

Machine Learning (Supervised Learning)



Hello!

I am Eslam Ahmed

I am a software engineer.

You can find me at jeksogsa@gmail.com



Hello!

I am Eman Ehab

I am a ML research engineer.

You can find me at
emanehab.ieee@gmail.com



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

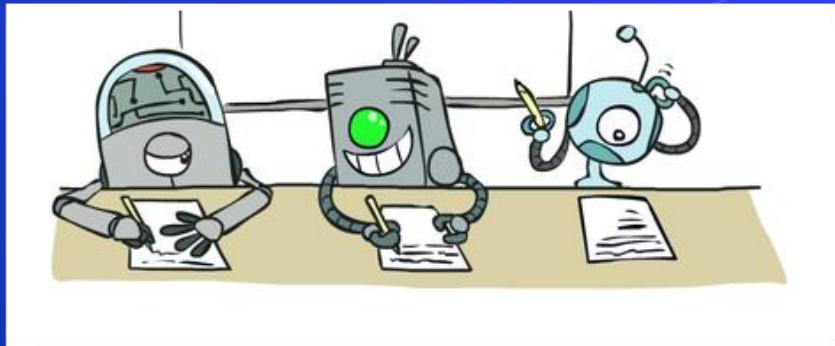
Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

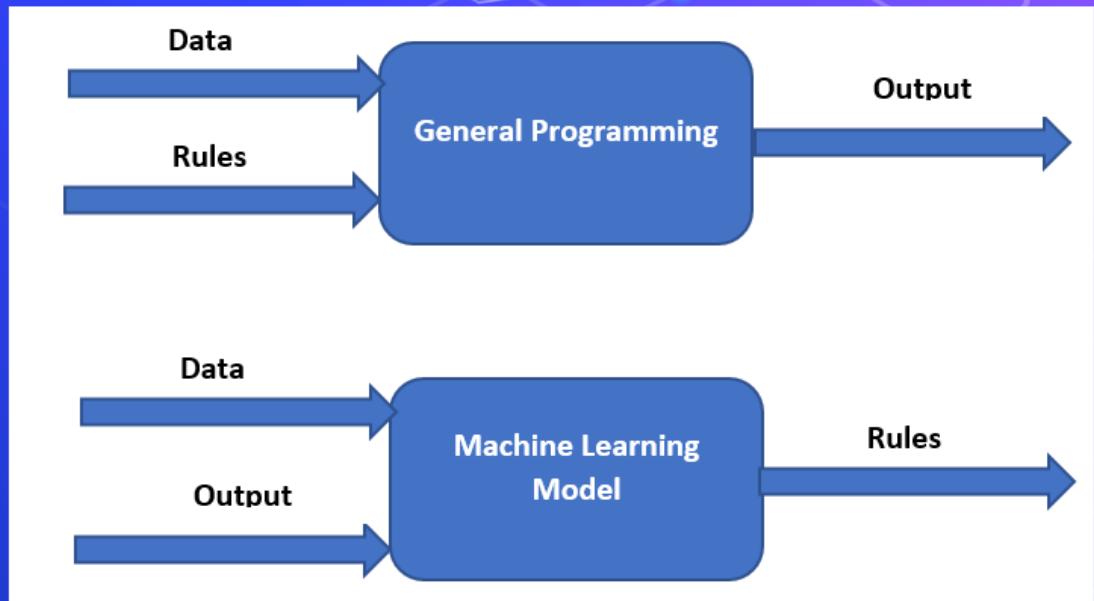
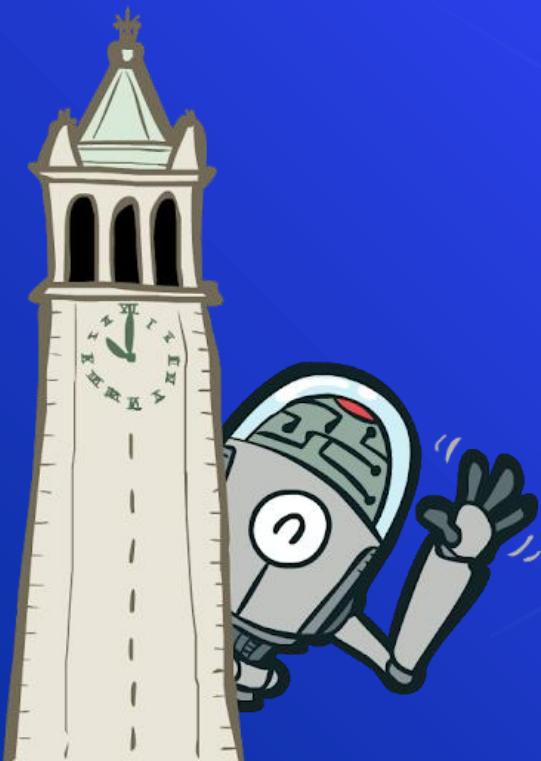
Recommendation Systems

Machine Learning

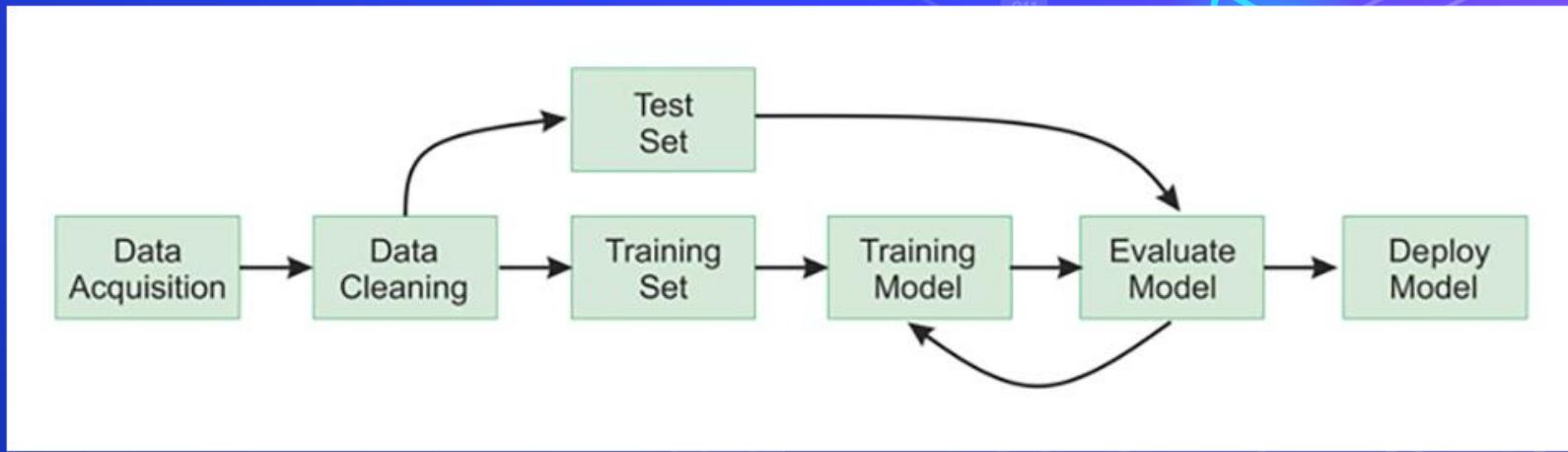
Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks.



