipline that deals with the construction and study of algorithms that can learn from data. Such algorithms operate by building a model based on inputs and using that to make predictions or decisions, rather than following only explicitly programmed instructions.”

MACHINE LEARNING WHAT IS IT?

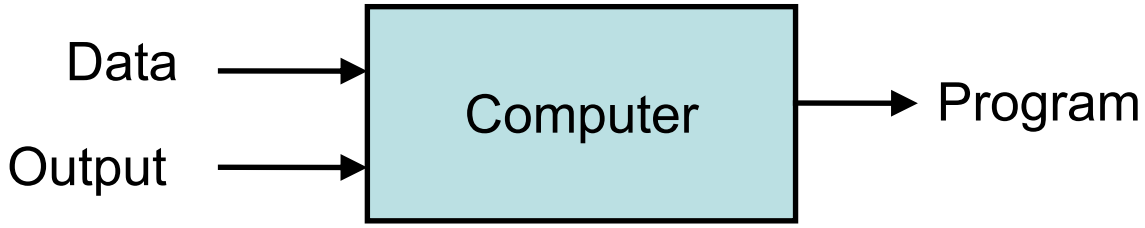
“MACHINE LEARNING REFERS TO A SYSTEM CAPABLE OF THE AUTONOMOUS ACQUISITION AND INTEGRATION OF KNOWLEDGE.”

"FIELD OF STUDY THAT GIVES COMPUTERS THE ABILITY TO LEARN WITHOUT BEING EXPLICITLY PROGRAMMED.”

“GETTING COMPUTERS TO PROGRAM THEMSELVES.”



Traditional Programming

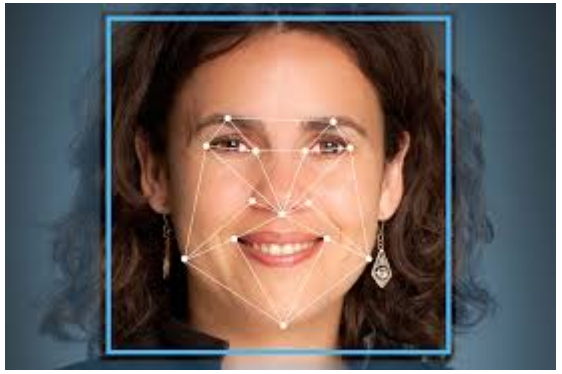


Machine Learning

WHEN WOULD WE USE MACHINE LEARNING?

- ✓ When patterns exists in our data
  - Even if we don't know what they are
- ✓ We can not pin down the functional relationships mathematically
  - Else we would just code up the algorithm
- ✓ When we have lots of (unlabeled) data
  - Labeled training sets harder to come by
  - Data is of high-dimension - for example, sensor data
  - Want to "discover" lower-dimension representations -Dimension reduction
- ✓ Rapidly changing phenomena
  - credit scoring, financial modeling
  - diagnosis, fraud detection

MACHINE LEARNING | FEW EXAMPLES



Text / Image / Voice Recognition

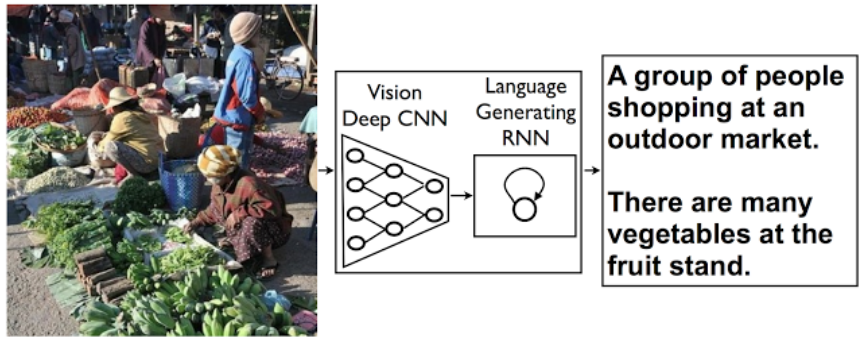
Spam Mail

Self Driving Cars

NETFLIX

Flipkart

Recommendations



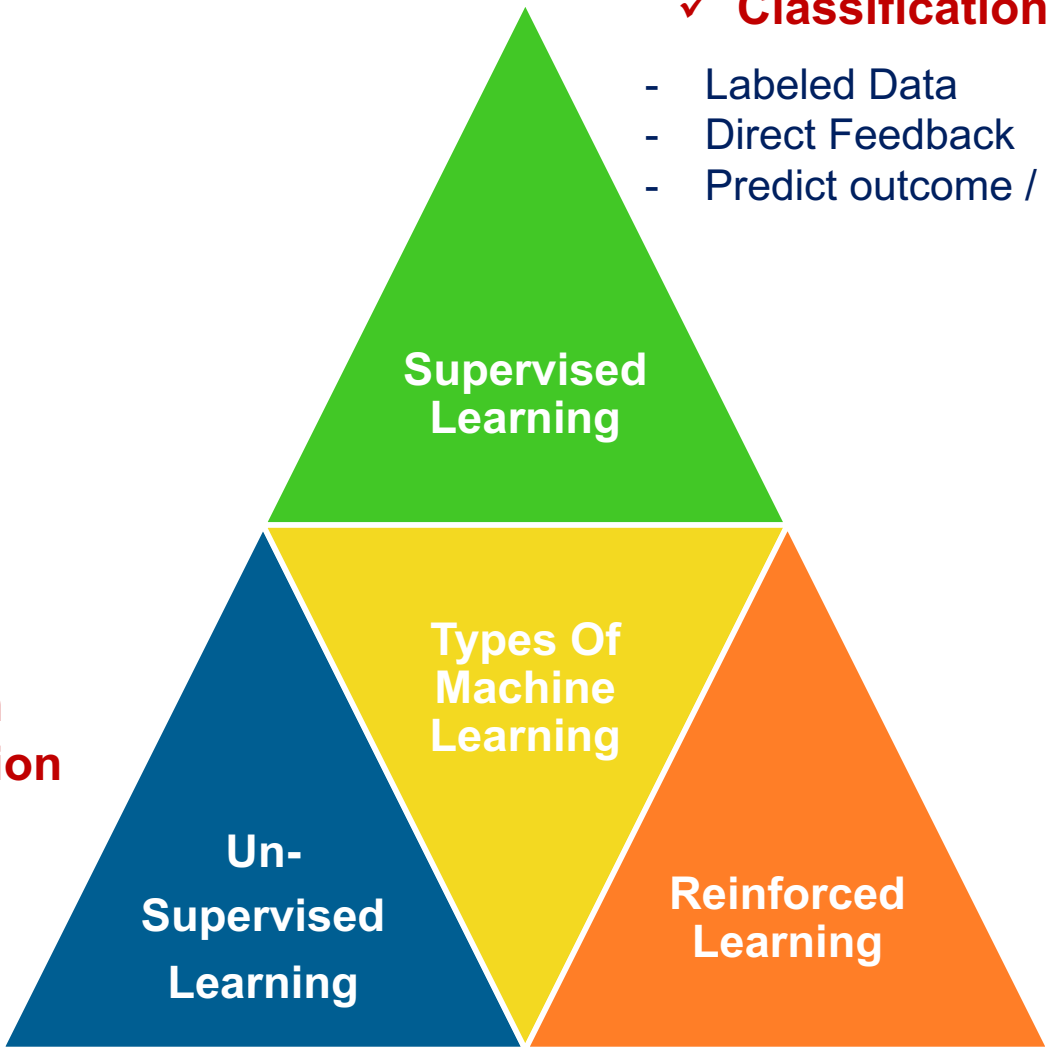
Picture Annotations

… and many more

MACHINE LEARNING | TYPES OF LEARNING

✓ **Regression**
✓ **Classification**

- Labeled Data
- Direct Feedback
- Predict outcome / Future

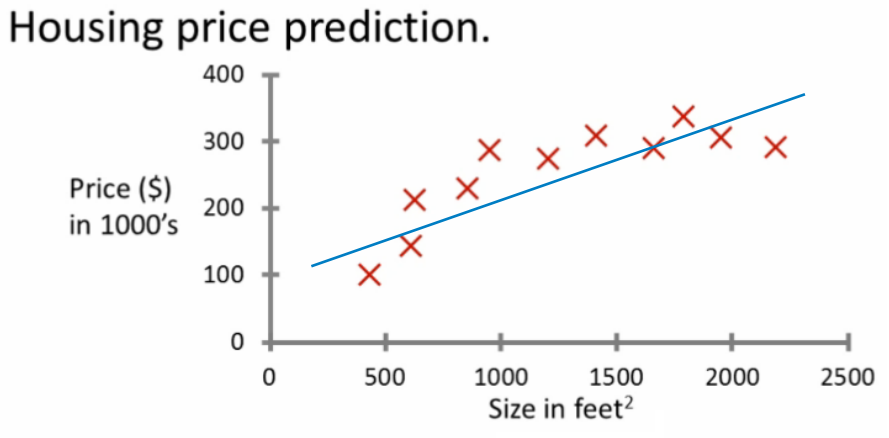


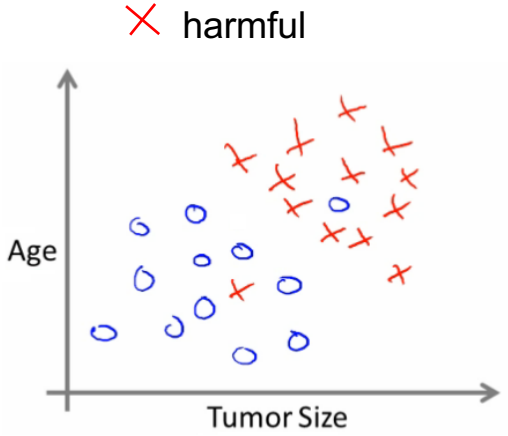
✓ **Clustering**
✓ **Anomaly Detection**
✓ **Dimension Reduction**

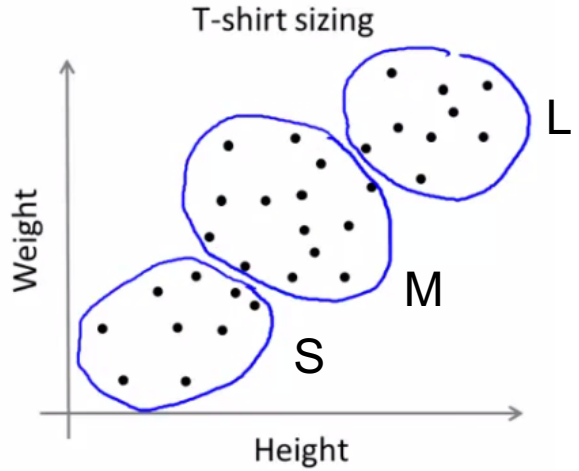
- No Labels
- No Feedback
- "Find Hidden Structure"

- Decision Process
- Reward System
- Learn Series of actions

MACHINE LEARNING | TYPES OF LEARNING







Regression

Linear & Polynomial Regression, Decision Tree, Random Forest

Classification

Naïve Bayes, Support Vector Machine(SVM), K Nearest Neighbor

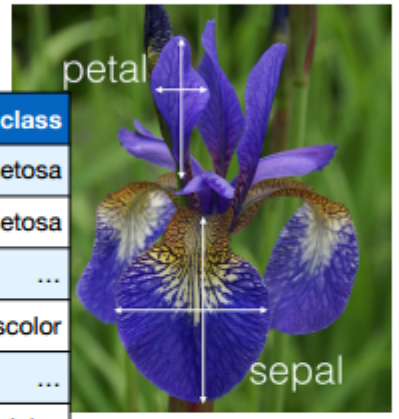
Clustering

K-Means, PCA, Singular Value Decomposition (SVD)

MACHINE LEARNING | NOMENCLETURES

Instances (samples, observations)

	sepal_length	sepal_width	petal_length	petal_width	class
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
...	...	...	...	...	...
50	6.4	3.2	4.5	1.5	veriscolor
...	...	...	...	...	...
150	5.9	3.0	5.1	1.8	virginica



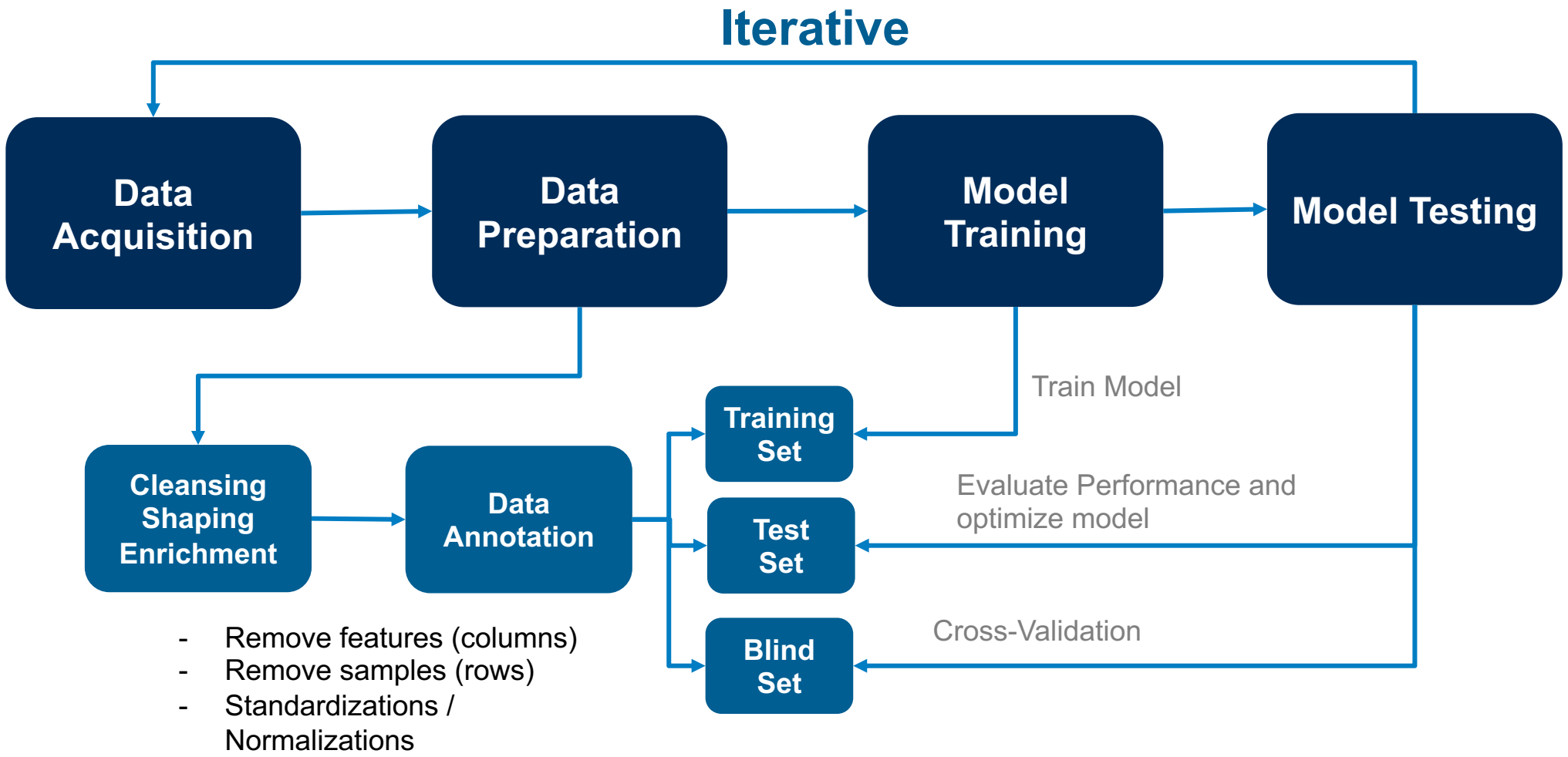
Features (attributes, dimensions)

Classes (targets)

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1D} \\ x_{21} & x_{22} & \cdots & x_{2D} \\ x_{31} & x_{32} & \cdots & x_{3D} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{ND} \end{bmatrix}$$

$$\mathbf{y} = [y_1, y_2, y_3, \cdots y_N]$$

MACHINE LEARNING | TYPICAL FLOW DIAGRAM



OVERVIEW OF SPECIFIC MACHINE LEARNING METHODS

- ✓ Linear Regression
- ✓ Polynomial Regression
- ✓ Logistic Regression
- ✓ Decision Tree
- ✓ Naive Bayes
- ✓ Support Vector Machine
- ✓ K-Nearest Neighbour
- ✓ SVD / PCA

“CLASSICAL” REGRESSION

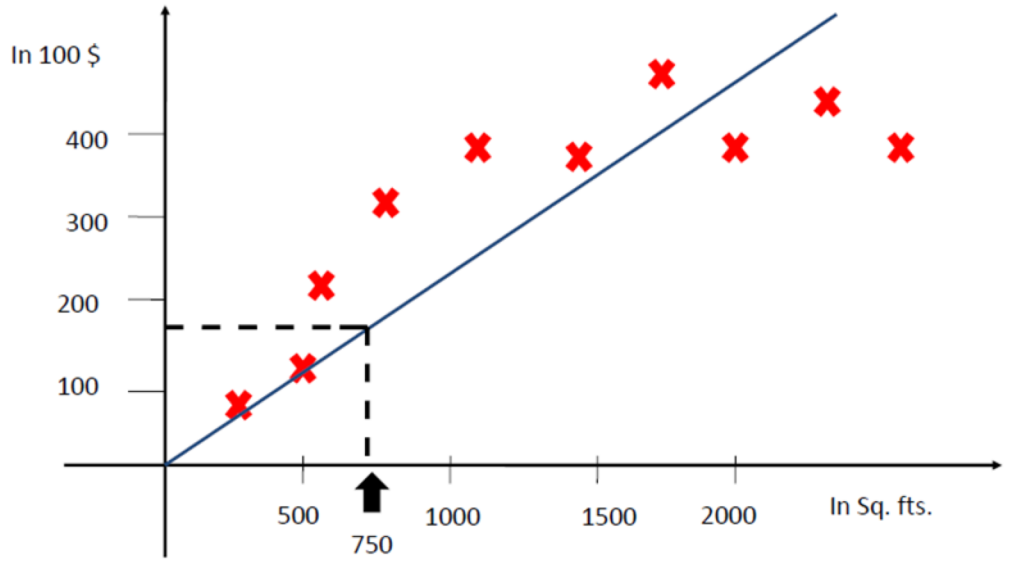
$$y_{pred} = \theta_0 + \theta_1 X$$

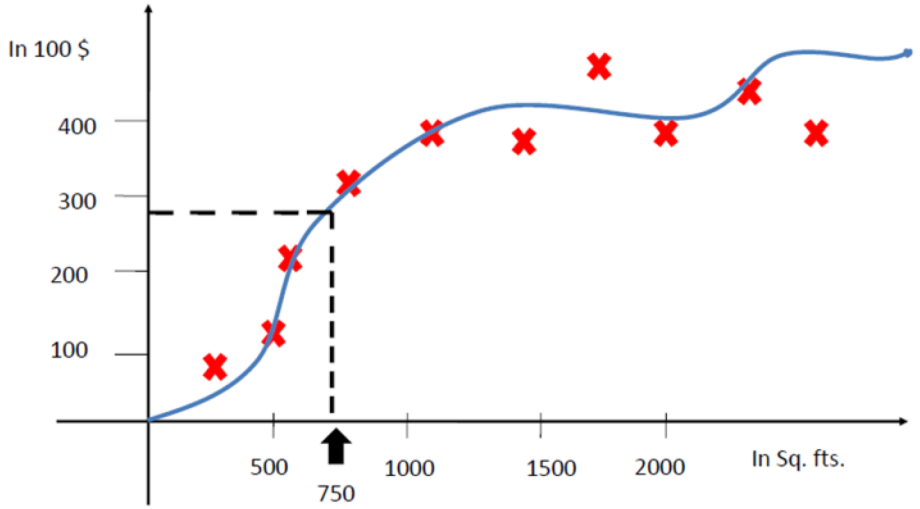
```
model= LinearRegression()
model.fit(X,y)

y_predicted = model.predict(X)
```