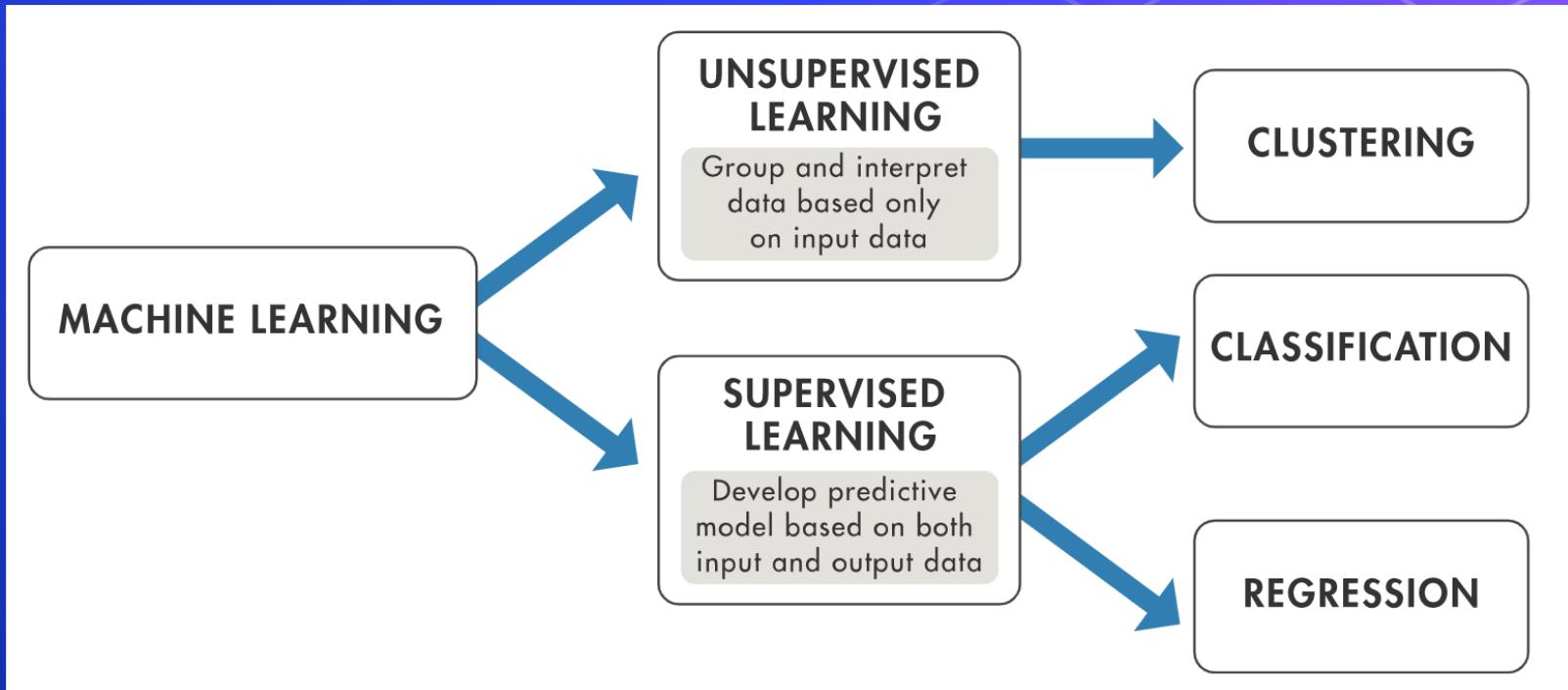
Machine Learning



Machine Learning



Machine Learning



Machine Learning

Classical Machine Learning

Task Driven

Supervised Learning

(Pre Categorized Data)



Classification

(Divide the socks by Color)

Eg. Identity Fraud Detection



Regression

(Divide the Ties by Length)

Eg. Market Forecasting

Data Driven

Unsupervised Learning

(Unlabelled Data)



Clustering

(Divide by Similarity)

Eg. Targeted Marketing



Association

(Identify Sequences)

Eg. Customer Recommendation



Dimensionality Reduction

(Wider Dependencies)

Eg. Big Data Visualization

Obj: Predictions & Predictive Models

Pattern/ Structure Recognition



Machine Learning

- Supervised learning : Task Driven (Classification, Regression)
- Unsupervised learning : Data Driven (Clustering)
- Reinforcement learning :
 - Close to human learning.
 - Algorithm learns a policy of how to act in a given environment.
 - Every action has some impact in the environment, and the environment provides rewards that guides the learning algorithm.

Machine Learning

Supervised Learning

Regression

- Simple Linear Regression
- Multiple Linear Regression
- Polynomial Regression
- Evaluating Model Performance

Classification

- Logistic Regression
- K-Nearest Neighbors (KNN)
- Naive Bayes
- SVM
- Decision Trees
- Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
- Evaluating Model Performance

Unsupervised Learning

Clustering

- KMeans
- Hierarchical Clustering
- Density Based Clustering - DBSCAN

Association rule mining

- Apriori

Dimension Reduction

- PCA
- LDA

Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

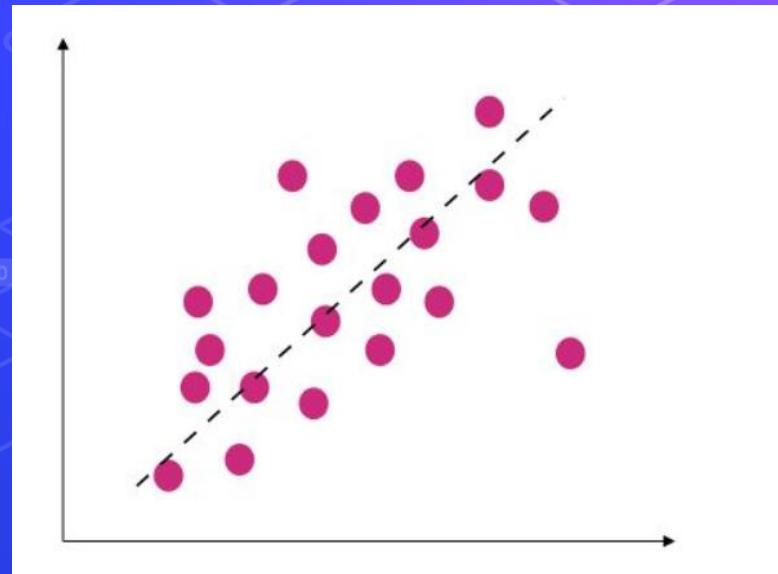
Recommendation Systems

Regression

Regression quantifies the relationship between one or more predictor variable(s) and one outcome variable.

For example,
it can be used to quantify the relative impacts of age, gender,
and diet (the predictors) on height (the output or dependent).

- House prices based on size, locations,...
- Salary prediction based on experience
- Stock Market prediction
- Weather prediction
- ...



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

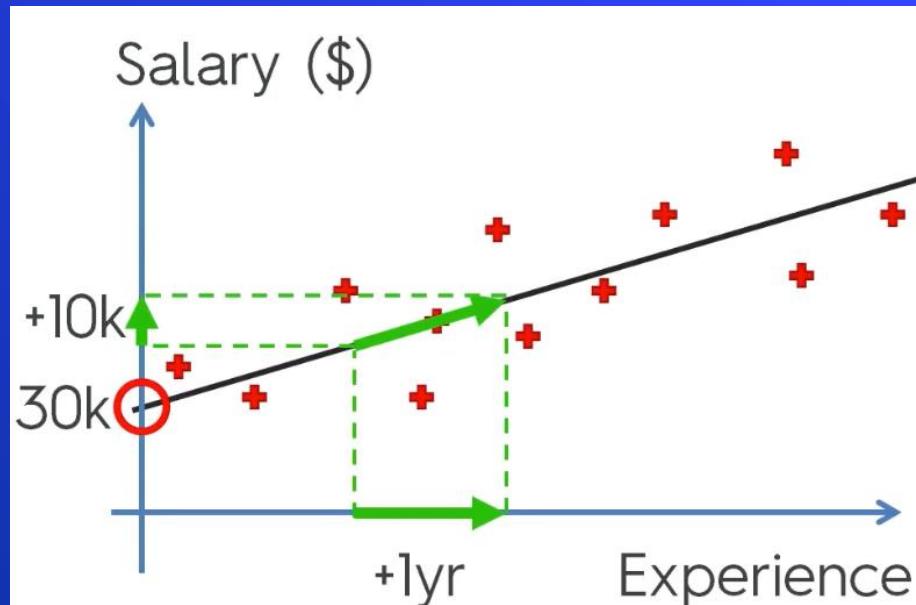
- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Simple Linear Regression (Equation of a straight line)



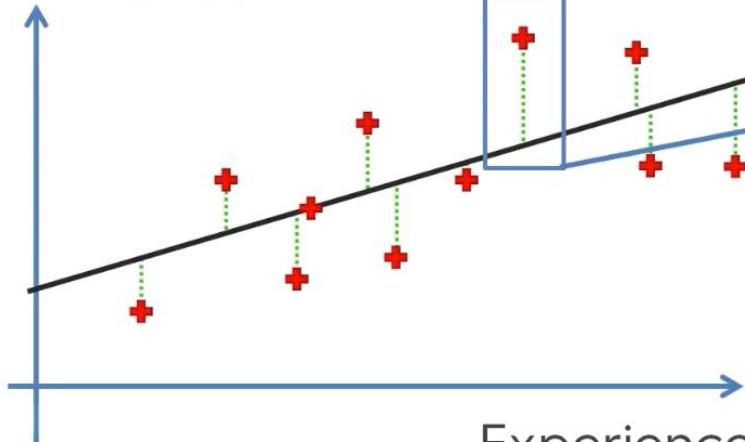
$$y = b_0 + b_1 * x$$

$$\text{Salary} = b_0 + b_1 * \text{Experience}$$

Simple Linear Regression (Calculate Cost Function)

Simple Linear Regression:

Salary (\$)



y_i

\hat{y}_i

Hypothesis:
$$h_{\theta}(x) = \theta_0 + \theta_1 x$$

Parameters: θ_0, θ_1

Cost Function:
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Goal:
$$\underset{\theta_0, \theta_1}{\text{minimize}} J(\theta_0, \theta_1)$$

Simple Linear Regression (Example)

Theta0 = 5 , theta 1 = 2

Equation $h(x) = 5 + 2x$

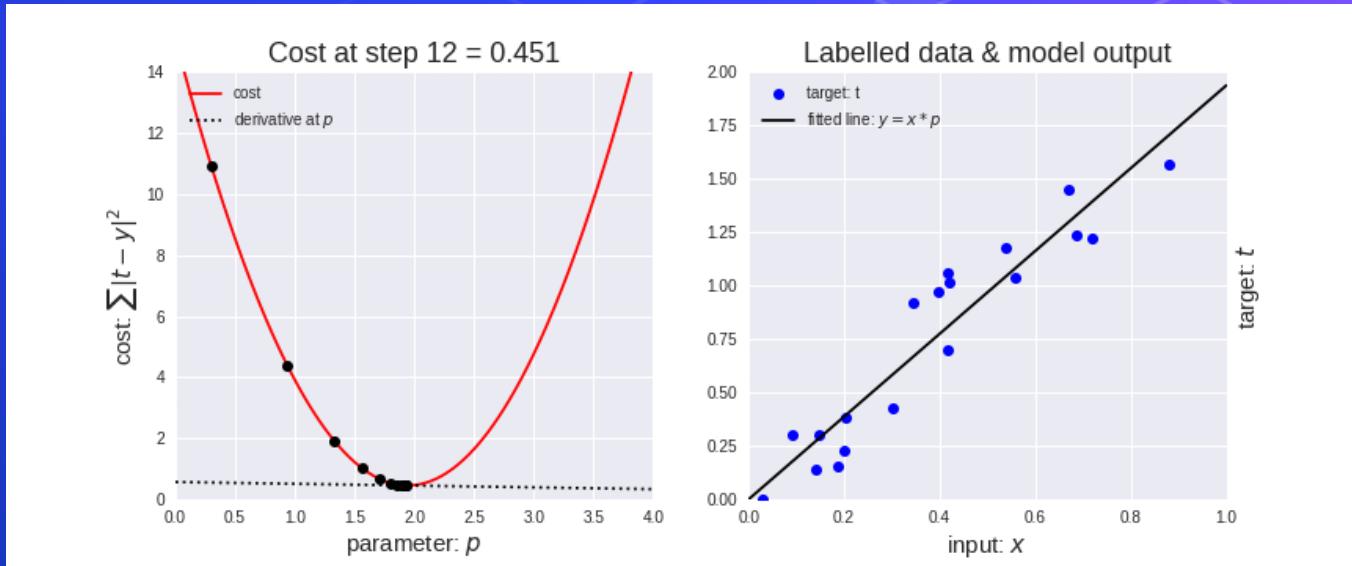
X	Y	$h(x)$	$h(x) - y$	$(h(x) - y)^2$
1	7	7	0	0
2	8	9	1	1
2	7	9	2	4
3	9	11	2	4
4	11	13	2	4
5	10	15	5	25
5	12	15	3	9

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$J = 1 / 14 (0+1+4+4+4+25+9)$$

$$J = 47/14 = 3.3$$

Simple Linear Regression (Minimize cost using Gradient Descent)



repeat until convergence:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x_i) - y_i)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m ((h_\theta(x_i) - y_i)x_i)$$

Simple Linear Regression (Minimize cost using Gradient Descent)

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m} \sum_{i=1}^m [h_\Theta(x_i) - y_i]^2$$

↑
Predicted Value ↑
True Value

Gradient Descent

$$\Theta_j = \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)$$

↑
Learning Rate

NOW,

$$\begin{aligned}\frac{\partial}{\partial \Theta} J_\Theta &= \frac{\partial}{\partial \Theta} \frac{1}{2m} \sum_{i=1}^m [h_\Theta(x_i) - y_i]^2 \\&= \frac{1}{m} \sum_{i=1}^m (h_\Theta(x_i) - y) \frac{\partial}{\partial \Theta_j} (\Theta x_i - y) \\&= \frac{1}{m} (h_\Theta(x_i) - y) x_i\end{aligned}$$

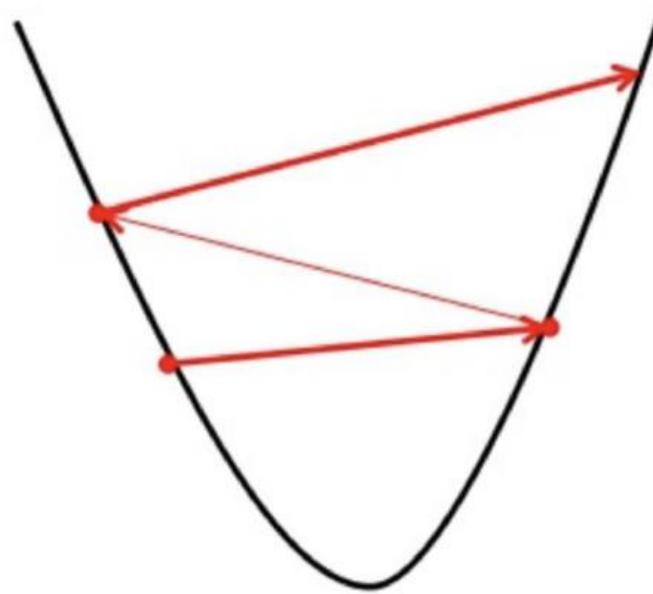
Therefore,

$$\Theta_j := \Theta_j - \frac{\alpha}{m} \sum_{i=1}^m [(h_\Theta(x_i) - y) x_i]$$

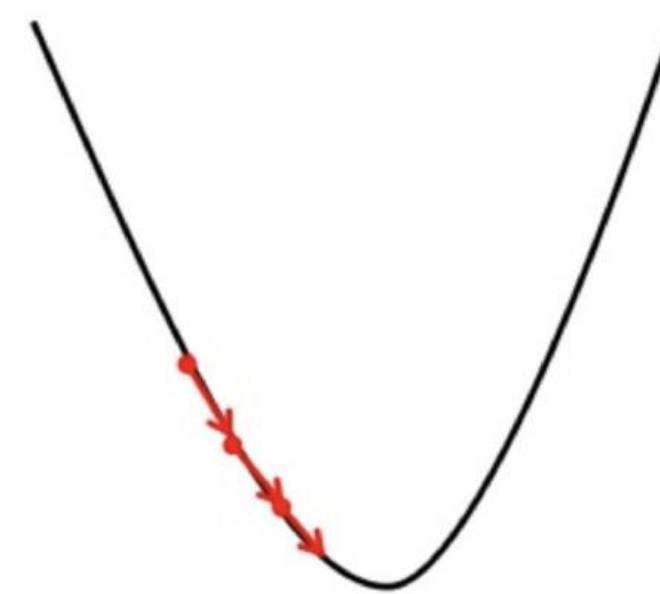


Simple Linear Regression (Alpha constant)

Big learning rate



Small learning rate



Simple Linear Regression (Code)

```
 1 from sklearn.linear_model import LinearRegression
 2
 3 # make model object
 4 model = LinearRegression()
 5
 6 # train
 7 model.fit(x_train, y_train)
 8
 9 # test
10 model.predict(x_test)
11
12 # calculate R2 score on training data
13 model.score(x_train, y_train)
14
15 # calculate R2 score on testing data
16 model.score(x_test, y_test)
17
18 # get model parameters
19 model.coef_
20 model.intercept_
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Multiple Linear Regression (Equation)

Simple
Linear
Regression

$$y = b_0 + b_1 * x_1$$

Multiple
Linear
Regression

Dependent variable (DV) Independent variables (IVs)

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \dots + b_n * x_n$$

Constant Coefficients