PREDICT | LINEAR REGRESSION

Predict housing price given the sq. ft.





n – number of training sample
p – number of features/attributes
X – input variable
Y – output variable
(X(i), Y(i)) – ith training sample

In general –

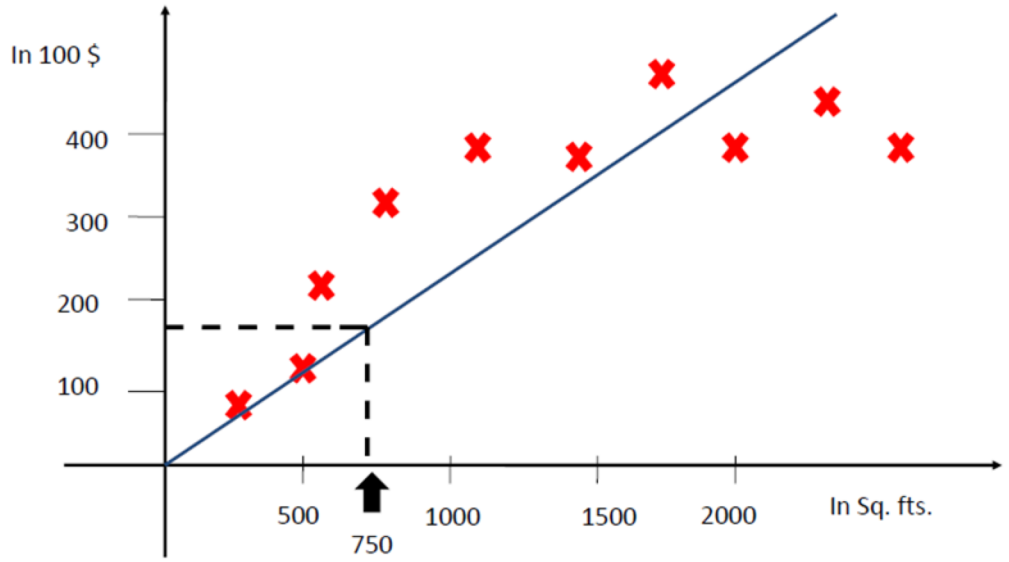
$$Y = (y_1, \cdots, y_n)^T, \qquad n \times 1$$

$$X = (X_1, \cdots, X_p), \qquad n \times p$$

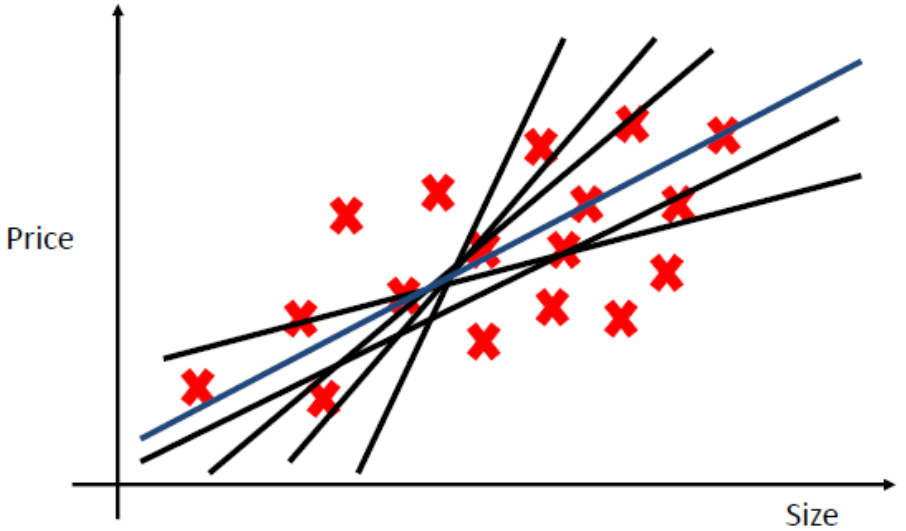
find hypothesis h that is linear combination and use $\hat{y} = x^T$.h to predict y

PREDICT LINEAR REGRESSION

Predict housing price given the sq. ft.

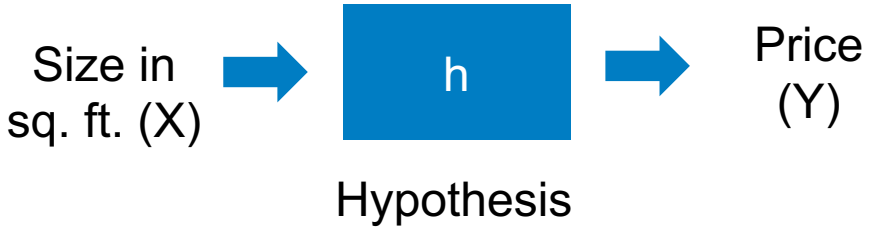


$h(X) = \Theta_0 + \Theta_1 X$



What is the best value of Θ_0 and Θ_1

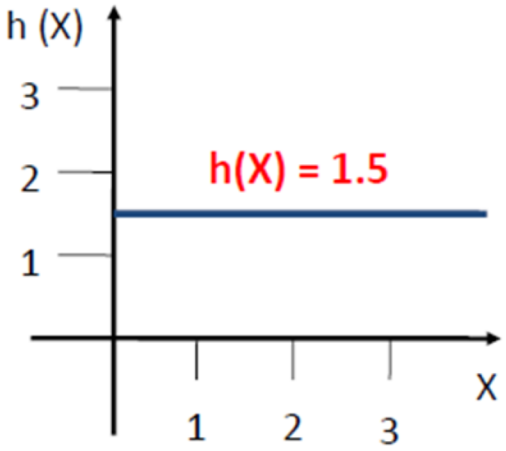
n – number of training sample
p – number of features/attributes
X – input variable
Y – output variable
(X(i), Y(i)) – ith training sample



PREDICT | LINEAR REGRESSION

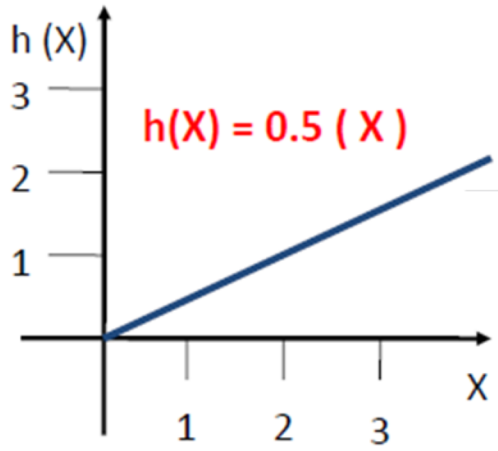
What is the best value of Θ_0 and Θ_1

$$h(X) = \Theta_0 + \Theta_1 X$$



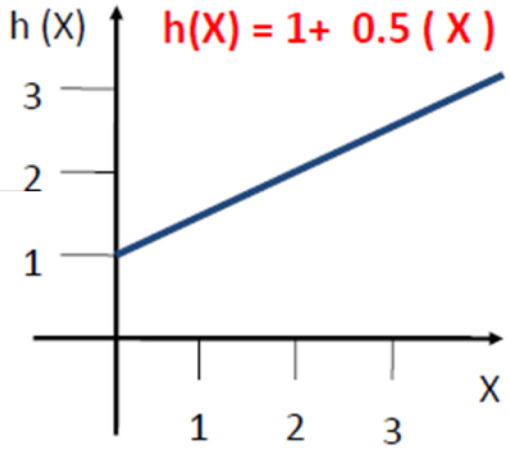
$\Theta_0 = 1.5$

$\Theta_1 = 0$



$\Theta_0 = 0$

$\Theta_1 = 0.5$



$\Theta_0 = 1$

$\Theta_1 = 0.5$

Root Mean Error Square

Cost Function

$$\underset{\text{minimize}}{J(\Theta_0, \Theta_1)} = \frac{1}{2m} \sum_{i=1}^{m} (h_\Theta(X^i) - Y^i)^2$$

Choose the value of Θ_0 and Θ_1 such that $h_\Theta(X)$ is close to Y, for all training set

minimize $(h_\Theta(X) - Y)^2$

Predicted Given

LINEAR REGRESSION

Cost Function

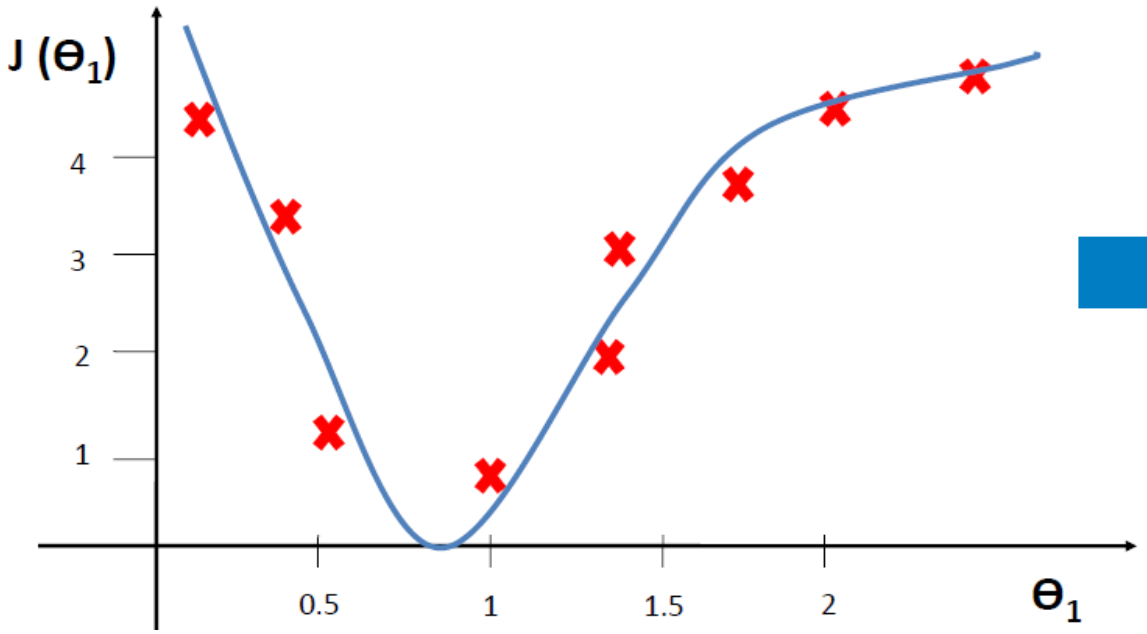
$$\underset{\text{minimize}}{J(\theta_0, \theta_1)} = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(X^i) - Y^i)\text{^}2$$

What is the best value of θ_0 and θ_1

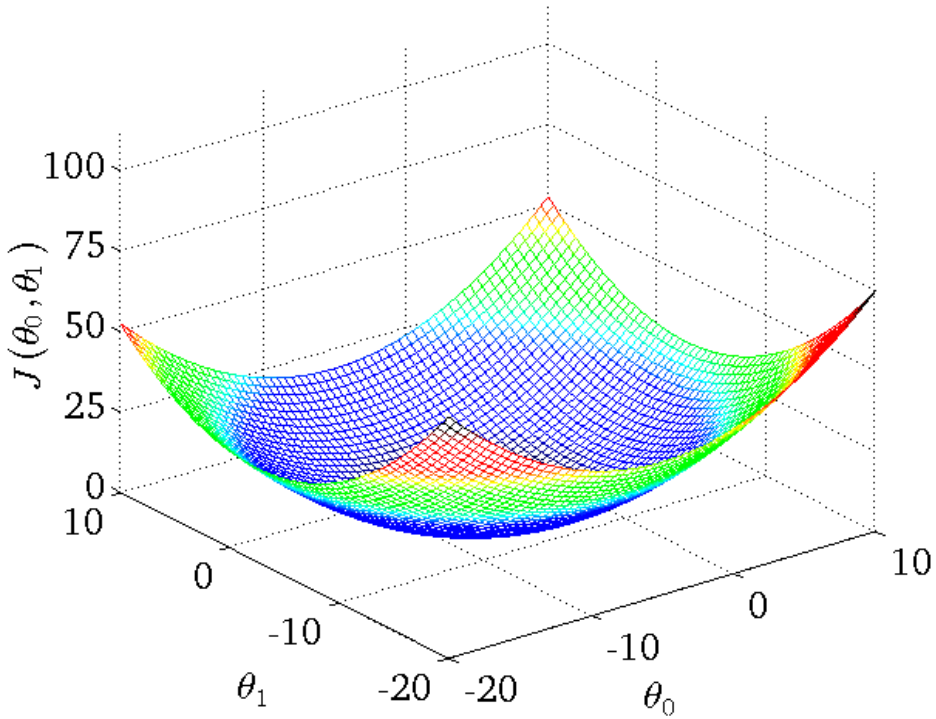
$$h(X) = \theta_0 + \theta_1 X$$

For $\theta_0 = 0$

Cost function Vs θ_1



Cost function Vs θ_0 & θ_1



Large number of iterations to find local minimum

PREDICT | **LINEAR REGRESSION**

What is the best value of Θ_0 and Θ_1

$h(X) = \Theta_0 + \Theta_1 X$

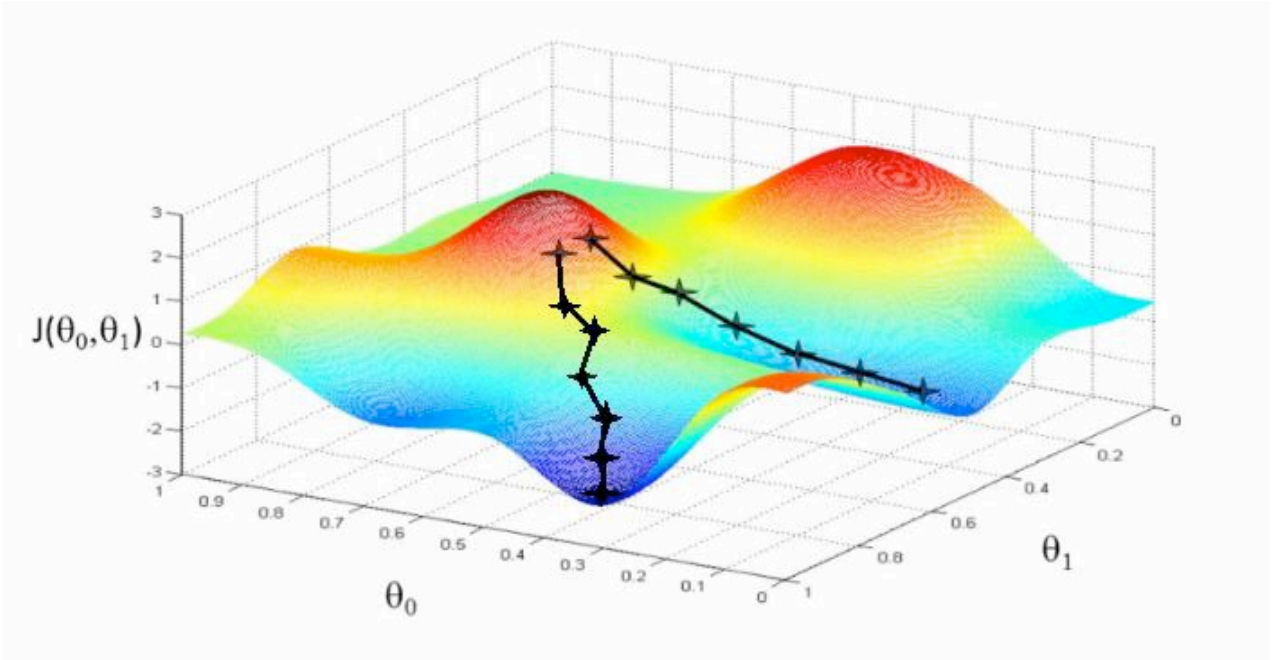
Cost Function

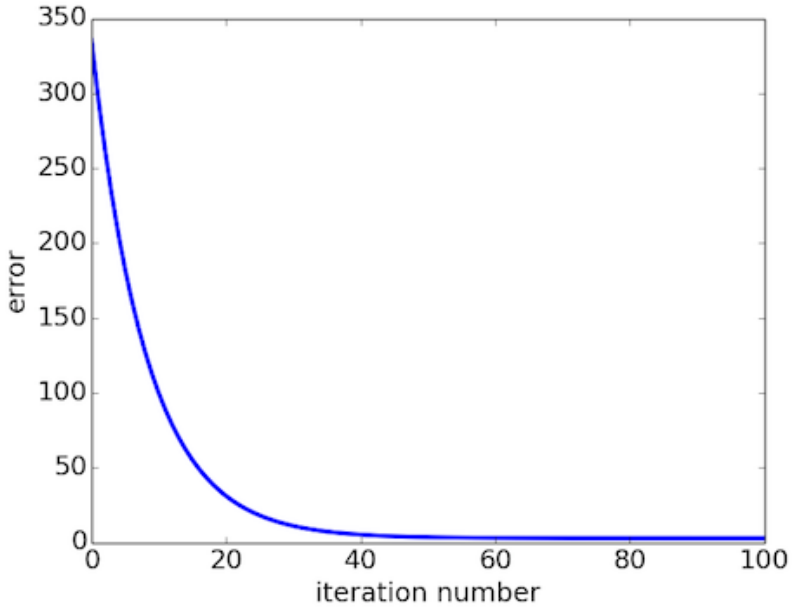
repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

Gradient Descent Algorithm



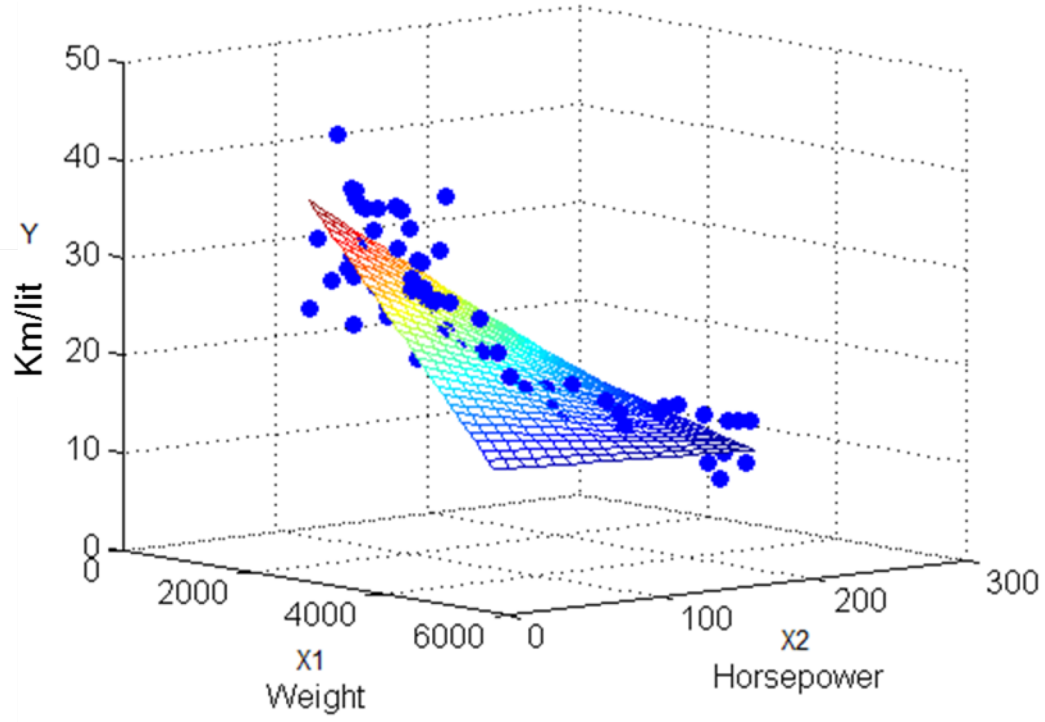


Error decreases with every iterations as it takes steps to negative gradient and hence much faster

PREDICT MULTIPLE LINEAR REGRESSION

What is the best value of Θ_o ,Θ_1,Θ_2

$h(X) = \Theta_o + \Theta_1 X1 + \Theta_2 X2$

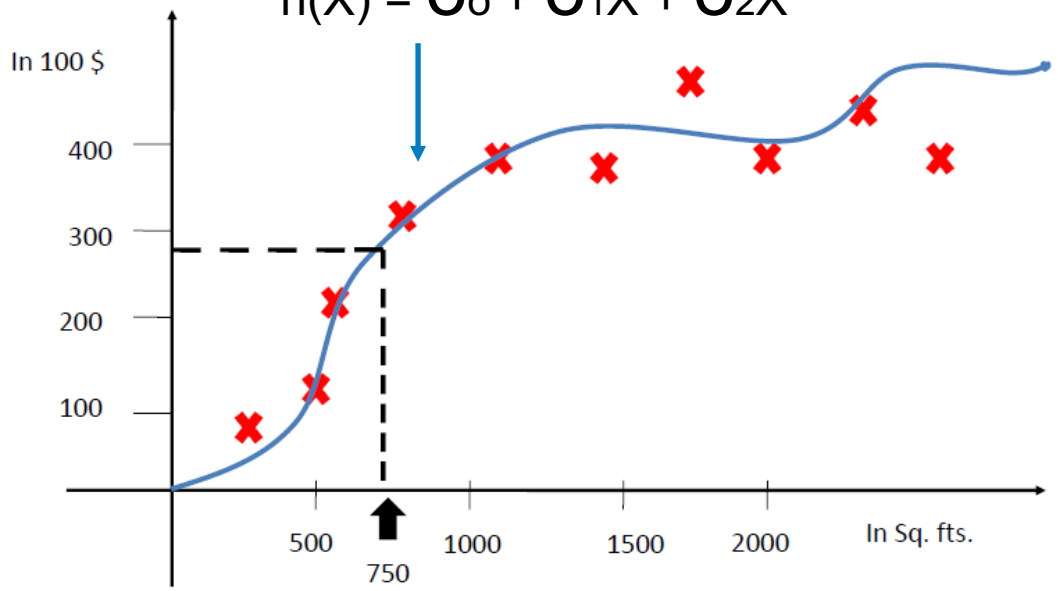


PREDICT | POLYNOMIAL REGRESSION

Demo

What is the best value of Θ_o ,Θ_1,Θ_2

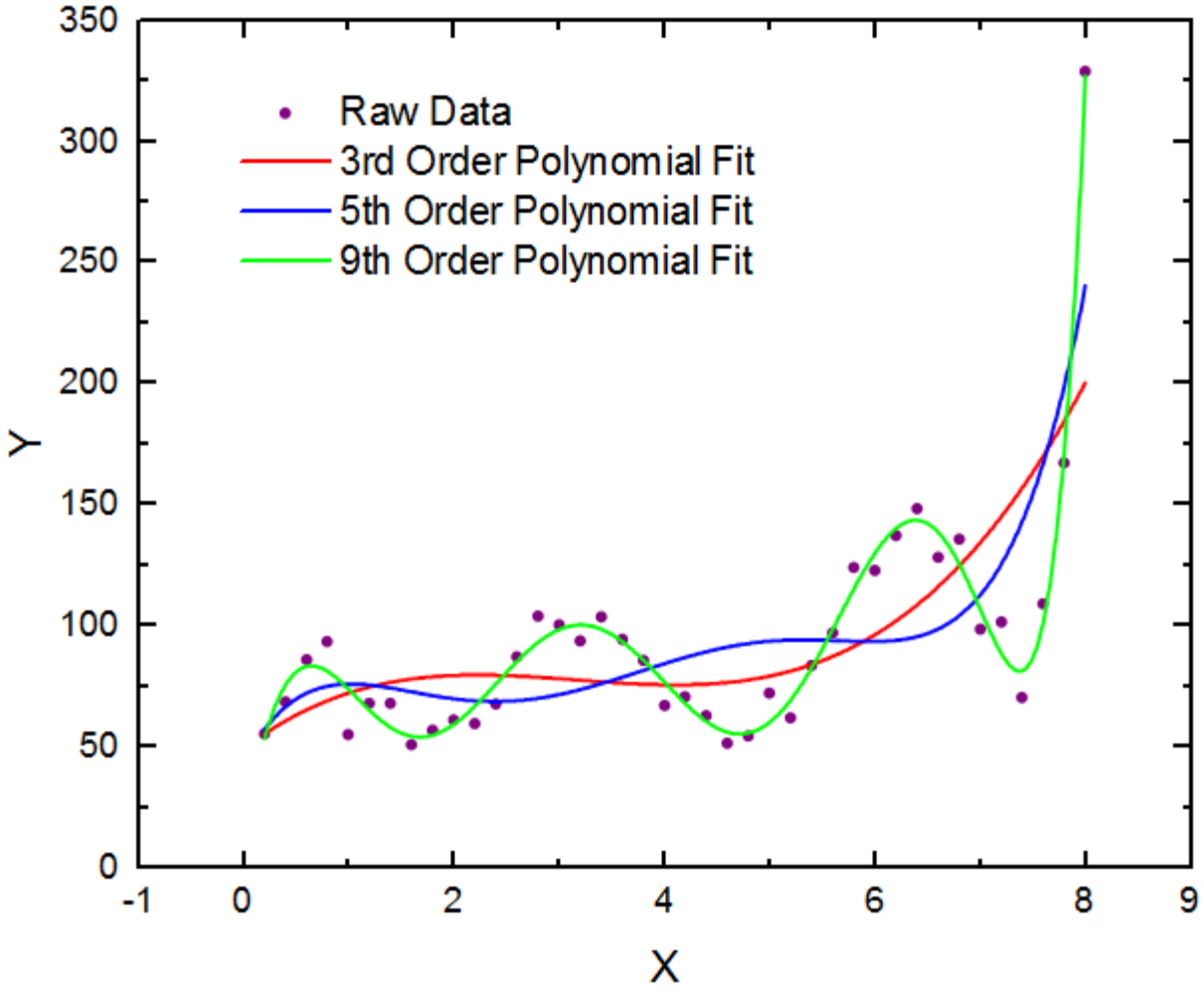
$$h(X) = \Theta_o + \Theta_1 X + \Theta_2 X^2$$




```
quadratic_featurizer = PolynomialFeatures(degree)
X_quadratic = quadratic_featurizer.fit_transform(X)

regressor_quadratic = LinearRegression()
regressor_quadratic.fit(X_quadratic, y)
```

PREDICT | BIAS , VARIANCE AND OVERFITTING



With high variance the data is more spread out

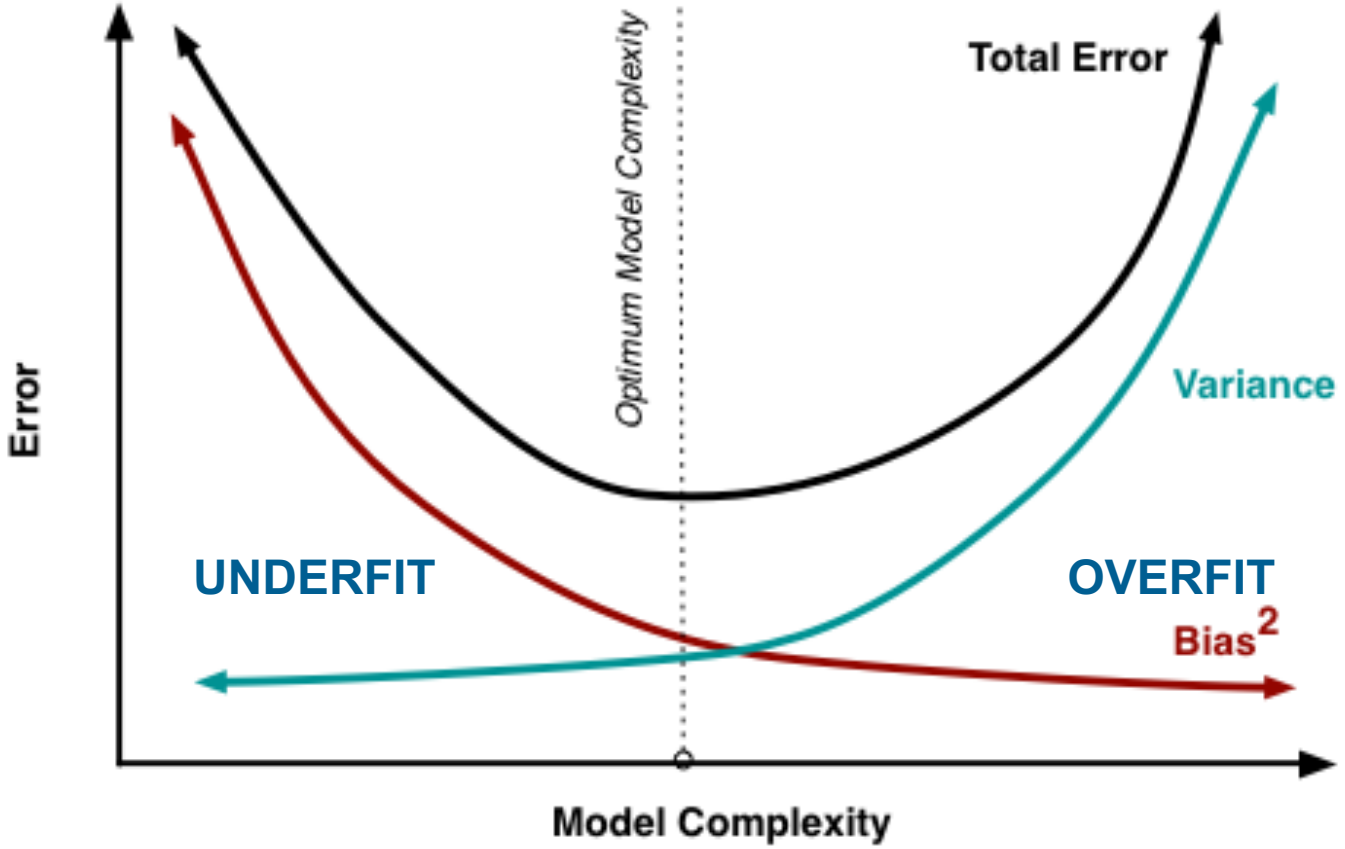
With high Bias the data is “biased” away from center

Bulls eye reflect the perfect Model

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_{d-1} x^{d-1} + a_d x^d$$

BIAS , VARIANCE AND OVERFITTING

Bias and Variance Trade OFF



- As the degree gets higher the function becomes more complex
- Low dimensional polynomial has high bias – it under fits the data
- High dimensional data may fit for train data but gives more variance for future prediction – it over fits the data

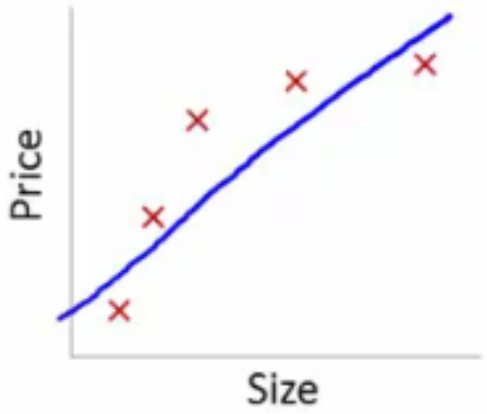
Mean square error:

$$r(d,\theta) = E[(d-\theta)^2] = (E[d] - \theta)^2 + E[(d-E[d])^2]$$

$$= \text{Bias}^2 + \text{Variance}$$

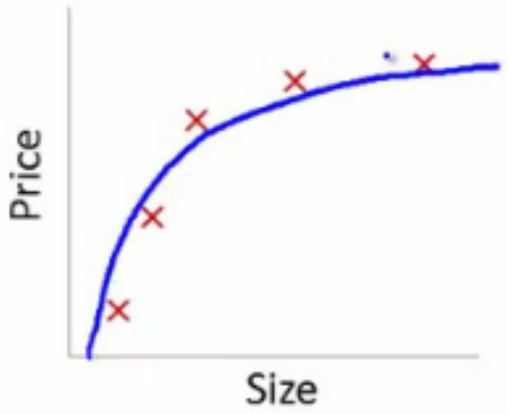
PREDICT | RIGHT FIT SOLUTION

Demo



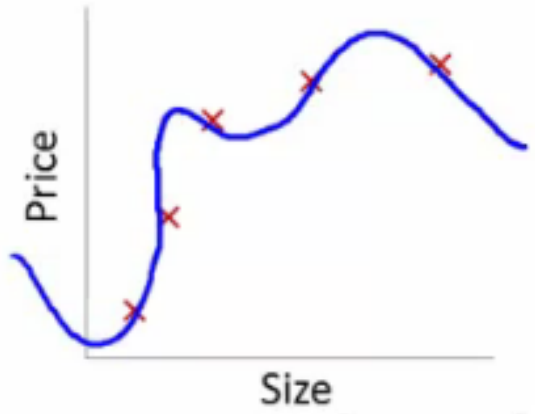
$\theta_0 + \theta_1 x$

High bias
(underfit)



$\theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

High variance
(overfit)

REGULARISATION | RIDGE AND LASSO

Demo

- When large number of features are present with many of them correlated , we use regularization techniques to solve over-fitting problem.
- It does so by penalizing the bent of the regression line that tries to closely match the noisy data points. (penalizing the magnitude of coefficients of polynomials)
- It leads to smoothening of the regression line and thus prevents over-fitting
- You penalize your loss function by adding a multiple of an L1 (LASSO) or an L2 (Ridge) norm of your weights vector w

RIDGE Regression (L2)

Performs L2 regularization, i.e. adds penalty equivalent to **square of the magnitude** of coefficients

Ridge Regression

$$\min_w ||Xw - y||_2^2 + \alpha||w||_2^2$$

```
from sklearn.linear_model import Ridge
ridgereg = Ridge(alpha=0, normalize=True)
ridgereg.fit(X_train, y_train)
y_pred = ridgereg.predict(X_test)
```

LASSO Regression (L1)

Performs L1 regularization, i.e. adds penalty equivalent to **absolute value of the magnitude** of coefficients

Lasso Regression

$$\min_w \frac{1}{2n_{samples}}||Xw - y||_2^2 + \alpha\rho||w||_1 + \frac{\alpha(1-\rho)}{2}||w||_2^2$$

```
from sklearn.linear_model import Lasso
model = Lasso(alpha=0.001, normalize=True)
model.fit(X_train, y_train)
```

PREDICT | LOGISTIC REGRESSION

Often we have to resolve questions with binary or yes/no outcomes. For example:

Does a patient have cancer?
Will the customer buy my product?

Will a team win the next game?
Will I get the loan?

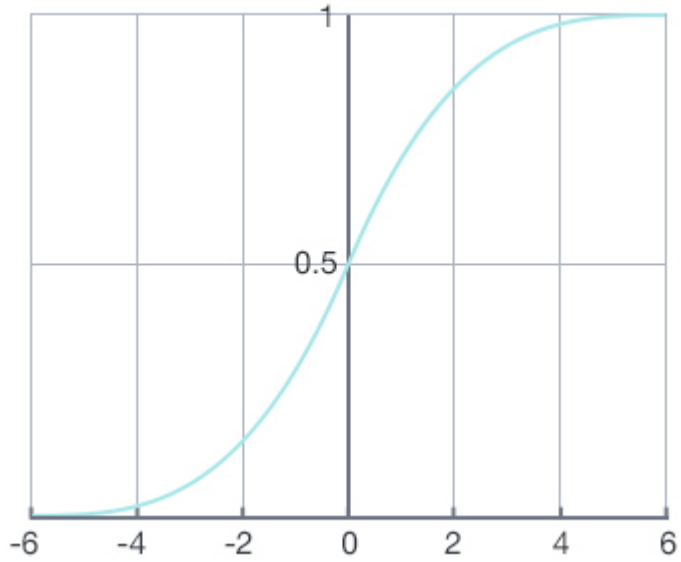
It is used to predict value that leads to classify it into two classes. This is classification but since it makes use of regression with logistic function it is called logistic regression.

Linear regression : $h(X) = \Theta_0 + \Theta_1 X$

Logistic regression : $h(X) = g(\Theta_0 + \Theta_1 X)$ so that $0 < h(X) < 1$

Where g is **sigmoid function** (logistic function)