```
graph TD; DV[Dependent variable DV] --> y; IVs[Independent variables IVs] --> x1; IVs --> x2; IVs --> xn; Constant[Constant] --> b0; Coefficients[Coefficients] --> b1x1; Coefficients --> b2x2; Coefficients --> bnxn;
```

Multiple Linear Regression (Update theta values)

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_0^{(i)}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_1^{(i)}$$

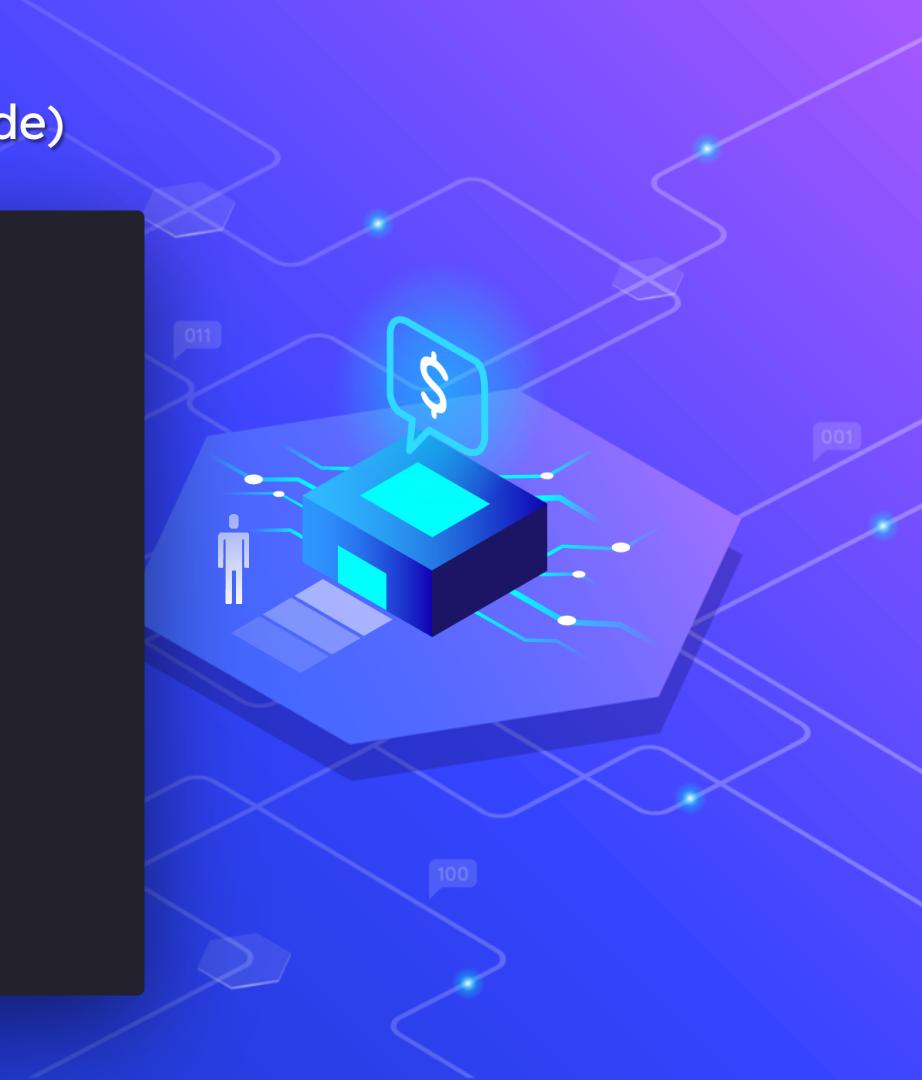
$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

...

}

Multiple Linear Regression (Code)

```
 1 from sklearn.linear_model import LinearRegression
 2
 3 # make model object
 4 model = LinearRegression()
 5
 6 # train
 7 model.fit(x_train, y_train)
 8
 9 # test
10 model.predict(x_test)
11
12 # calculate R2 score on training data
13 model.score(x_train, y_train)
14
15 # calculate R2 score on testing data
16 model.score(x_test, y_test)
17
18 # get model parameters
19 model.coef_
20 model.intercept_
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - **Polynomial Regression**
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

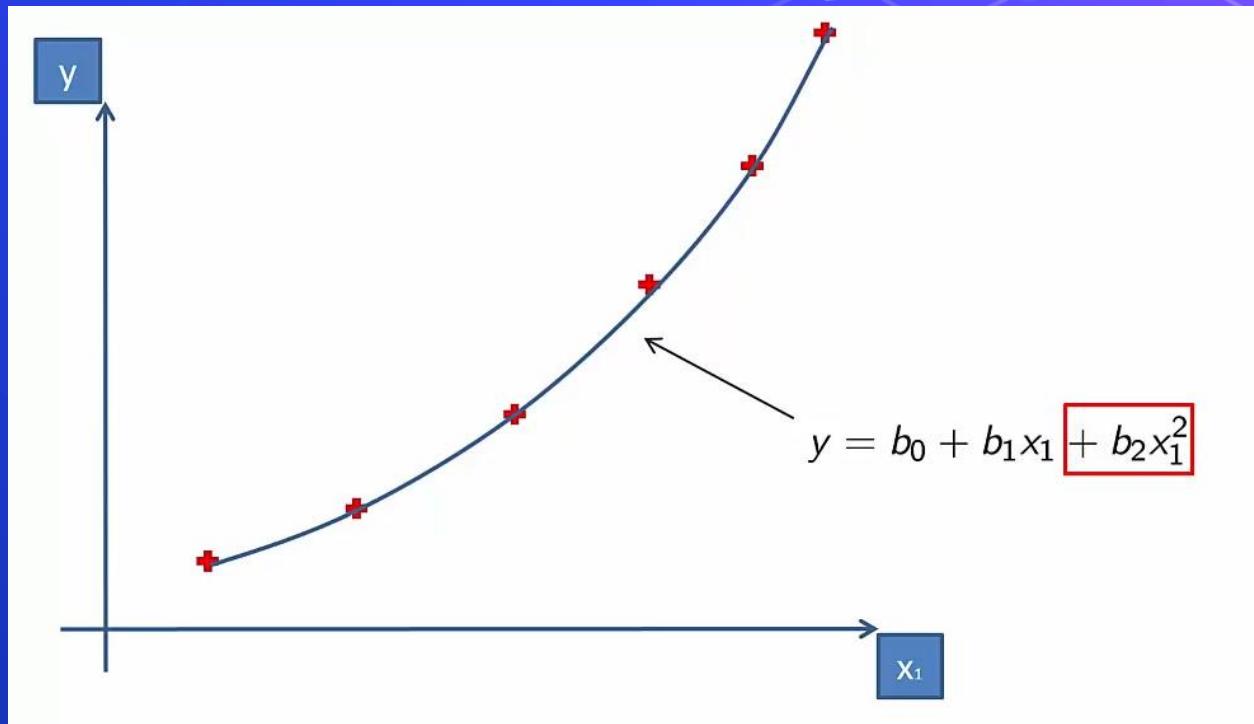
- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Polynomial Regression (Poly equation)



Polynomial Regression (Poly equation)