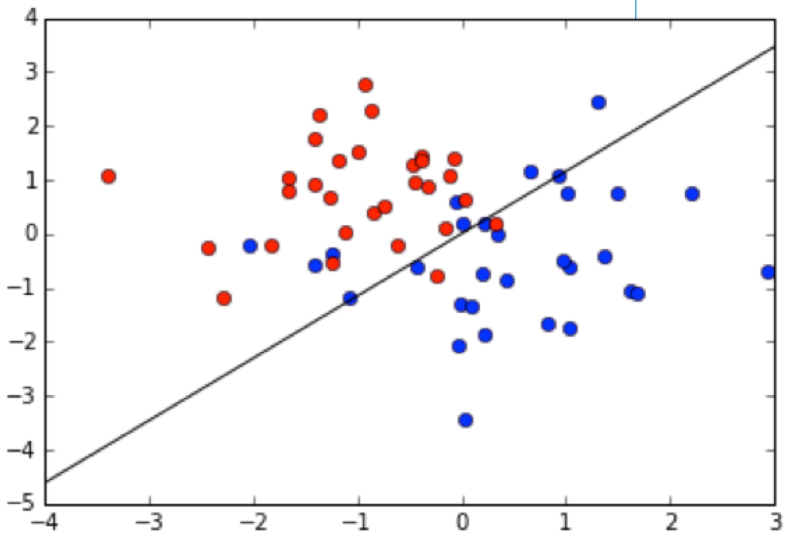
$$g(z) = \frac{1}{1+e^{-z}}$$

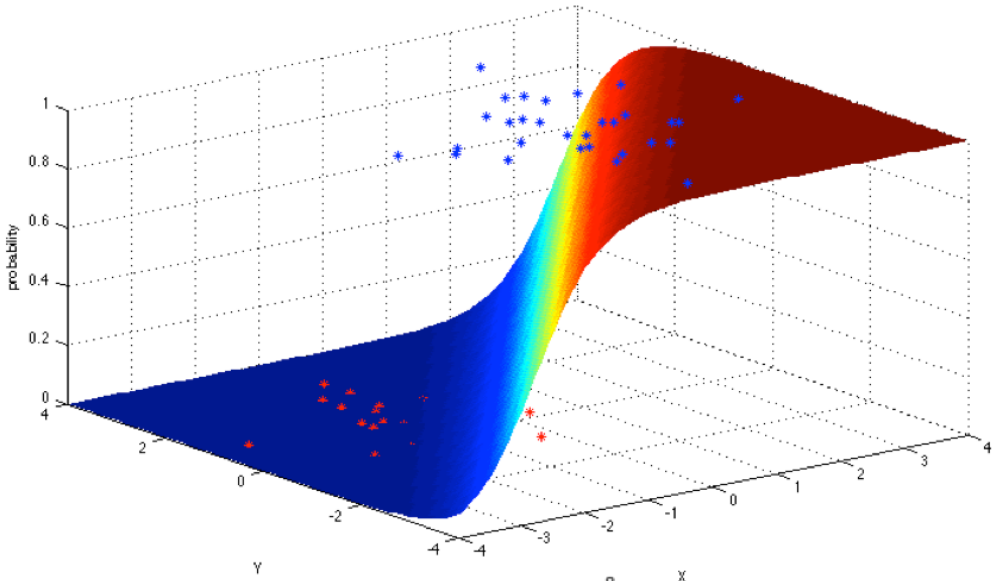


Applying linear regression model for classification is not great idea

PREDICT | LOGISTIC REGRESSION

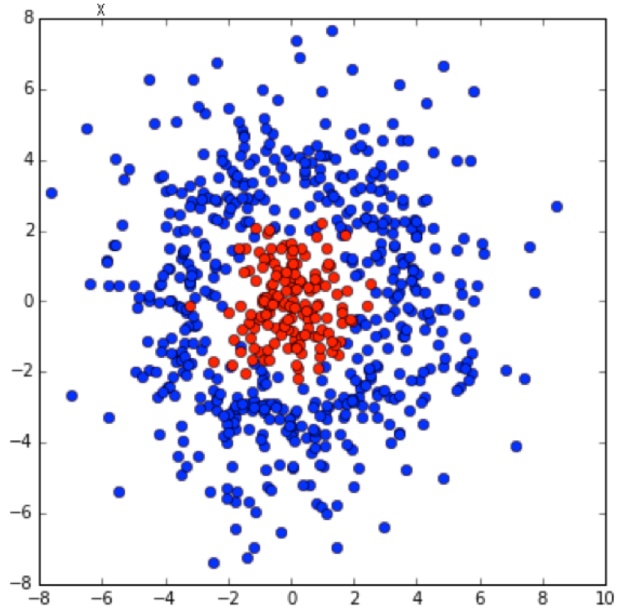
Demo





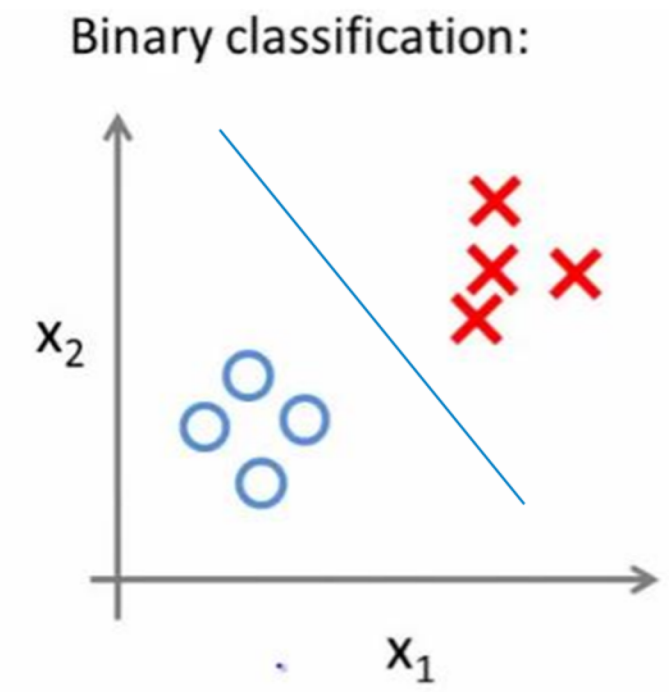
A logistic regression model makes a linear boundary between the two classes

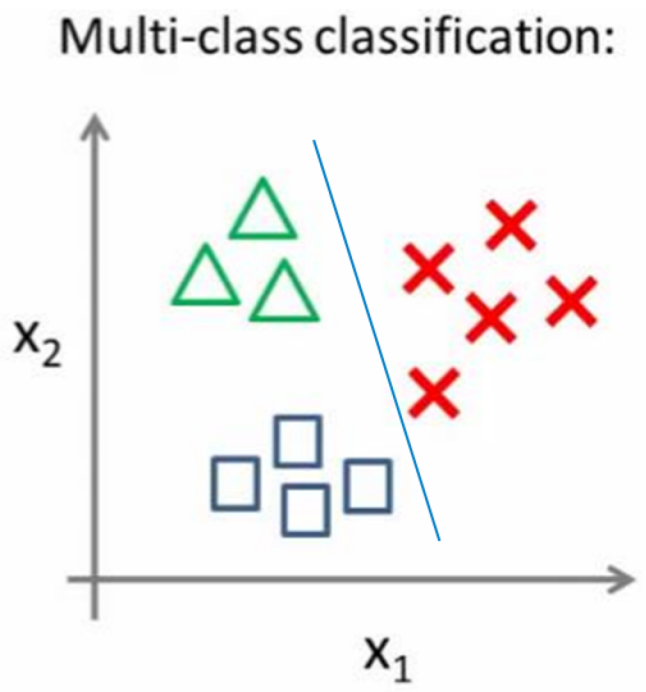
Logistic regression cannot draw a more complicated boundary than a line and hence not suitable for data set with many features

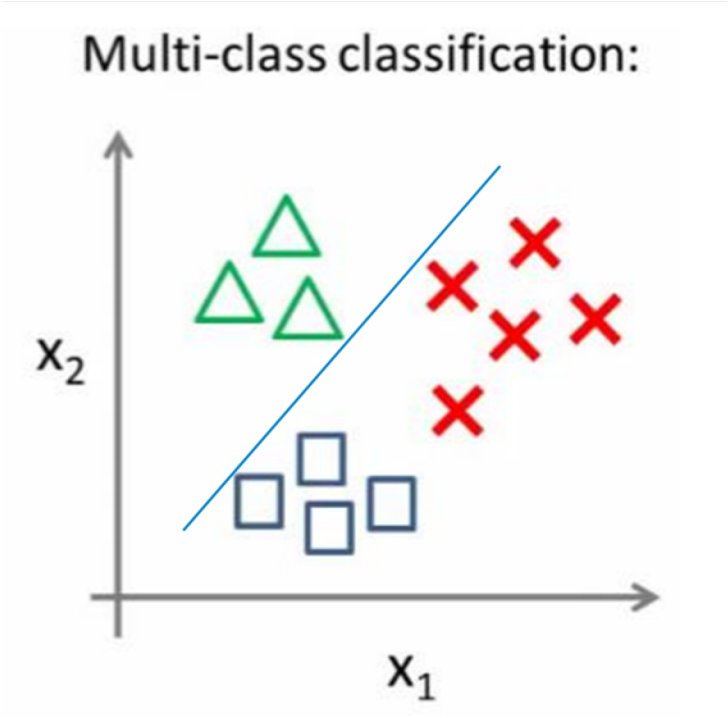


```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X,y)
```

PREDICT | MULTICLASS LOGISTIC REGRESSION







MODEL PERFORMACNE

ACCURACY, PRECISION, RECALL & F1 SCORE

Precision

- How often does our algorithm have false positives

= true positives / # predicted positive
= true positives / (true positive + false positive)

- High precision is good (i.e. closer to 1)
- You want a big number, because you want false positive to be as close to 0 as possible

$$P = \frac{TP}{TP + FP}$$

Accuracy:

$$acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall

- How sensitive is our algorithm?

Of all patients in set that actually have cancer, what fraction did we correctly detect
= true positives / # actual positives
= true positive / (true positive + false negative)

- High recall is good (i.e. closer to 1)
- You want a big number, because you want false negative to be as close to 0 as possible

$$R = \frac{TP}{TP + FN}$$

F1Score (fscore)

Fscore is like taking the average of precision and recall giving a higher weight to the lower value

$$F1 = 2\frac{PR}{P + R}$$

DECISION TREES

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X,y)
```

Tuning Parameters :

criterion = 'gini', 'entropy'

max_depth = 1, 2, 3..

DECISION TREES

How does it work? A simple example

Suppose we have the given data set

✓ 9 - Yes
✓ 5 - No

We want to know if Sachin will play golf or not on Day-15?

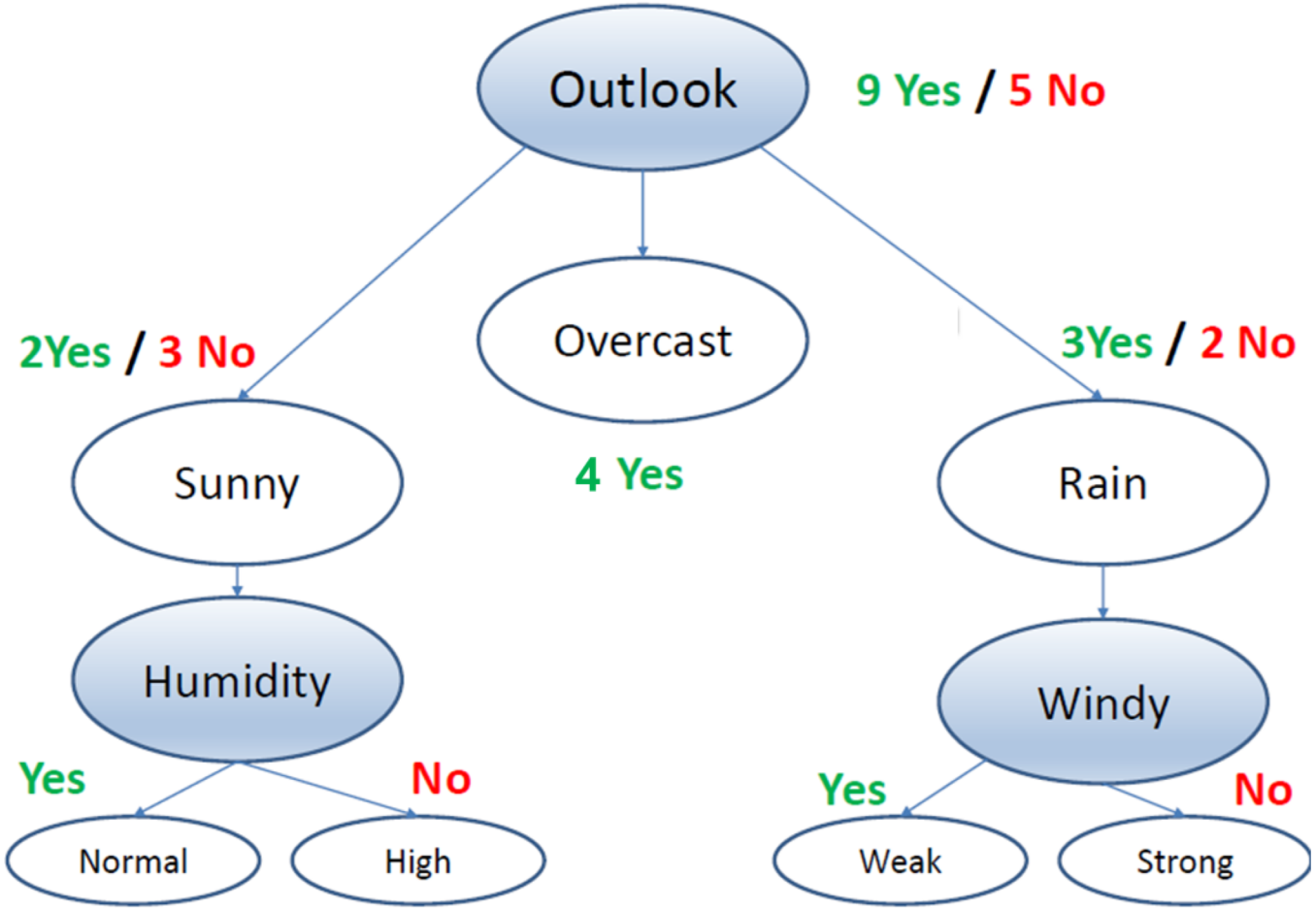
Training Samples

9 Yes /5 No

Day	Outlook	Humidity	Windy	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No
D15	Rain	High	Weak	?

DECISION TREES

- If outlook is overcast , Sachin always plays a golf
- If outlook is sunny and humidity is high , he will not play
- If outlook is Rainy and there is a strong wind , he will not play



Logical Formulation:
(Outlook = Sunny ∧ Humidity=Normal)
∨ (Outlook = Overcast)
∨ (Outlook = Rain ∧ Wind = Weak)

How to split the attributes ???

ENTROPY CALCULATION

WHICH ATTRIBUTE IS BEST?

If,

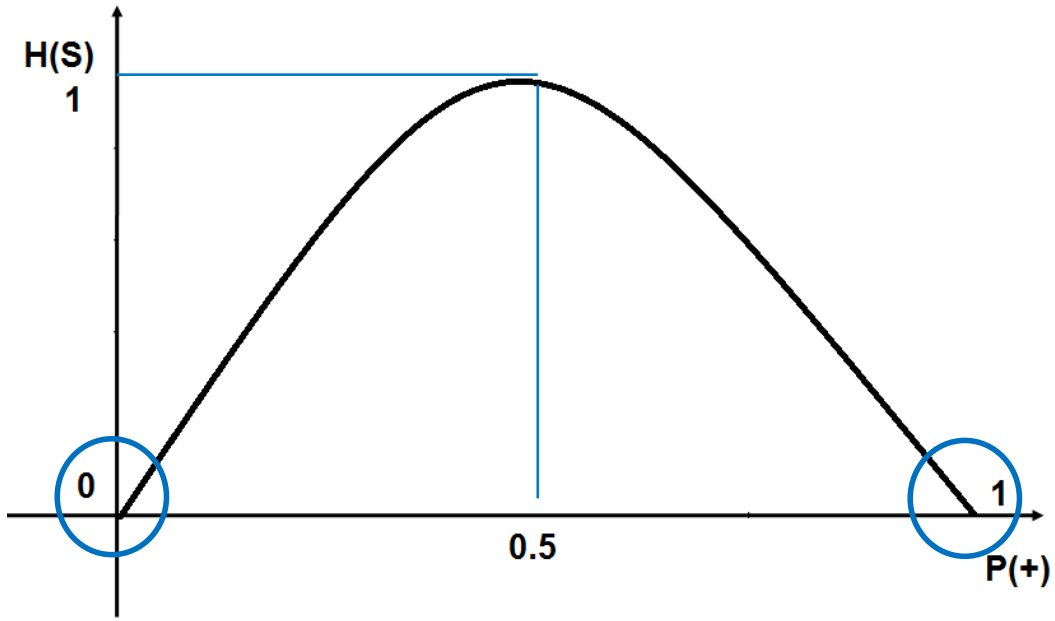
- S is a sample of training examples
- p⊕ is the proportion of positive examples in S
- pΘ is the proportion of negative examples in S

Entropy measures the impurity of S

Entropy(S) ≡ H(S) ≡ −p⊕ log2 p⊕ − pΘ log2 pΘ

Pure Set (4 Yes /0 No)
H(S) = 0 if sample is pure (all + or all -)
H(S) = -4/4log2(4/4) -0/4log2(4/4) = 0

Impure Set (3 Yes /3 No)
H(S) = -3/6log2(3/6) -3/6log2(3/6) = 1
H(S) = 1 if p⊕ = pΘ = 0.5



INFORMATION GAIN

WHICH ATTRIBUTE IS BEST?

Step-1) Calculate entropy of the target …required in step-2

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

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

= 0.94

INFORMATION GAIN WHICH ATTRIBUTE IS BEST?

Step-2) The dataset is then split on the different attributes. The entropy for each branch is calculated.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

INFORMATION GAIN

WHICH ATTRIBUTE IS BEST?

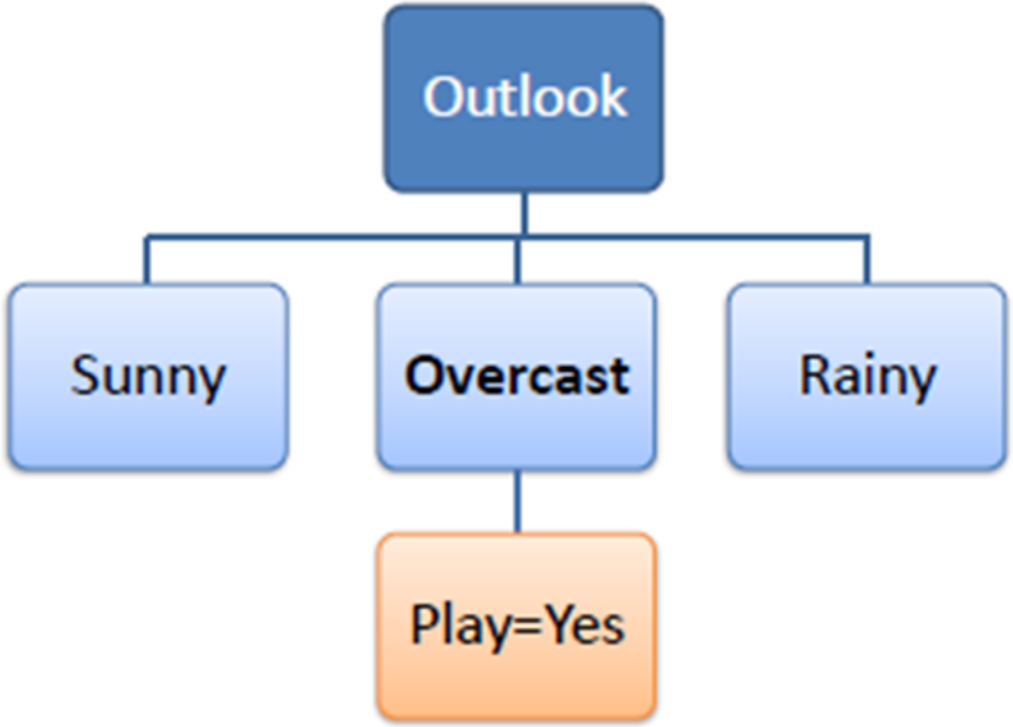
Step-3) Choose attribute with the largest information gain as the decision node.

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

INFORMATION GAIN | WHICH ATTRIBUTE IS BEST?

Step-4a) A branch with entropy of 0 is a leaf node.

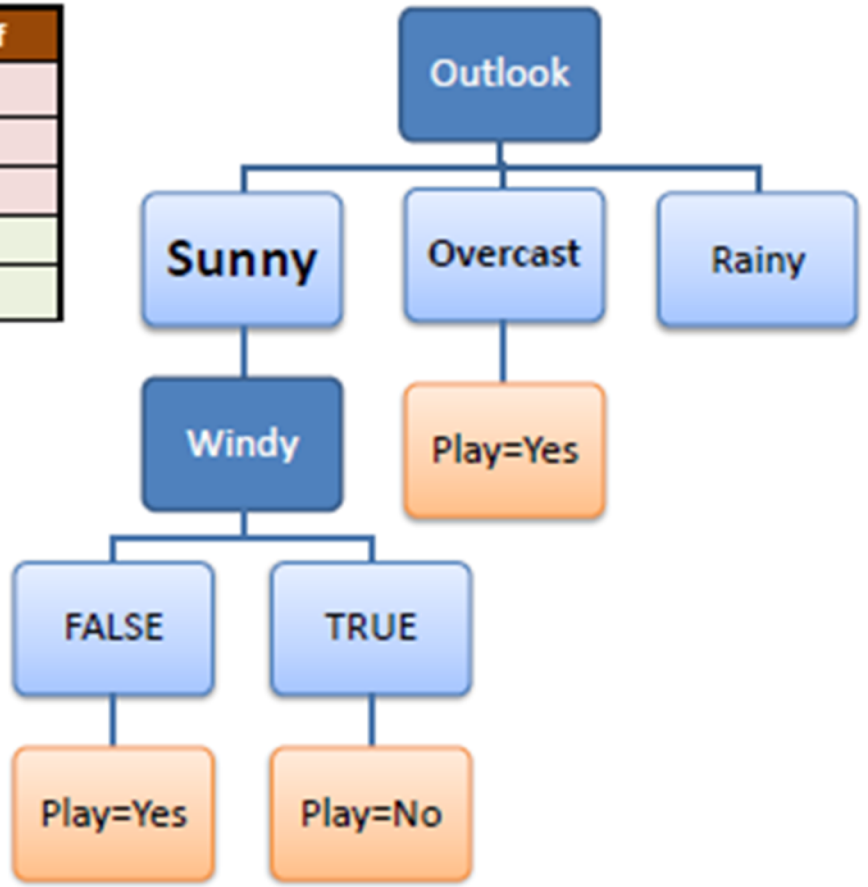
Humidity	Windy	Play Golf
High	FALSE	Yes
Normal	TRUE	Yes
High	TRUE	Yes
Normal	FALSE	Yes
High	FALSE	Yes



INFORMATION GAIN | WHICH ATTRIBUTE IS BEST?

Step-4a) A branch with entropy more than 0 needs further splitting.

Humidity	Windy	Play Golf
High	FALSE	Yes
Normal	FALSE	Yes
Normal	FALSE	Yes
Normal	TRUE	No
High	TRUE	No



INFORMATION GAIN WHICH ATTRIBUTE IS BEST?

Step-5 – The algorithm is run recursively on the non-leaf branches, until all data is classified.

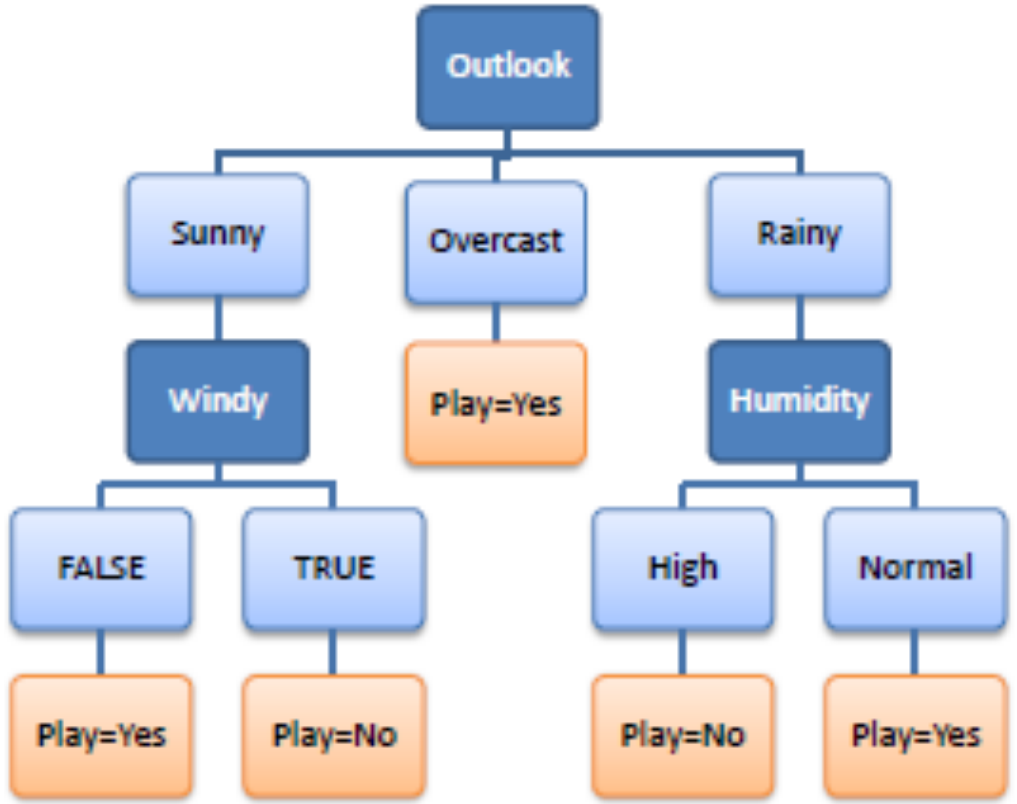
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

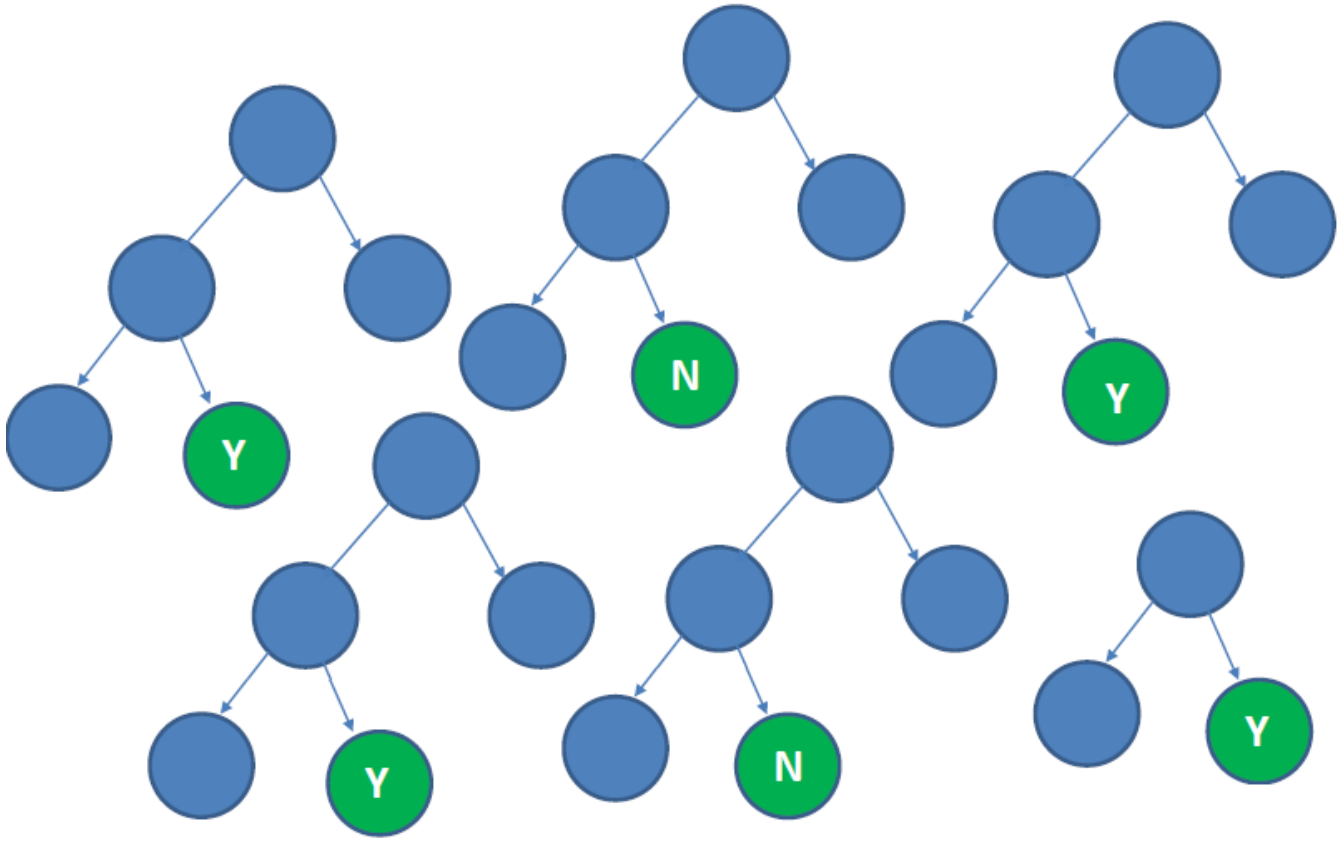
R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



RANDOM FOREST | HOW IT WORKS?

Demo

- An ensemble classifier using many decision tree models.
- In a dataset, where M is the total number of input attributes in the dataset, only m attributes are chosen at random for each tree where m<M



Not suitable for prediction of continuous attribute.

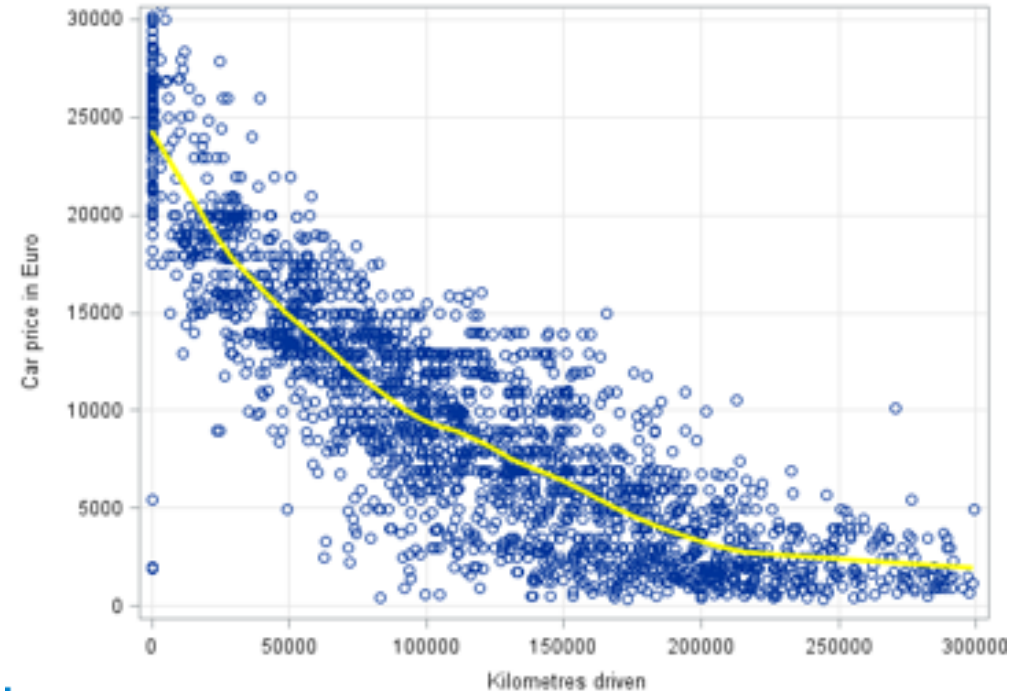
Decision trees pros and cons

pros

- Interaction between variables
- Interpretable rules
- Missing values easy to incorporate.

cons

- May become Unstable if data set is improper
- “Lack-of-Smoothnes”
- prone to errors in classification, owing to differences in perceptions



NAIVE BAYES CLASSIFIER

BAYES THEOREM

Likelihood of evidence 'X' if Hypothesis 'C' is true

Prior Probability

$$P(c \mid X) = \frac{P(X \mid C)\ P(C)}{P(X)}$$

Posterior Probability of 'C' Given the evidence X

Priori probability that the evidence itself is true

Frequency Table

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3

NAIVE BAYES | WILL SACHIN PLAY ?

P (c|X) = $P(X|C) * P(C) / P(X)$

P (Yes|Sunny) = $P(Sunny|Yes) * P(Yes) / P(Sunny)$
= (3/9) * (9/14) / (5/14)

$$P(x \mid C) = P (Sunny \mid yes) = {}^{3}/_{9}$$

		Playing Golf		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	2/9	3/5	5/14
		9/14	5/9	

$$P(x) = P (Sunny) = {}^{5}/_{14}$$

$$P(C) = P (Yes) = {}^{9}/_{14}$$

NAIVE BAYES | WILL SACHIN PLAY ?

Outlook : Rainy

Humidity : Normal

Windy : Strong

Will Sachin play ?

Likelihood of Yes =

P (Outlook: Rainy | Yes) * P (Humidity: Normal | Yes) * P (Windy: Strong | Yes)* P (Yes)

Likelihood of No =

P (Outlook: Rainy | No) * P (Humidity: Normal | No) * P (Windy: Strong | No)* P (Yes)

$$P(yes) = \frac{\text{Likelihood of } Yes}{(\text{Likelihood of Yes} + \text{Likelihood of No})}$$

$$P(No) = \frac{\text{Likelihood of } No}{(\text{Likelihood of Yes} + \text{Likelihood of No})}$$

NAIVE BAYES | BASED ON DISTRIBUTION OF FEATURES

Demo

- **GaussianNB**: The features follow a normal distribution. Instead of discrete counts, we have continuous features
- **MultinomialNB**: Text classification data are typically represented as word vector counts
- **BernoulliNB**: The binomial model is useful if your feature vectors are binary

SUPPORT VECTOR MACHINE (SVM)