Polynomial
Linear
Regression

$$y = b_0 + b_1x_1 + b_2x_1^2 + \dots + b_nx_1^n$$

Polynomial Regression (Code)

```
 1 from sklearn.linear_model import LinearRegression
 2 from sklearn.preprocessing import PolynomialFeatures
 3
 4 # create poly features for a feature
 5 poly = PolynomialFeatures(degree=2)
 6 x_train_poly = poly.fit_transform(x_train)
 7 x_test_poly = poly.fit_transform(x_test)
 8
 9 # make model object
10 model = LinearRegression()
11
12 # train
13 model.fit(x_train_poly, y_train)
14
15 # test
16 model.predict(x_test_poly)
17
18 # calculate R2 score on training data
19 model.score(x_train_poly, y_train)
20
21 # calculate R2 score on testing data
22 model.score(x_train_poly, y_train)
23
24 # get model parameters
25 model.coef_
26 model.intercept_
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - **Evaluating Model Performance**
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Evaluating Model Performance (R²)

Hrs Studied (X)	Marks (Y)
0	40
2	52
3	53
4	55
4	56
5	72
6	71
6	88
7	56
7	74
8	89
9	67
9	89
5.38	66.31
Mean	

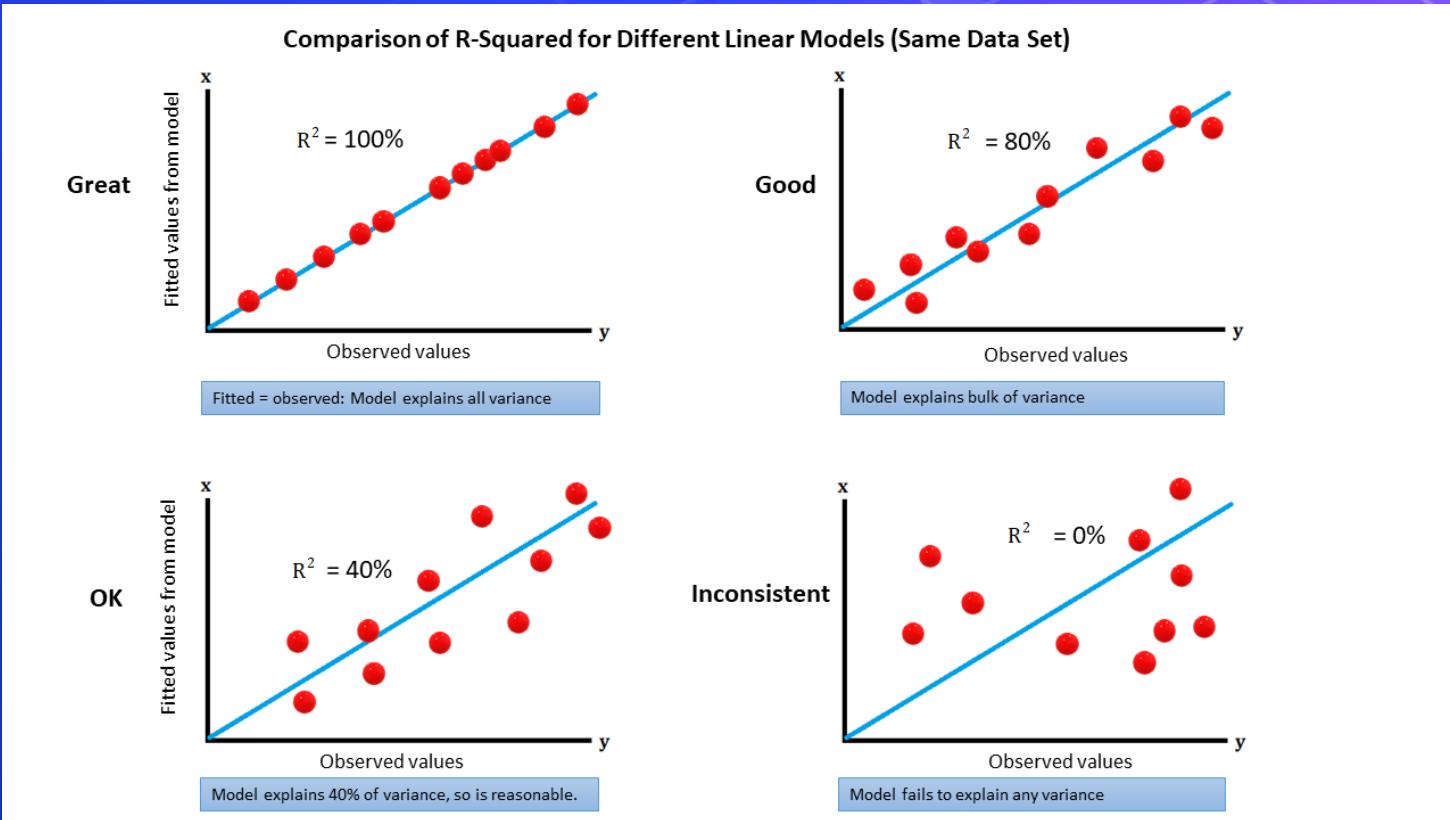
$$\begin{aligned} R^2 &= \frac{\text{SSR}}{\text{SST}} \\ &= \frac{1844.12}{3028.77} \\ &= 0.60886 \end{aligned}$$

Higher the value → Variation in Y is explained by variation in X.

$$\hat{y} \rightarrow y \quad \text{SSR} \rightarrow \text{SST} \quad R^2 \approx 1$$

$(Y - \bar{Y})^2$	$(\hat{Y} - \bar{Y})^2$
692.22	600.74
204.78	237.47
177.16	117.94
127.92	39.82
106.30	39.82
32.38	3.10
22.00	7.78
470.46	7.78
106.30	53.88
59.14	53.88
514.84	141.37
0.48	270.27
514.84	270.27
3028.77	1844.12
SST	SSR

Evaluating Model Performance (R^2)



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

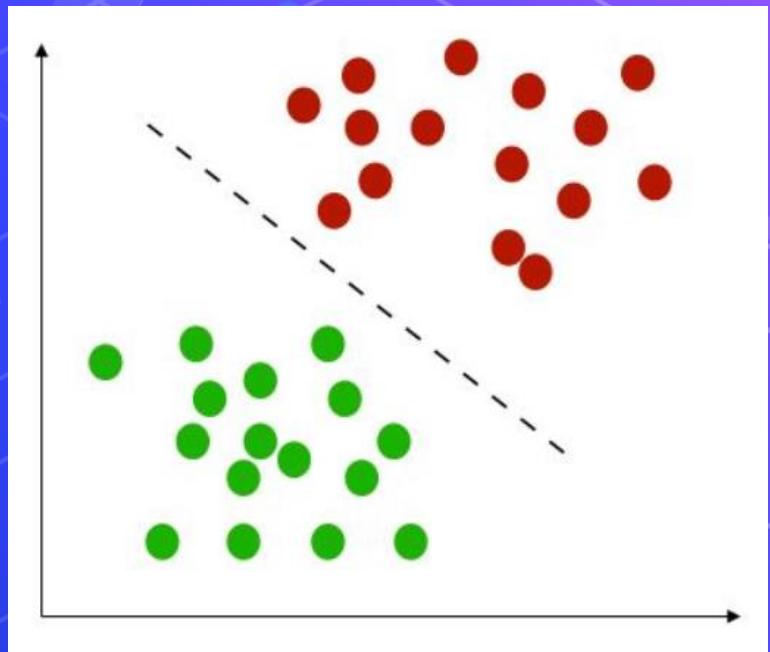
Recommendation Systems

Classification

Classification specifies the class to which data elements belong to and is best used when the output has finite and discrete values. It predicts a class for an input variable as well.

For example,
it can be used to classify the animal in an image, it's either a cat or a dog.

- Email spam detection
- Face classification
- Patient has cancer or not
- Fraud detection
- ...



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - **Logistic Regression**
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Logistic Regression

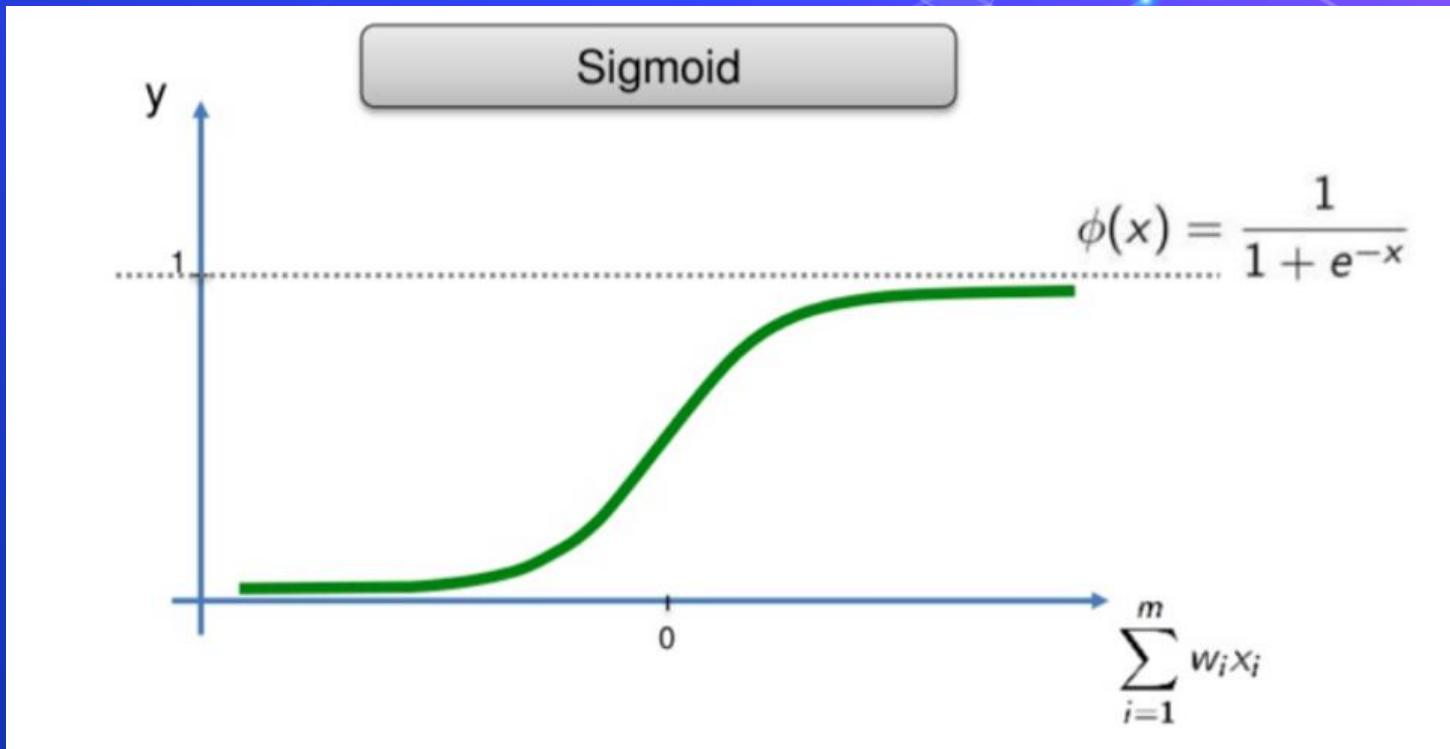
The Logistic Function

$$y = \frac{1}{1 + e^{-f(x_1, x_2, \dots, x_n)}} \in (0, 1)$$

where

$$f(x_1, x_2, \dots, x_n) = a_0 + a_1 x_1 + \dots + a_n x_n \in (-\infty, +\infty)$$

Logistic Regression



Logistic Regression