```
from sklearn import svm
model = svm.SVC()
model.fit(X,y)
```

Tuning Parameters :

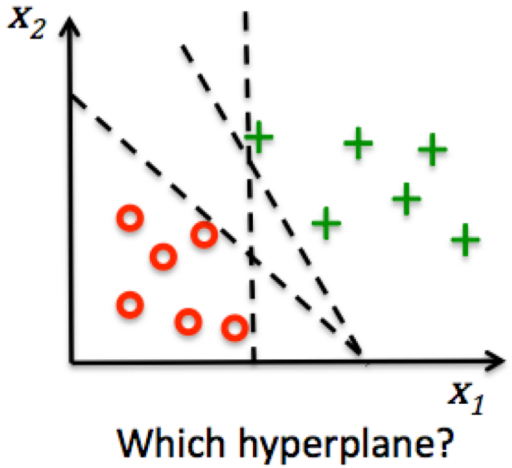
kernel = ‘ linear ’, C = 1 or 0.1 etc

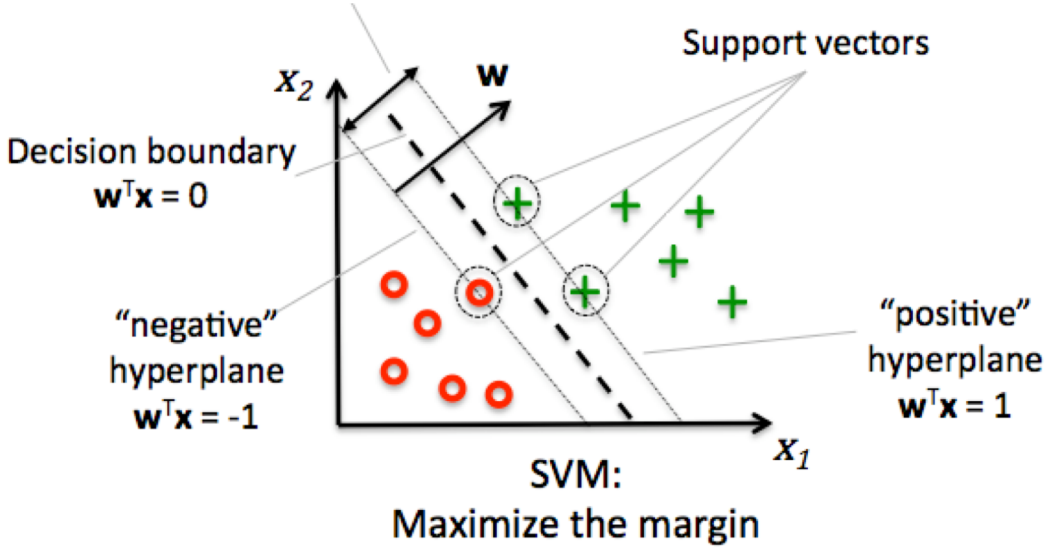
kernel = ‘ rbf ’, C , gamma=0, 0.1 or 10

SVM | HOW IT WORKS

Demo

A Support Vector Machine (SVM) performs classification by constructing an *N*-dimensional hyper plane that optimally separates the data into two categories.





Generaly SVM deals with

- more than two attribute variables
- separating the points with non-linear curves
- handling the cases where clusters cannot be completely separated
- handling classifications with more than two categories.

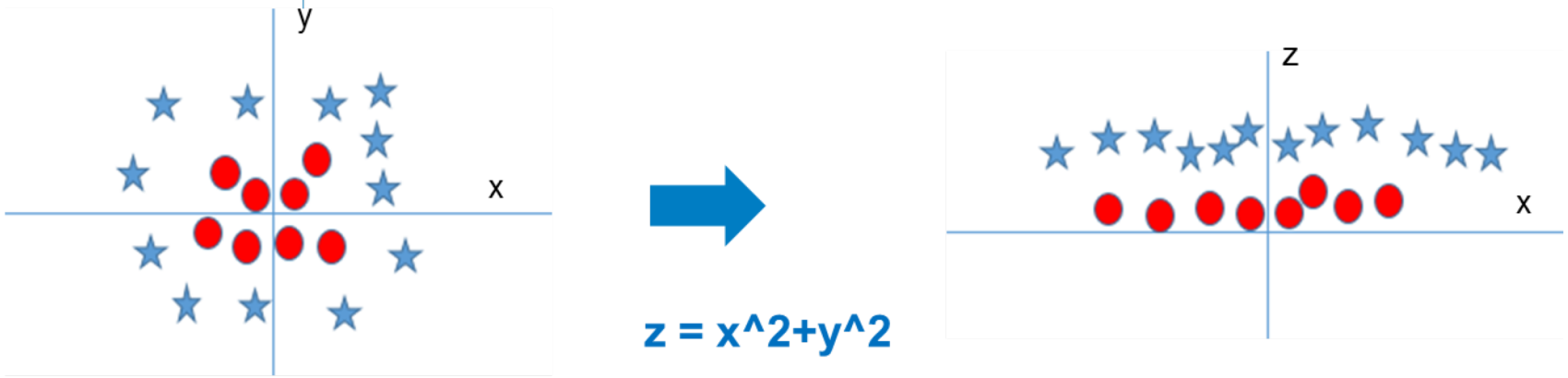
Linear SVM $x_i \cdot x_j$

Non-linear SVM $\phi(x_i) \cdot \phi(x_j)$

Kernel function $k(x_i \cdot x_j)$

SVM KERNEL TRICK

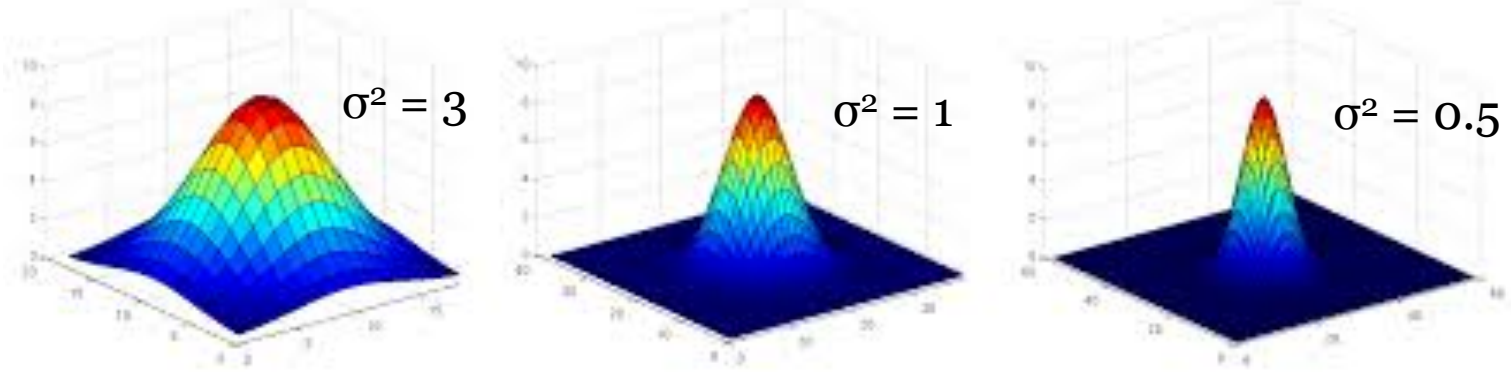
Demo



Kernels are functions which takes low dimensional input space and transform it to a higher dimensional space.

It is mostly useful in non-linear separation problem.

It converts not separable problem to separable problem.



σ² is a Gaussian parameter

* You can develop your own basis function

K-NEAREST NEIGHBORS (KNN) CLASSIFIER

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(X,y)
```

Tuning Parameters :

metric = 'minkowski', Euclidean, Manhattan

n_neighbors = 3, 5,7,9..