```
1 from sklearn.linear_model import LogisticRegression  
2 from sklearn.metrics import classification_report  
3 from sklearn.metrics import accuracy_score  
4 from sklearn.metrics import confusion_matrix  
5  
6  
7 model = LogisticRegression()  
8 model.fit(x_train, y_train)  
9 y_pred = model.predict(x_test)  
10  
11 print(confusion_matrix(y_test, y_pred))  
12 print(accuracy_score(y_test, y_pred))  
13 print(classification_report(y_test, y_pred))
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - **K-Nearest Neighbors (KNN)**
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

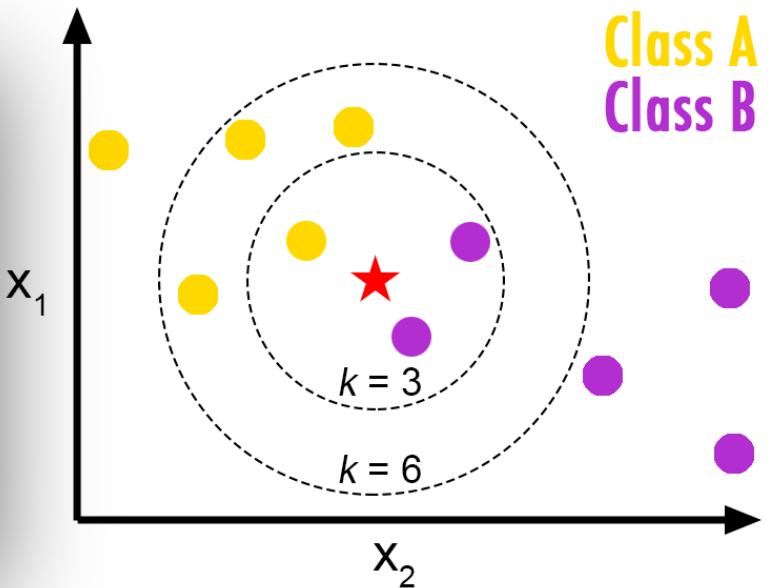
Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

K-Nearest Neighbors (KNN)

```
● ● ●  
1 from sklearn.neighbors import KNeighborsClassifier  
2 from sklearn.metrics import classification_report  
3 from sklearn.metrics import accuracy_score  
4 from sklearn.metrics import confusion_matrix  
5  
6  
7 model = KNeighborsClassifier()  
8 model.fit(x_train, y_train)  
9 y_pred = model.predict(x_test)  
10  
11 print(confusion_matrix(y_test, y_pred))  
12 print(accuracy_score(y_test, y_pred))  
13 print(classification_report(y_test, y_pred))
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - **Naive Bayes**
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

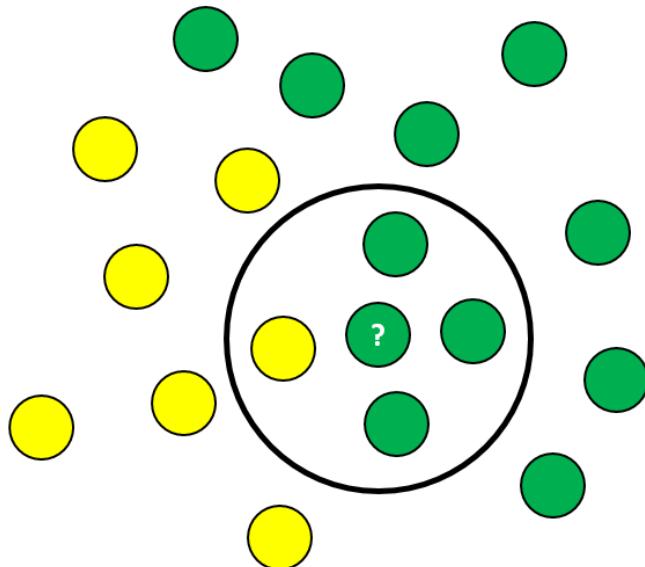
- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Naive Bayes



$$P(\text{yellow}) = \frac{7}{17}$$

$$P(\text{green}) = \frac{10}{17}$$

$$P'(? | \text{green}) = \frac{3}{10}$$

$$P'(? | \text{yellow}) = \frac{1}{7}$$

prior probabilities

number of samples in a given class
divided by the total number of samples

we consider just the vicinity
of the new sample we want to classify

$$P''(\text{? is green}) = P(\text{green}) * P'(\text{?} | \text{green}) = \frac{10}{17} * \frac{3}{10} = \frac{30}{170}$$

$$P''(\text{? is yellow}) = P(\text{yellow}) * P'(\text{?} | \text{yellow}) = \frac{7}{17} * \frac{1}{7} = \frac{7}{119}$$

Naive Bayes



```
1 from sklearn.naive_bayes import MultinomialNB
2 from sklearn.metrics import classification_report
3 from sklearn.metrics import accuracy_score
4 from sklearn.metrics import confusion_matrix
5
6
7 model = MultinomialNB()
8 model.fit(x_train, y_train)
9 y_pred = model.predict(x_test)
10
11 print(confusion_matrix(y_test, y_pred))
12 print(accuracy_score(y_test, y_pred))
13 print(classification_report(y_test, y_pred))
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
 - Evaluating Model Performance