KNN | HOW IT WORKS ?

K nearest neighbors is a simple algorithm that stores all available instances and classifies new instance based on a similarity measure (e.g., distance functions).

- Decide K (generally odd number)
- Calculate the distance between any two points
- Find the nearest neighbors based on these pairwise distances
- Majority vote on a class labels based on the nearest neighbor list of K distances

Demo

KNN | HOW IT WORKS ?

Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^{k}|x_i - y_i|$$

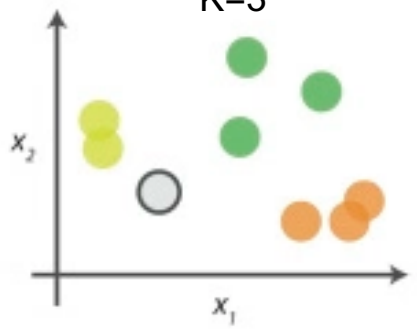
Minkowski

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

Also called as Lazy learning

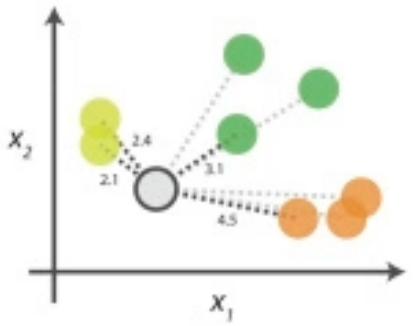
0. Look at the data

K=3



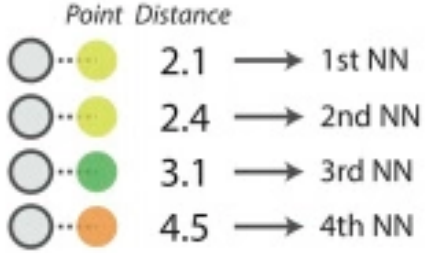
Say you want to classify the grey point into a class. Here, there are three potential classes - lime green, green and orange.

1. Calculate distances



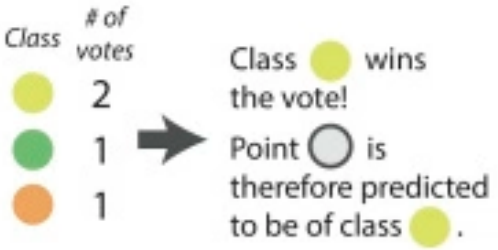
Start by calculating the distances between the grey point and all other points.

2. Find neighbours



Next, find the nearest neighbours by ranking points by increasing distance. The nearest neighbours (NNs) of the grey point are the ones closest in dataspace.

3. Vote on labels



Vote on the predicted class labels based on the classes of the k nearest neighbours. Here, the labels were predicted based on the k=3 nearest neighbours.

UNSUPERVISED LEARNING

CLUSTERING K MEANS

K-MEANS CLUSTERING

HOW IT WORKS ?

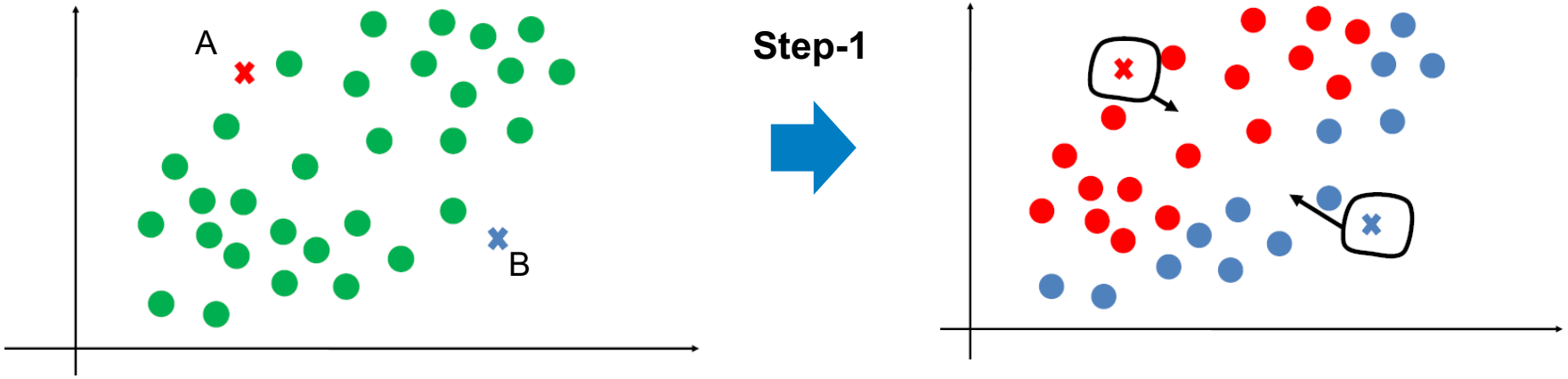
Simple and easy way to classify a given data set through a certain number of clusters (k clusters)

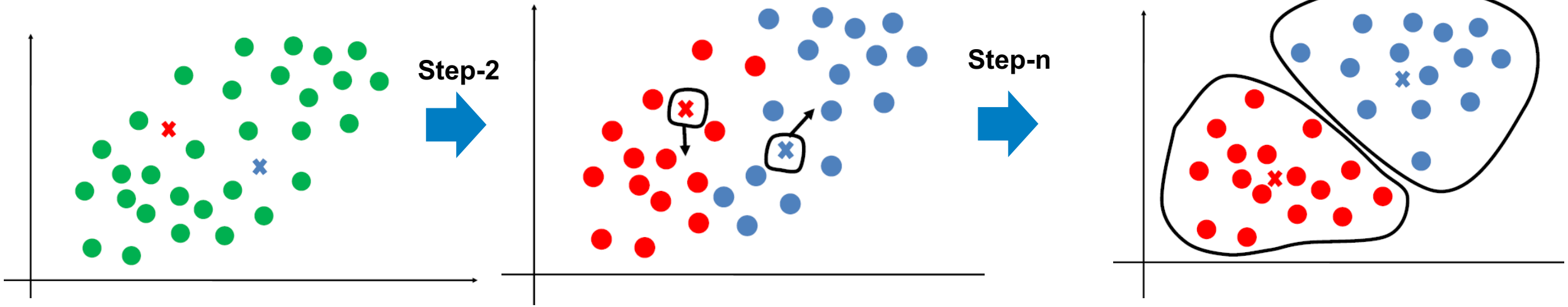
- Define k centers randomly , one for each cluster
- Take each point belonging to a given data set and associate it to the nearest center
- Re-calculate k new centroids as barycenter of the clusters resulting from the previous step
- Repeat this procedure. As a result of this, we may notice that the k centers change their location step by step until no more changes are done or in other words centers do not move any more.

K-MEANS CLUSTERING

HOW IT WORKS ?

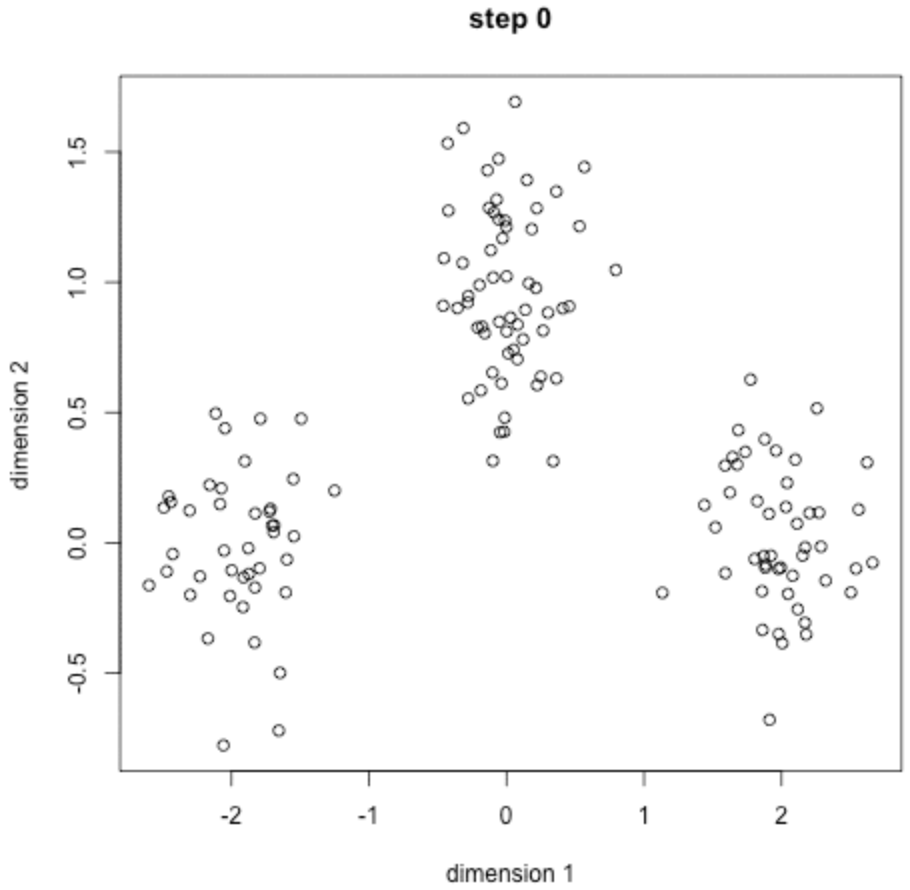
K =2 (2 cluster)





K-MEANS CLUSTERING

HOW IT WORKS ?



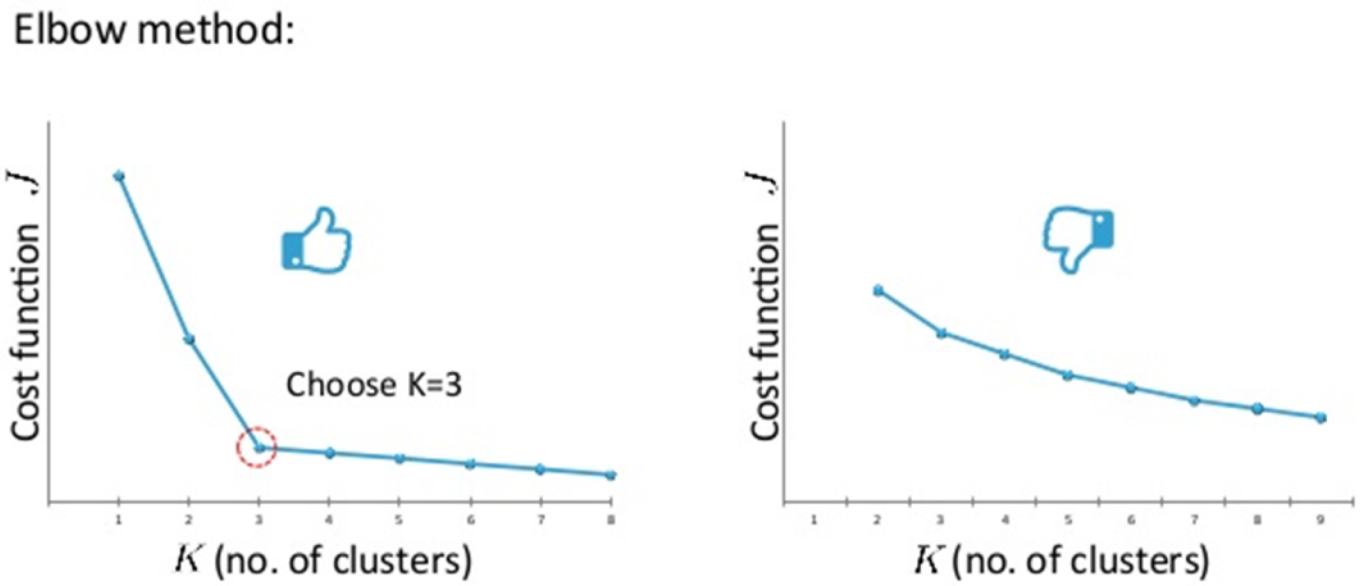
K-MEANS CLUSTERING

HOW MANY CLUSTERS?

Demo

Elbow Criterion

- Draw graph of cost function (sum of square of variance within cluster) against the number of clusters
- The first clusters will not add much information (explain a lot of variance), but at some point the marginal gain will drop, giving an angle in the graph.
- The number of clusters are chosen at this point, hence the “elbow criterion”.



K-MEANS CLUSTERING

PROS AND CONS

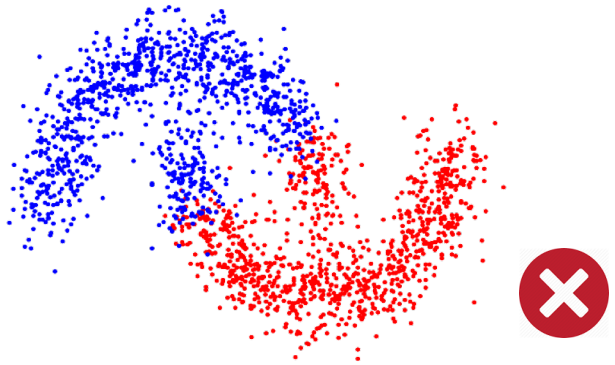
Demo

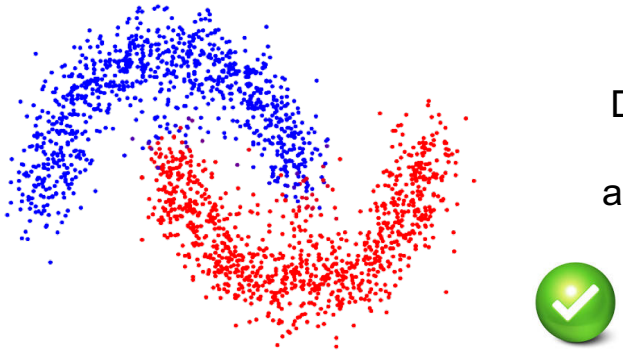
PROS

- Fast, robust and easier to understand.
- Gives best result when data set are distinct or well separated from each other.

CONS

- If there are two highly overlapping data then k-means will not be able to resolve that there are two clusters.
- Algorithm fails for non-linear data set.
- Applicable only when mean is defined i.e. fails for categorical data.
- Unable to handle noisy data and outliers.





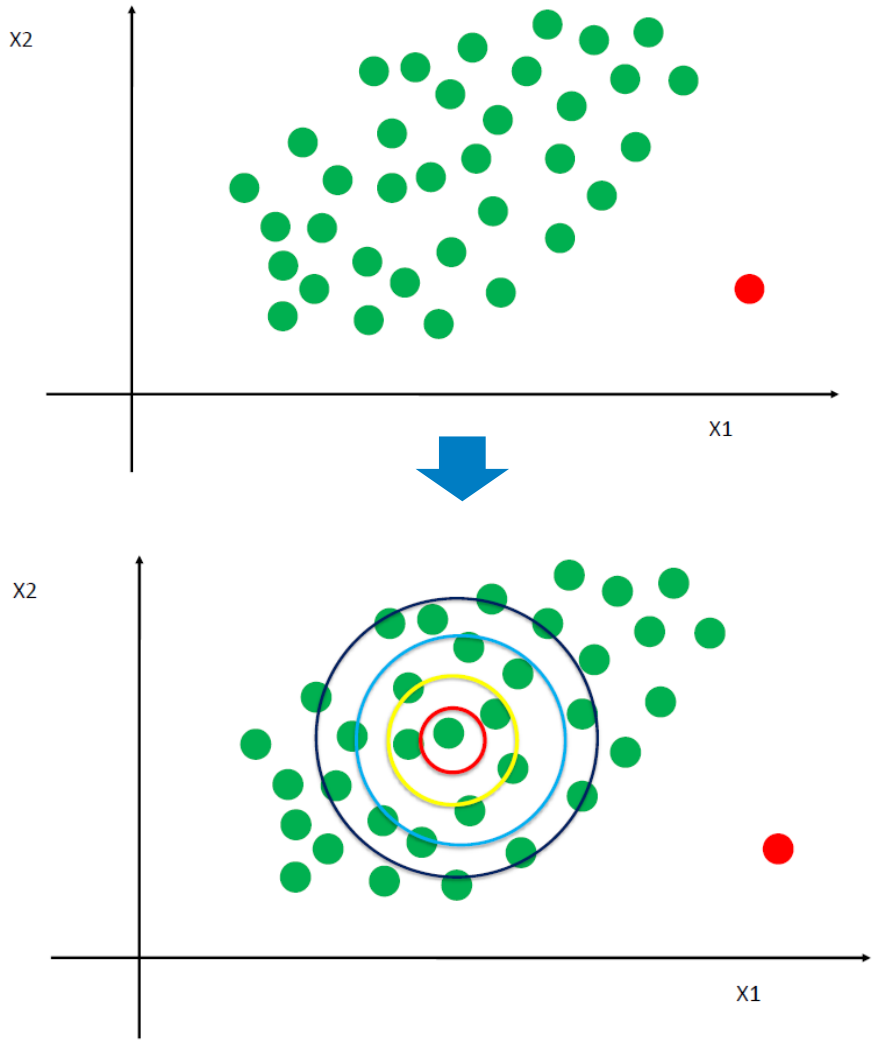
DBSCAN

Density-based spatial clustering of applications with noise

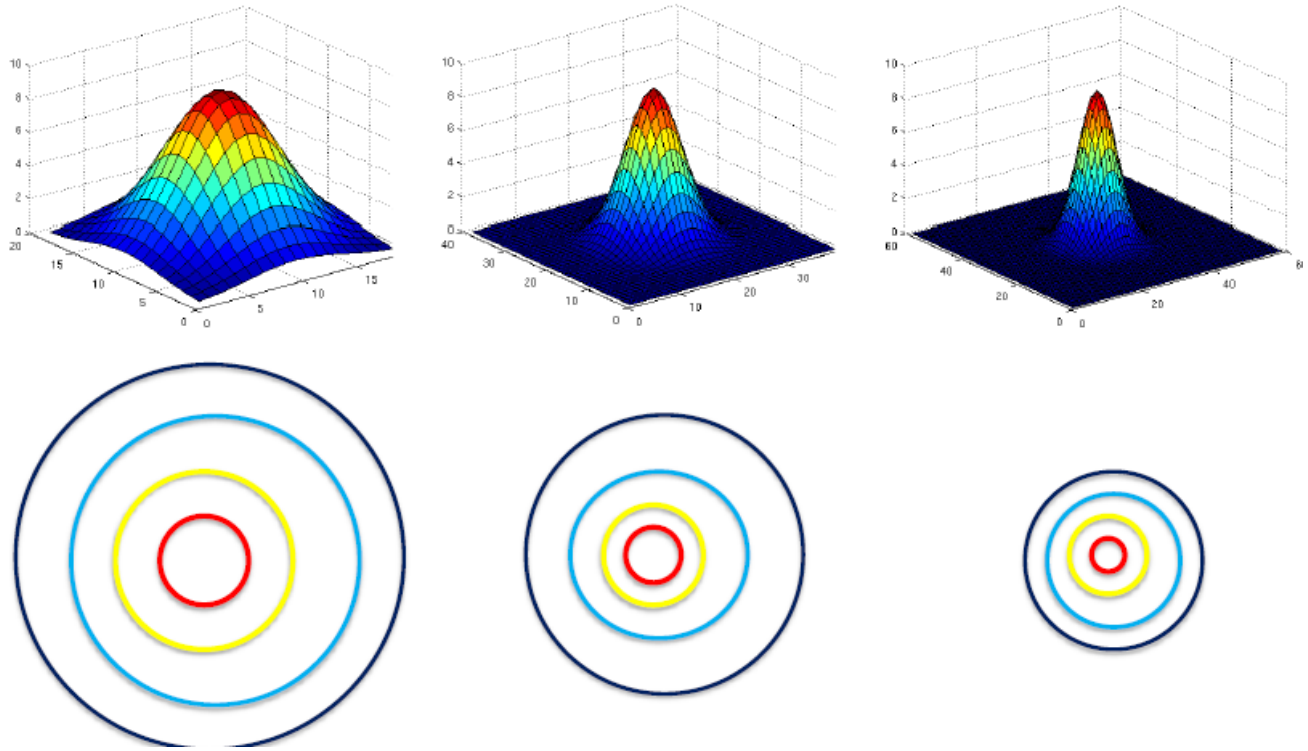
ANAMOLY DETECTION

ANAMOLY DETECTION

HOW IT WORKS

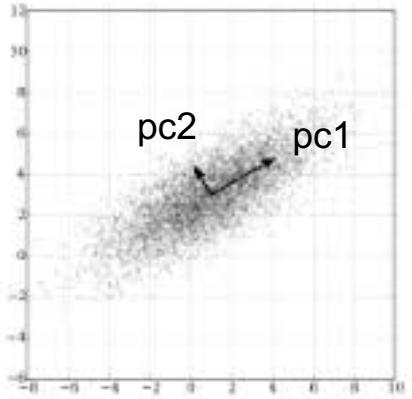


Gaussian Kernel

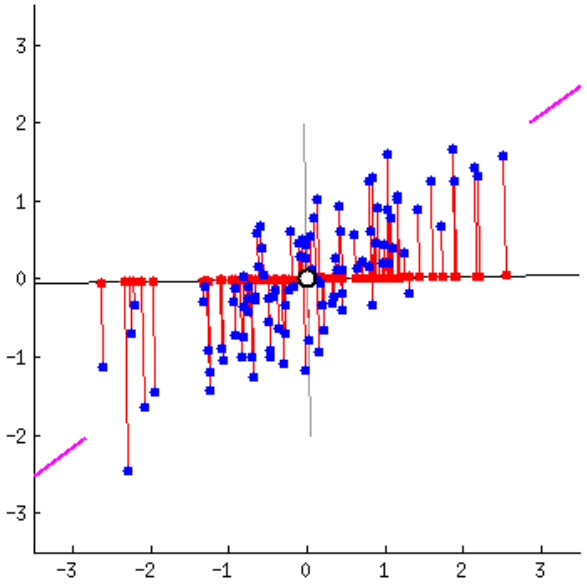


DIMENSIONALITY REDUCTION – PCA / SVD

PRINCIPAL COMPONENT ANALYSIS



- PCA is transformation onto a new feature subspace
- Find directions of maximum variance
- Retain most of the information