Unsupervised Learning

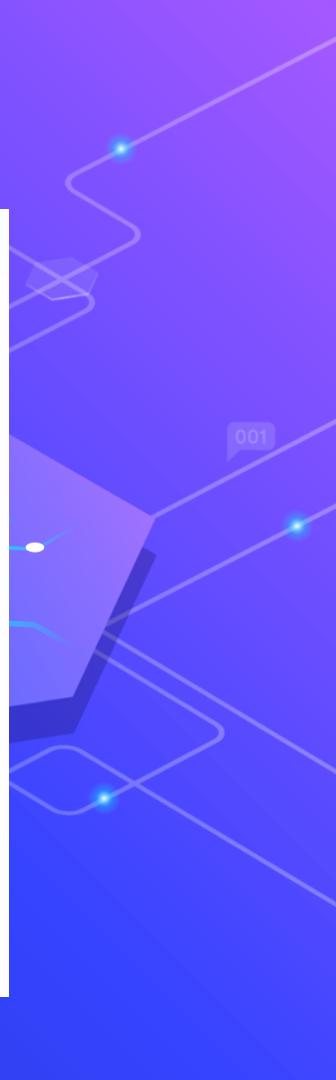
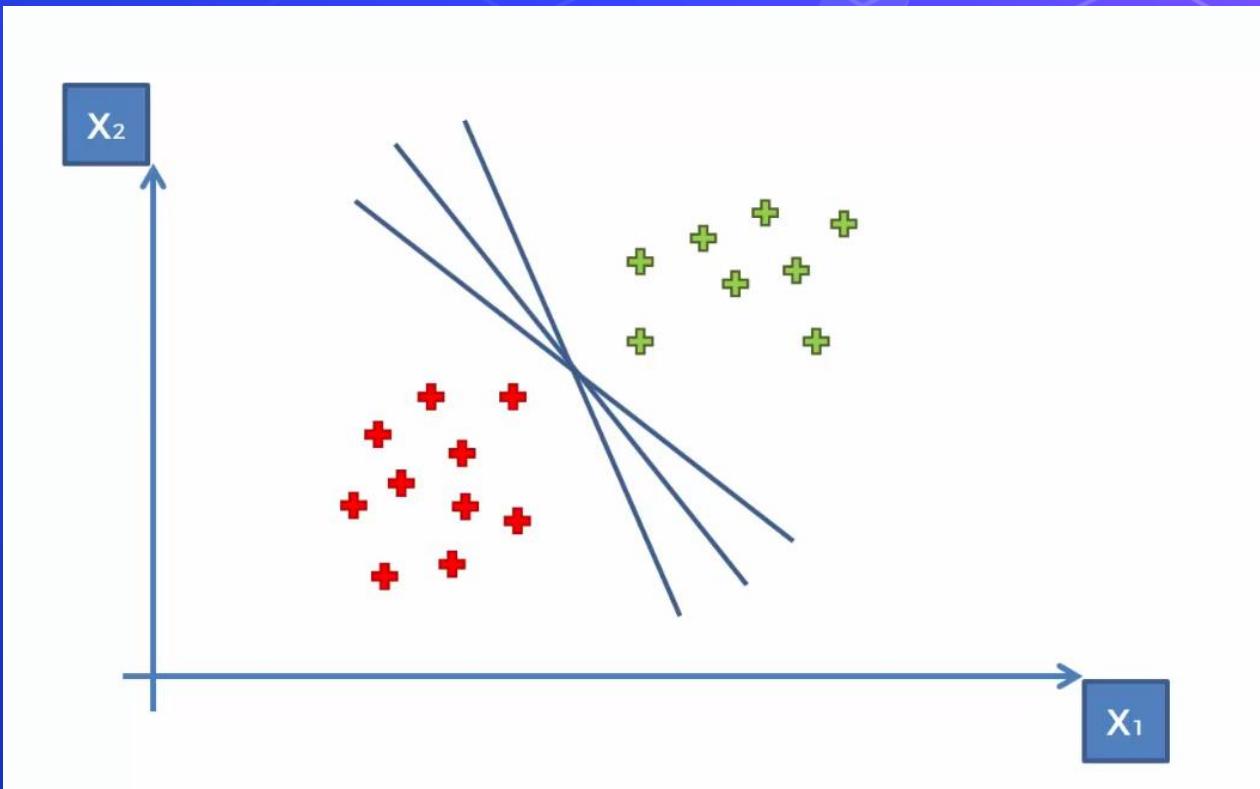
- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

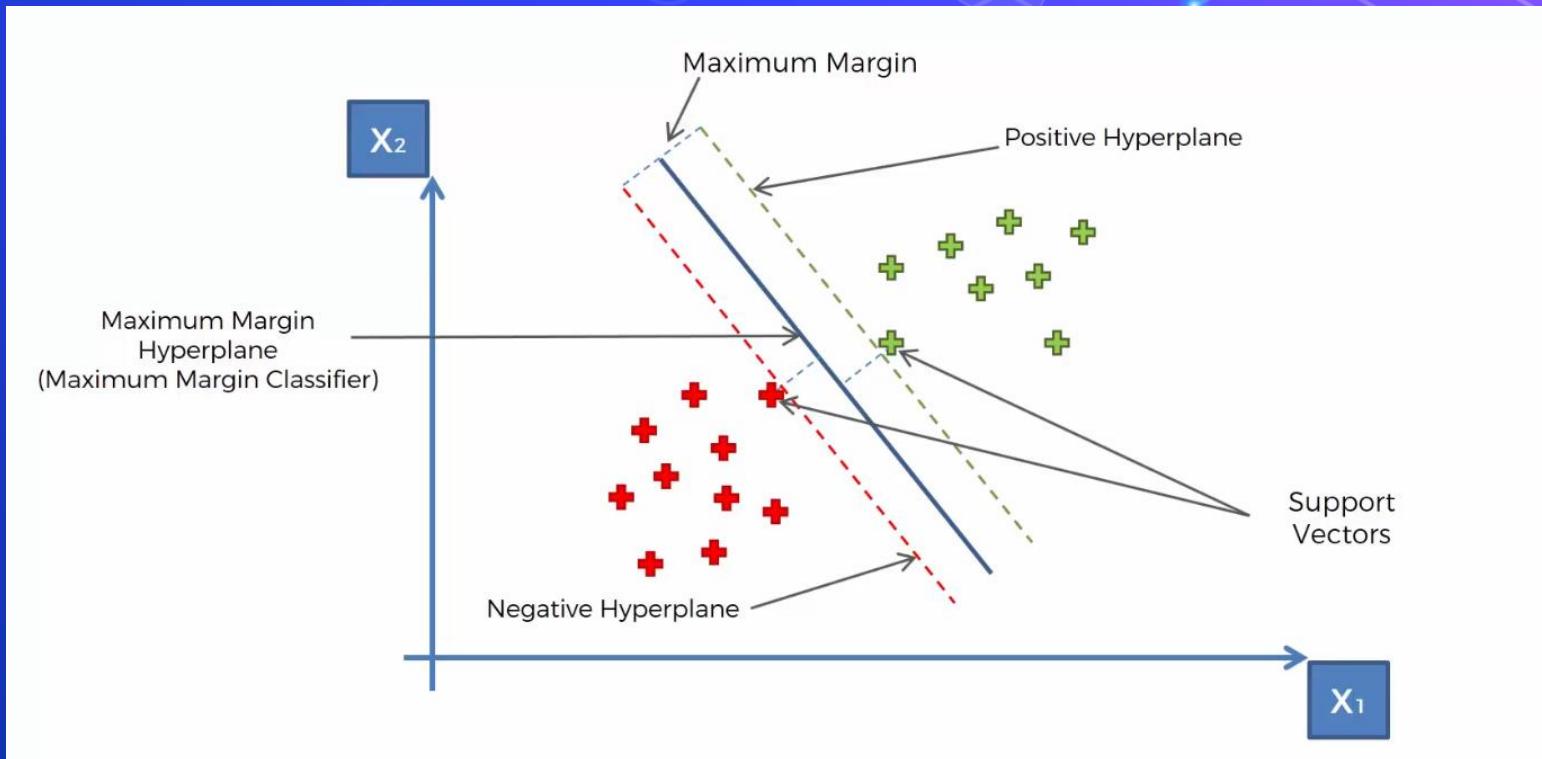
- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

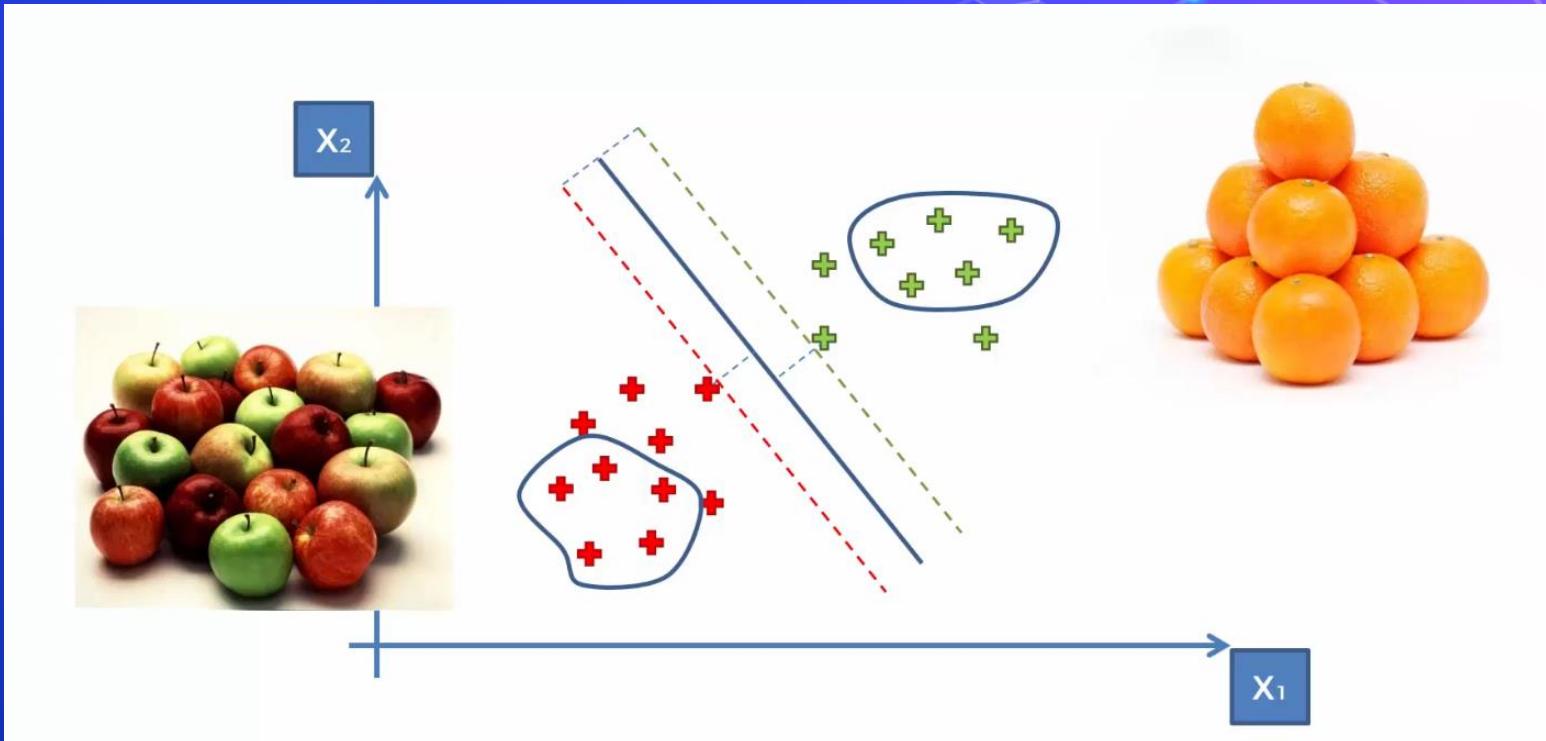
SVM (Support Vector Machine)