Demo

The Math behind

$$\begin{bmatrix} p_{11} & p_{21} \\ \vdots & \vdots \\ p_{1n} & p_{2n} \end{bmatrix} = \begin{bmatrix} x_{11} & x_{21} \\ \vdots & \vdots \\ x_{1n} & x_{2n} \end{bmatrix} \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix}$$

With two dimensions

$$\mathbf{P = X\,W}$$

In general

w_{11} and w_{12} are the *loadings* corresponding to the first principle component.

w_{21} and w_{22} are the *loadings* corresponding to the second principle component.

It turns out that the columns of W
are the eigenvalue vectors of the matrix X^TX

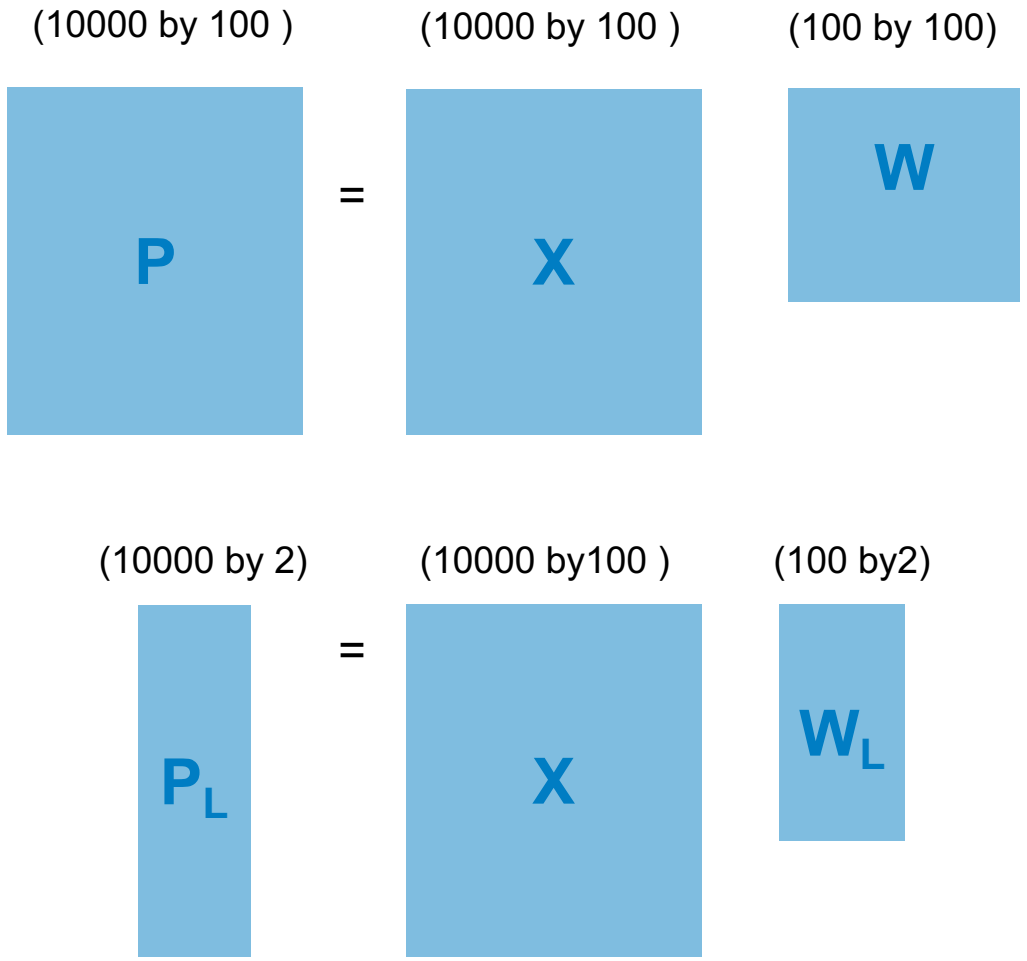
DIMENSION REDUCTION

$$P = X W$$

Now only take the first L columns of W

$$P_L = X W_L$$

For example for visualization only use the first 2 or 3 columns so that P_L only has 2 or 3 columns that can be visualized in scatter or contour plots



Demo

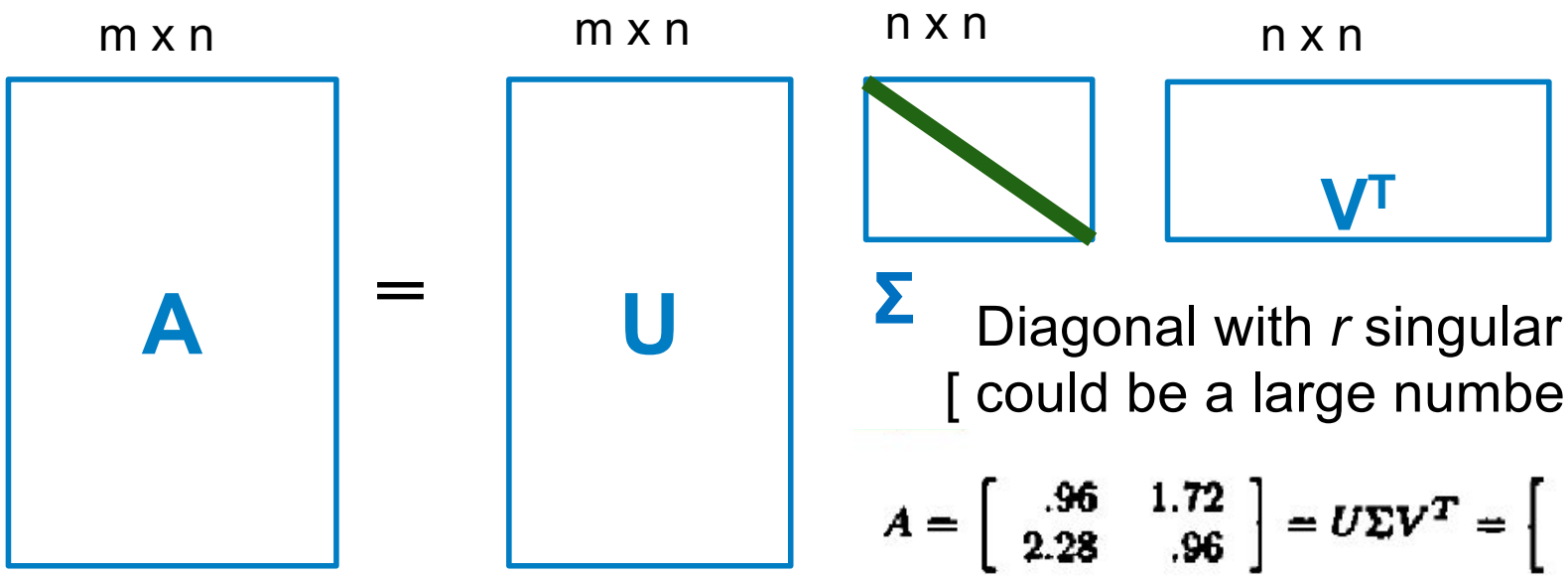
Cocktail party

Applications of PCA

- ✓ Dimension reduction
- ✓ Data compression (communication system)
- ✓ Feature extraction (computer vision)
- ✓ Visualisation (machine learning)
  - ✓ → Outlier / anomalie detectie
- ✓ PCA regression
  - ✓ Use PC instead of the original inputs

SINGULAR VALUE DECOMPOSITION

Matrix SVD decomposition: $A = U\Sigma V^T$



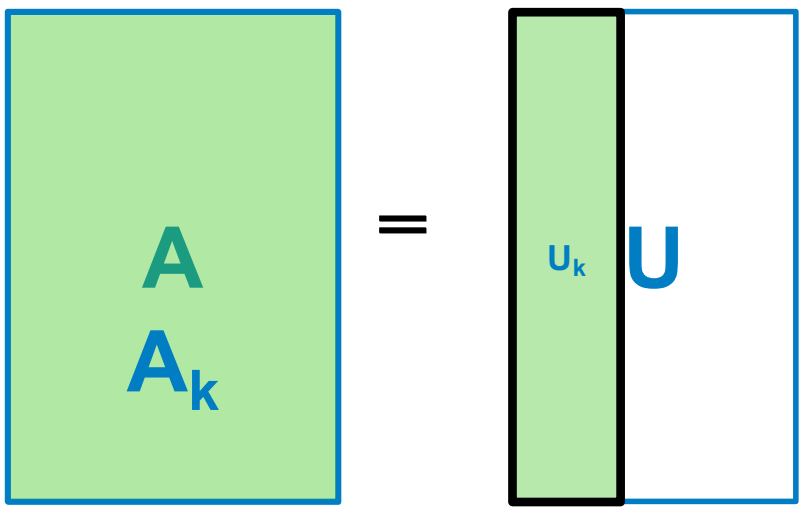
Diagonal with *r* singular values
[could be a large number]

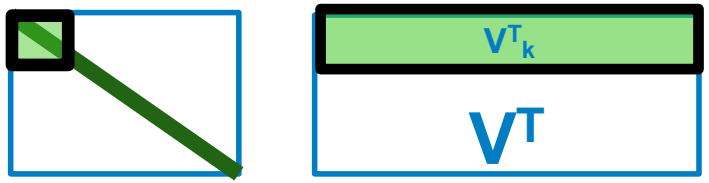
$$A = \begin{bmatrix} .96 & 1.72 \\ 2.28 & .96 \end{bmatrix} = U\Sigma V^T = \begin{bmatrix} .6 & -.8 \\ .8 & .6 \end{bmatrix} \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} .8 & .6 \\ .6 & -.8 \end{bmatrix}^T$$

- Any real m x n matrix A can be decomposed uniquely $A = U\Sigma V^T$
- U is m x n and column orthonormal ($U^T.U=I$) (the columns of U are called **Eigen vectors**)
- ∑ is n x n and diagonal (**Covariance Matrix**)
  - σi are called singular values of A
  - It is assumed that σ1 ≥ σ2 ≥ … ≥ σn ≥ 0
- V is n x n and orthonormal ($V.V^T=V^T.V=I$)

SINGULAR VALUE DECOMPOSITION

Matrix SVD decomposition: $A = U\Sigma V^T$



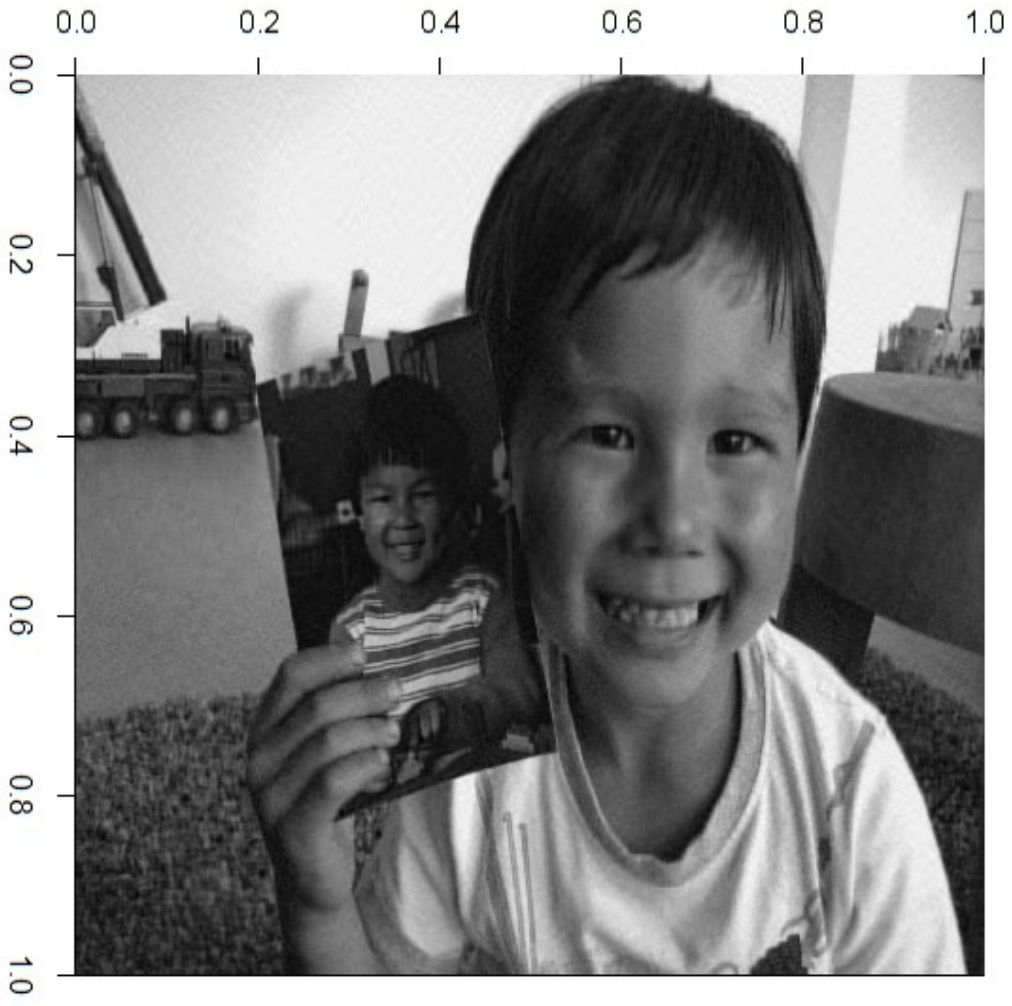


Σ Diagonal with *r* singular values [could be a large number]

Σ_k Take only k << r singular values

A datapoint *d* can now be represented by *k* dimensional point $\hat{\mathbf{d}} = U_k^T \mathbf{d}.$

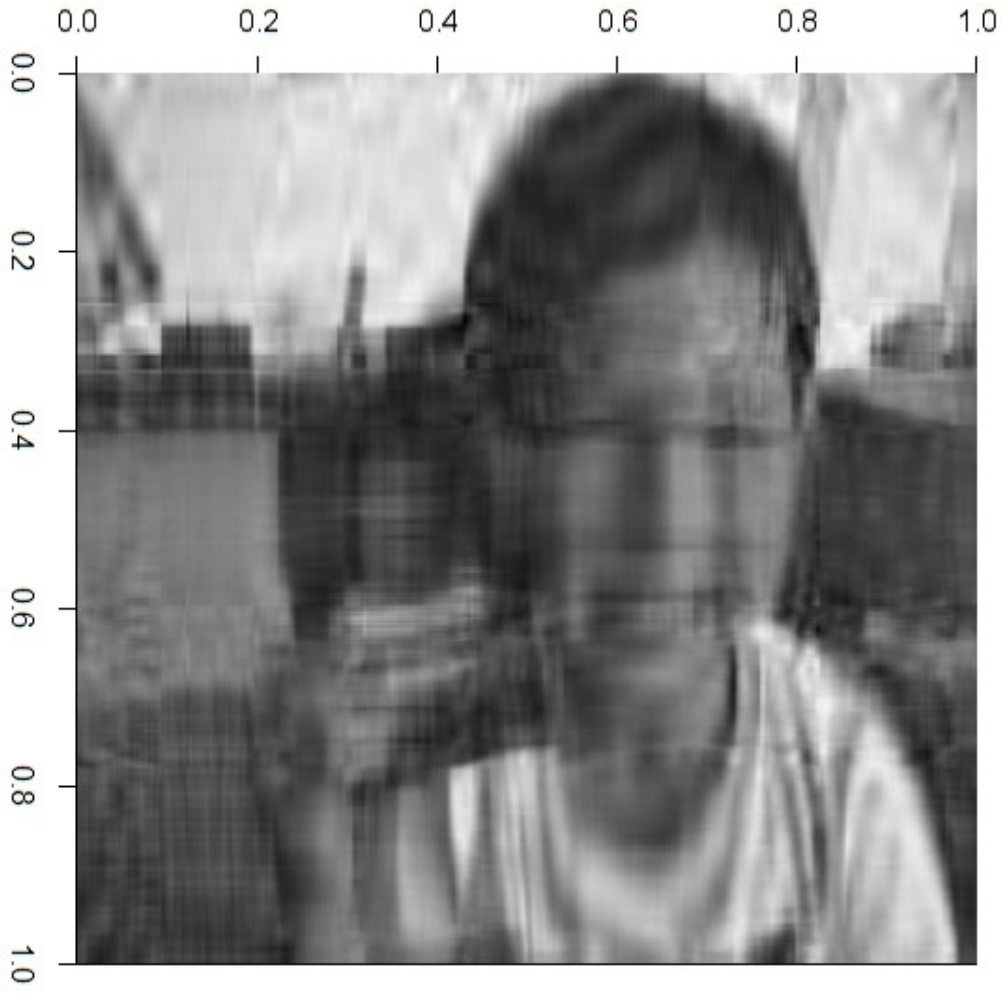
SVD EXAMPLE | AS AN EXPERIMENT



Original

2448 X 3264 ~ 8 mln numbers

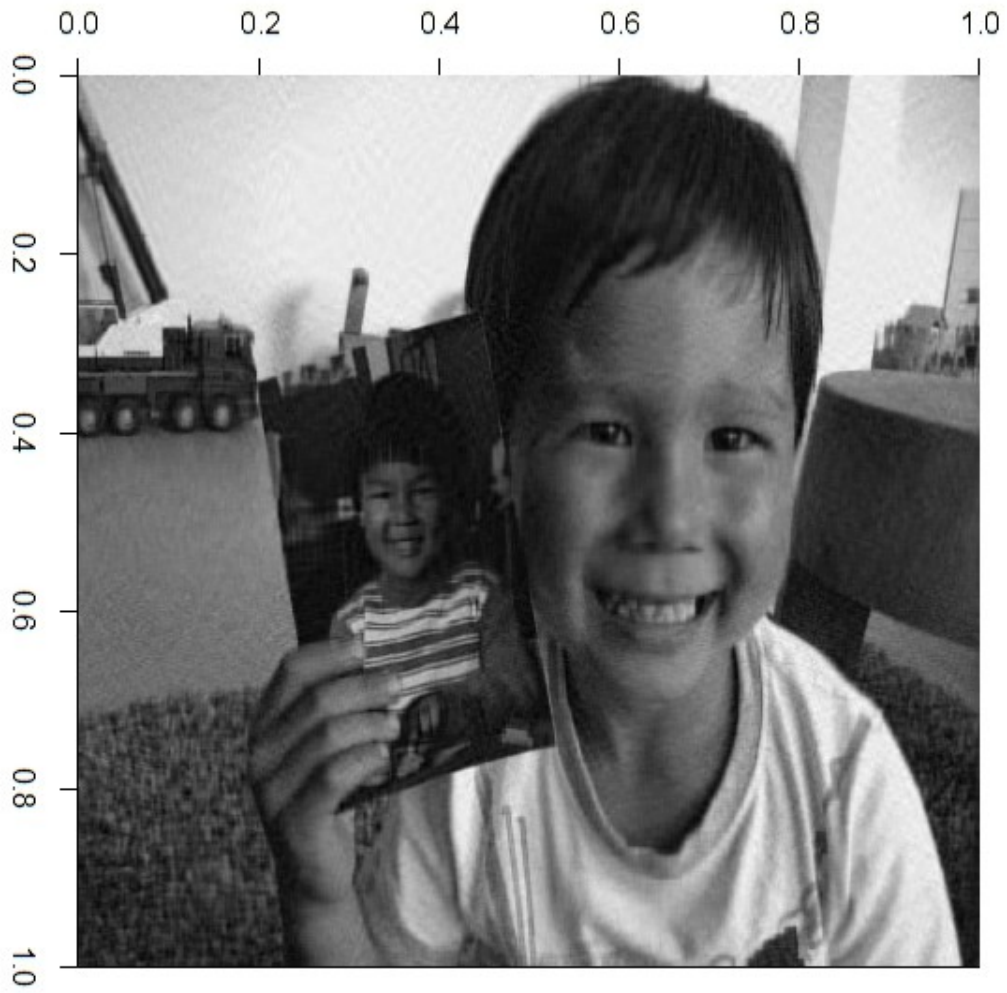
SVD EXAMPLE | AS AN EXPERIMENT



SVD: 15 largest SV's

1% of the data

SVD EXAMPLE | AS AN EXPERIMENT



SVD: 75 largest V's
5% of the data

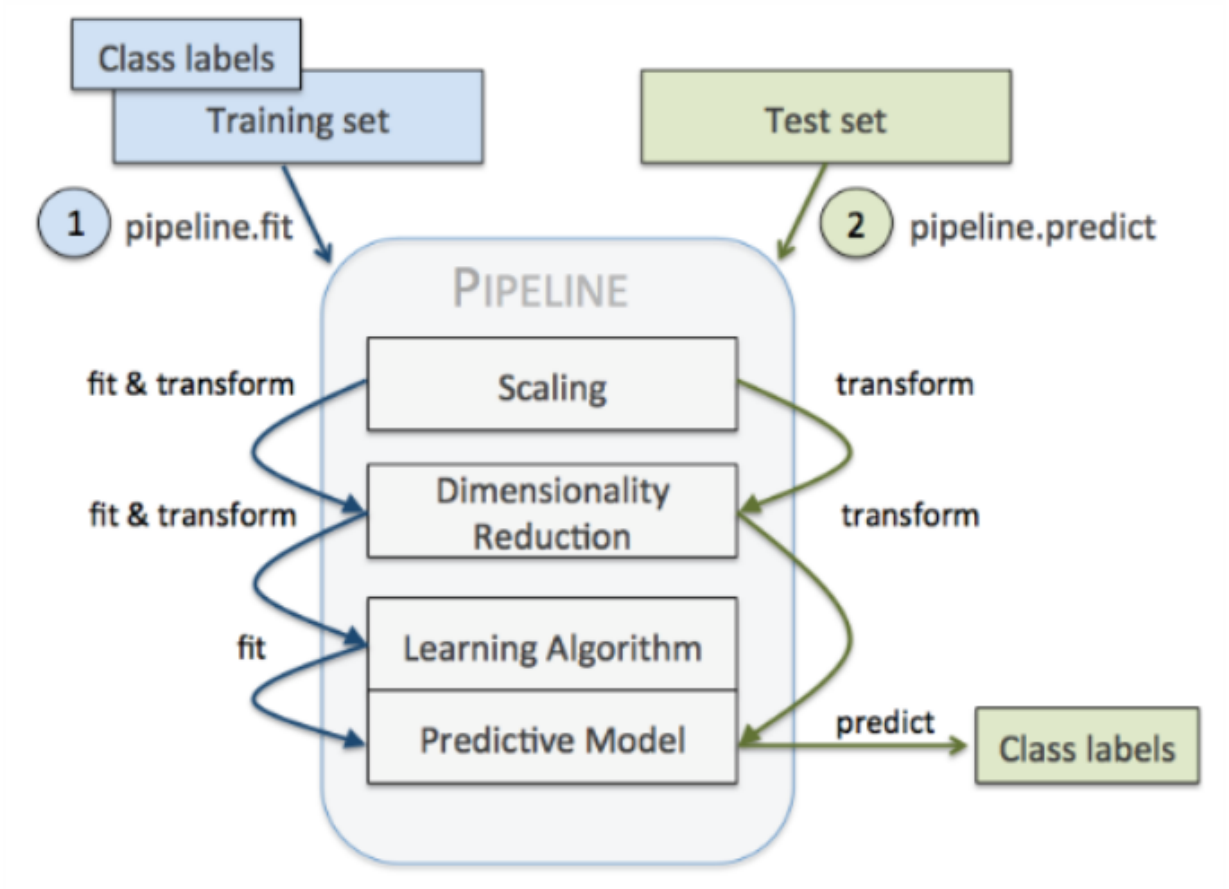
ML BEST PRACTICES

BEST PRACTICES | STANDARDISATION AND OUTLIERS

- Data set has features whose values are significantly different in magnitude and range.
- This disparity can degrade the performance
- To mitigate this transform your interval feature values to be on a similar scale, commonly standardizing to have mean 0 and variance 1

- You will often find outliers in your data—observations that are very distinct from the others in one or more of the feature values.
- Although outliers can certainly be very informative and can identify anomalies that deserve special attention, they can be quite detrimental to training an effective generalizable model.
- In general, outliers drag clusters artificially toward the outside of your feature space
- Determine whether an outlying value is simply an invalid or erroneous entry that can be disregarded

BEST PRACTICES | PIPELINES




```
import numpy as np
from sklearn.svmimport SVR
from sklearn.pipelineimport Pipeline
from sklearn.preprocessingimport StandardScaler

x,y = np.load('data.npz')
x_test = np.linspace(0, 200)

model = Pipeline([
    ('standardize', StandardScaler()),
    ('svr', SVR(kernel='rbf', verbose=0, C=5e6,
                epsilon=20)) ])
model.fit(x[::, np.newaxis], y)
y_test = model.predict(x_test[::, np.newaxis])
```

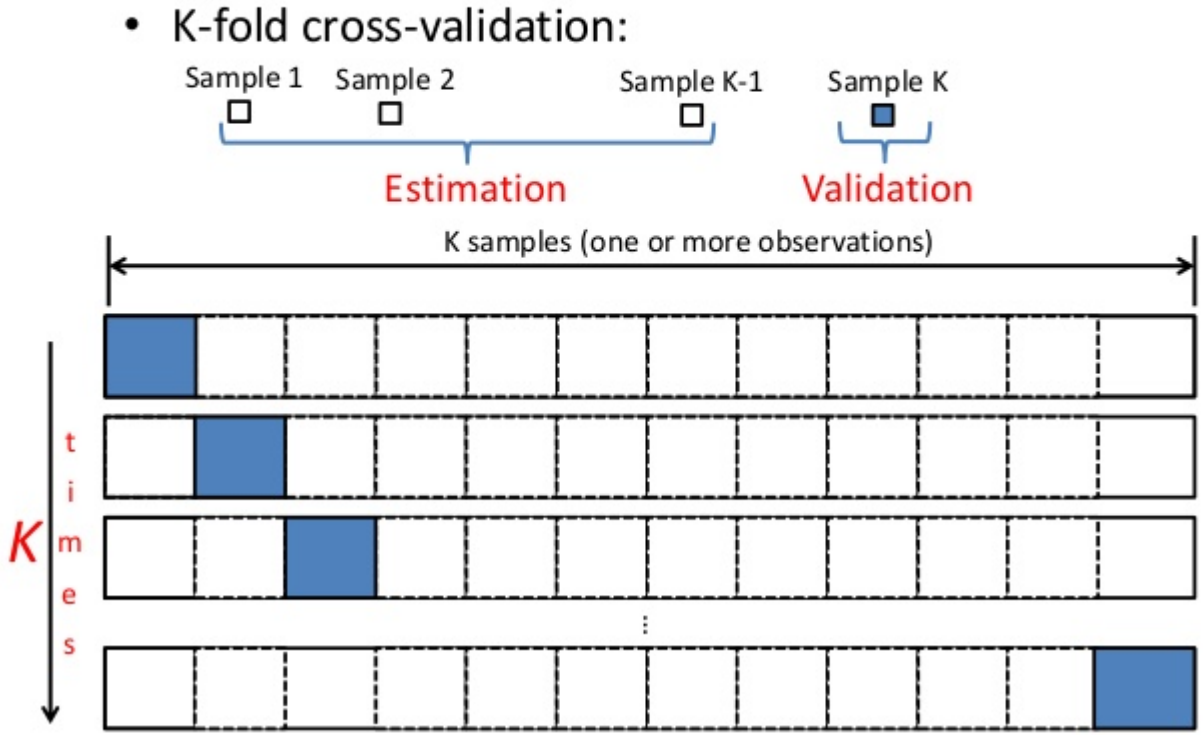
regularization term

Pipelines - Sequence of data transforms to be chained together culminating in a modeling process that can be evaluated.

BEST PRACTICES K- FOLD CROSS VALIDATION

Demo

- Cross-validation avoids overlapping test sets
    - First step: split data into k subsets of equal size
    - Second step: use each subset in turn for testing, the remainder for training
- Called k-fold cross-validation (K = 10)
- The process is repeated K times (folds)
- Often the subsets are stratified (ranked) before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

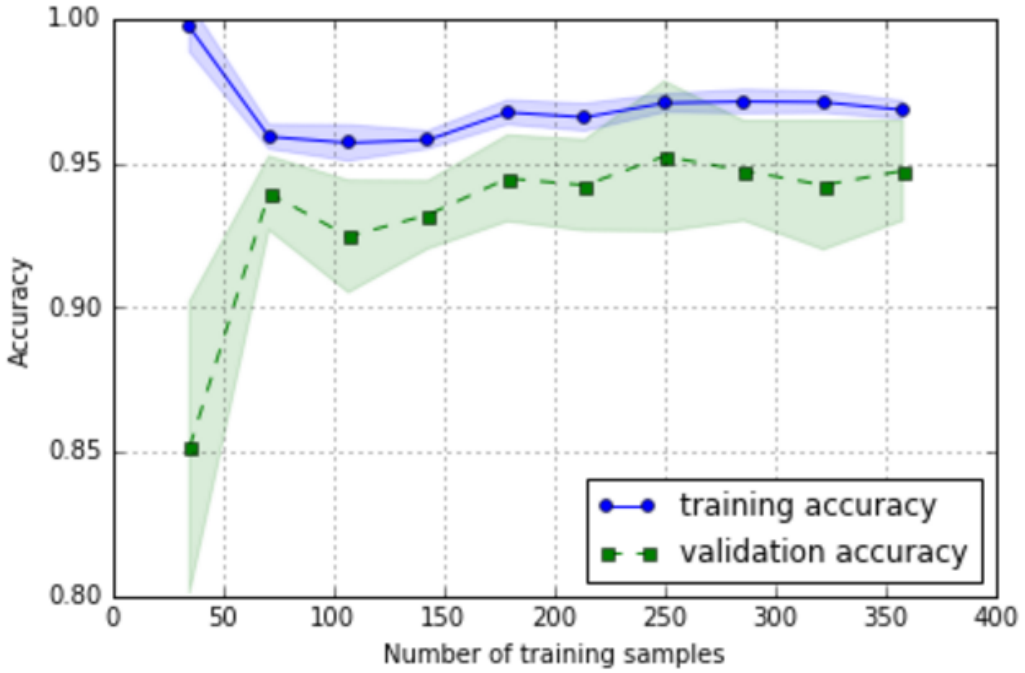


BEST PRACTICES | PERFORMANCE EVALUATION MATRIX – GRID SEARCH

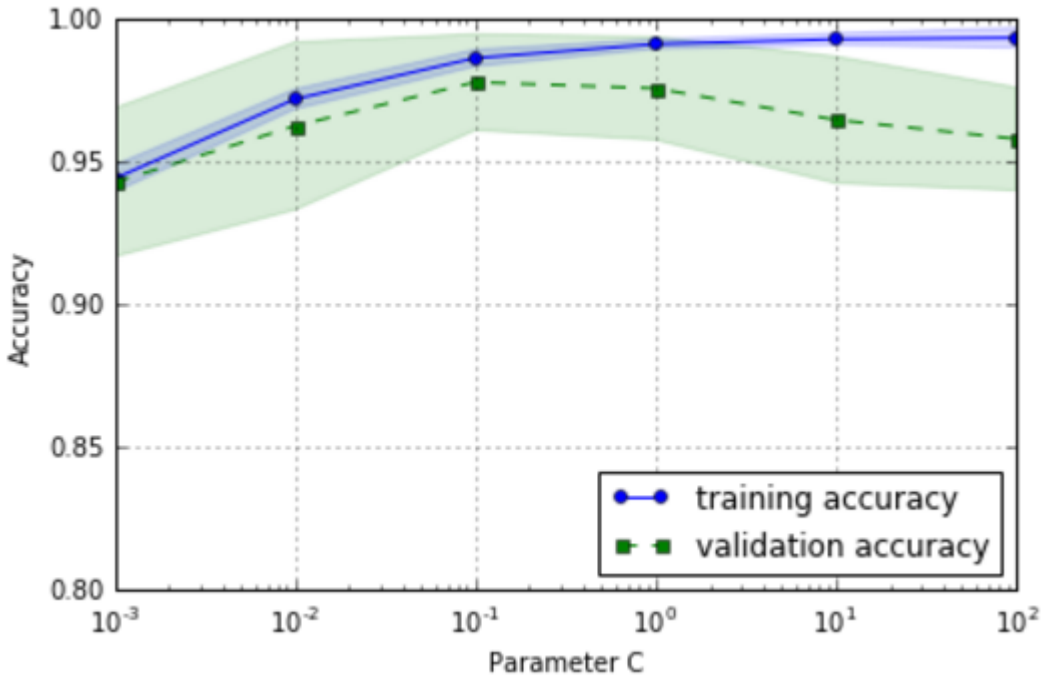
The general belief is that more samples are required for better accuracy.

Demo

Learning Curve

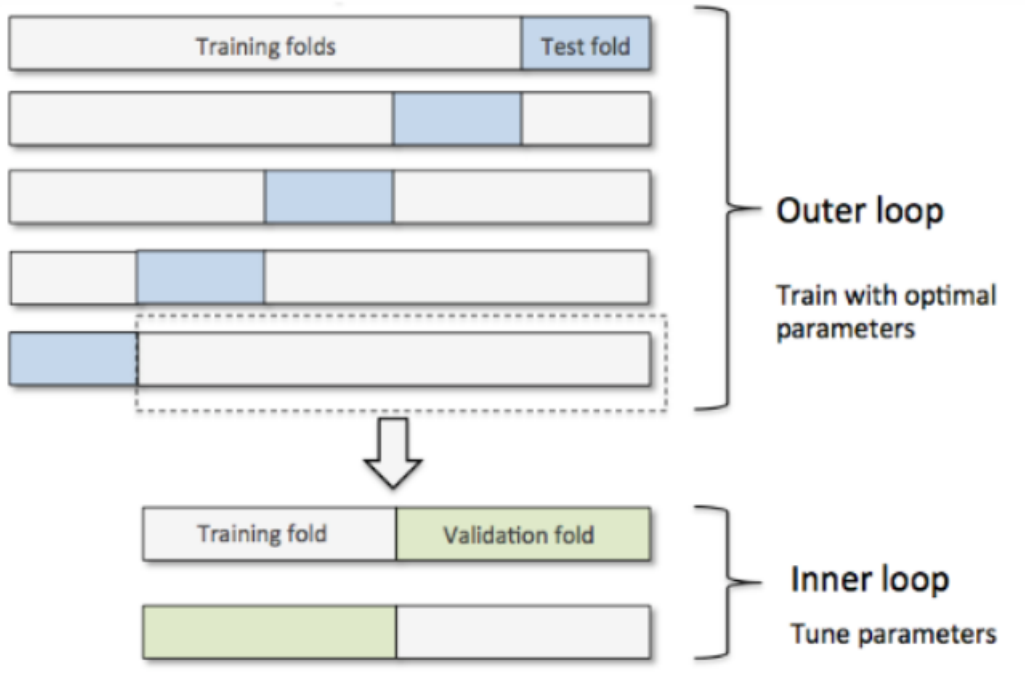


Validation Curve

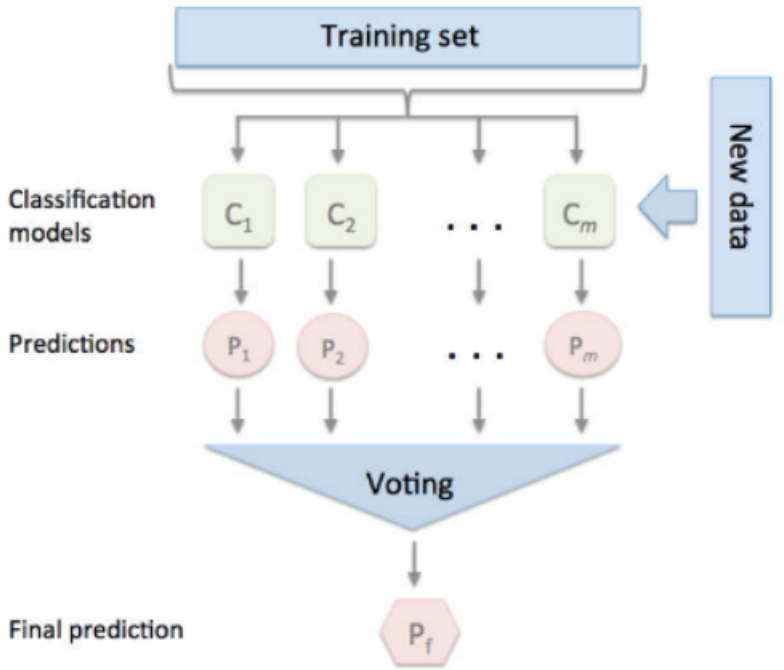


Demo

Grid Search + Cross Validation



Majority voting Classifier



BEST PRACTICES | MACHINE LEARNING PLATFORMS

THANK YOU

PURUSHOTTAM DARSHANKAR