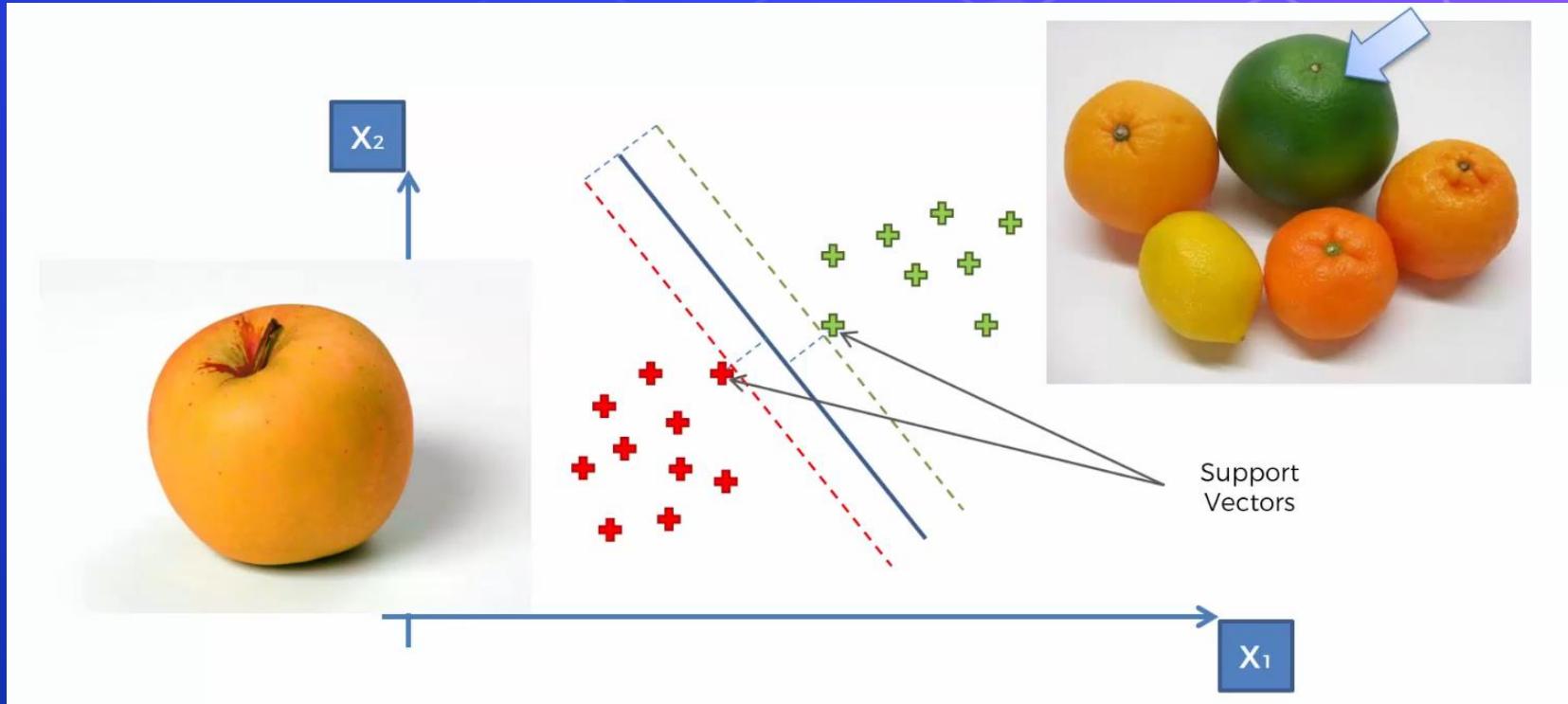
SVM (Support Vector Machine)



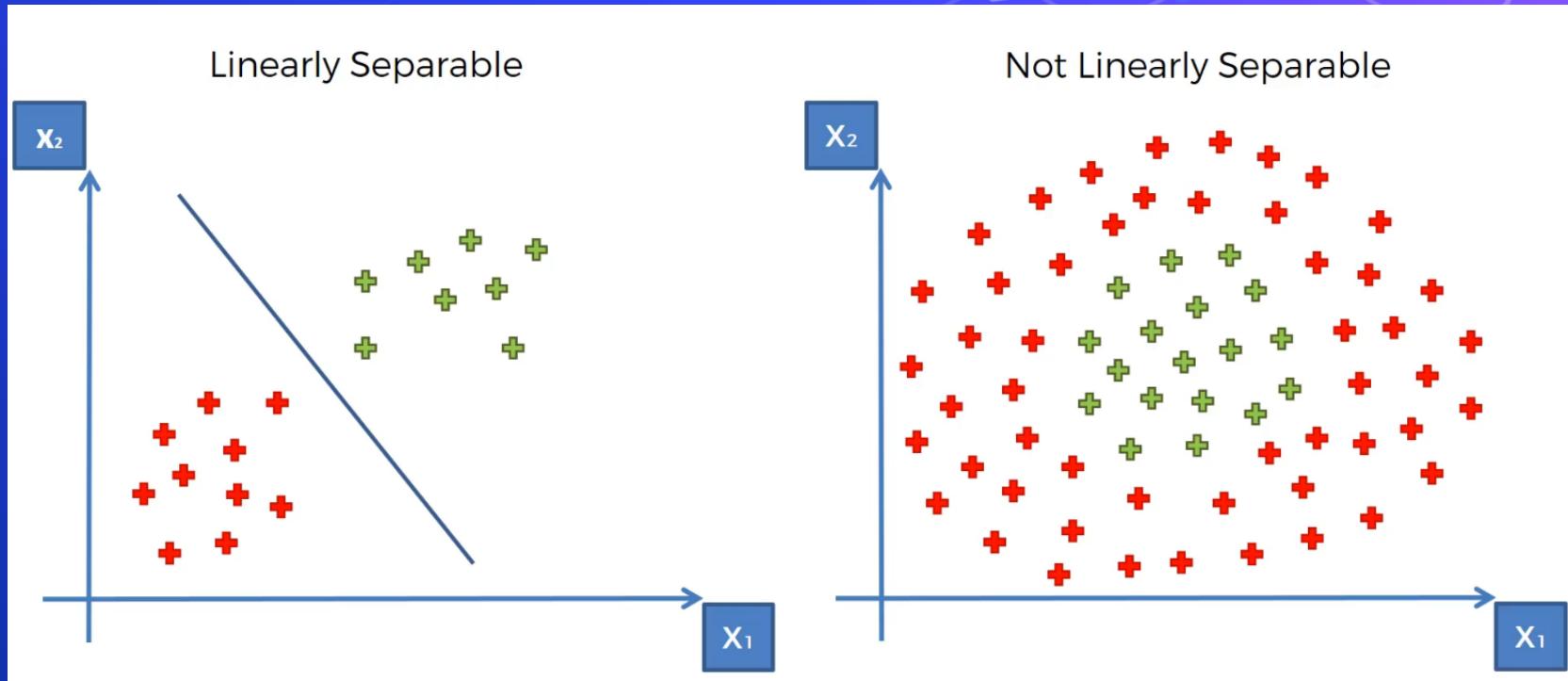
SVM (Support Vector Machine)



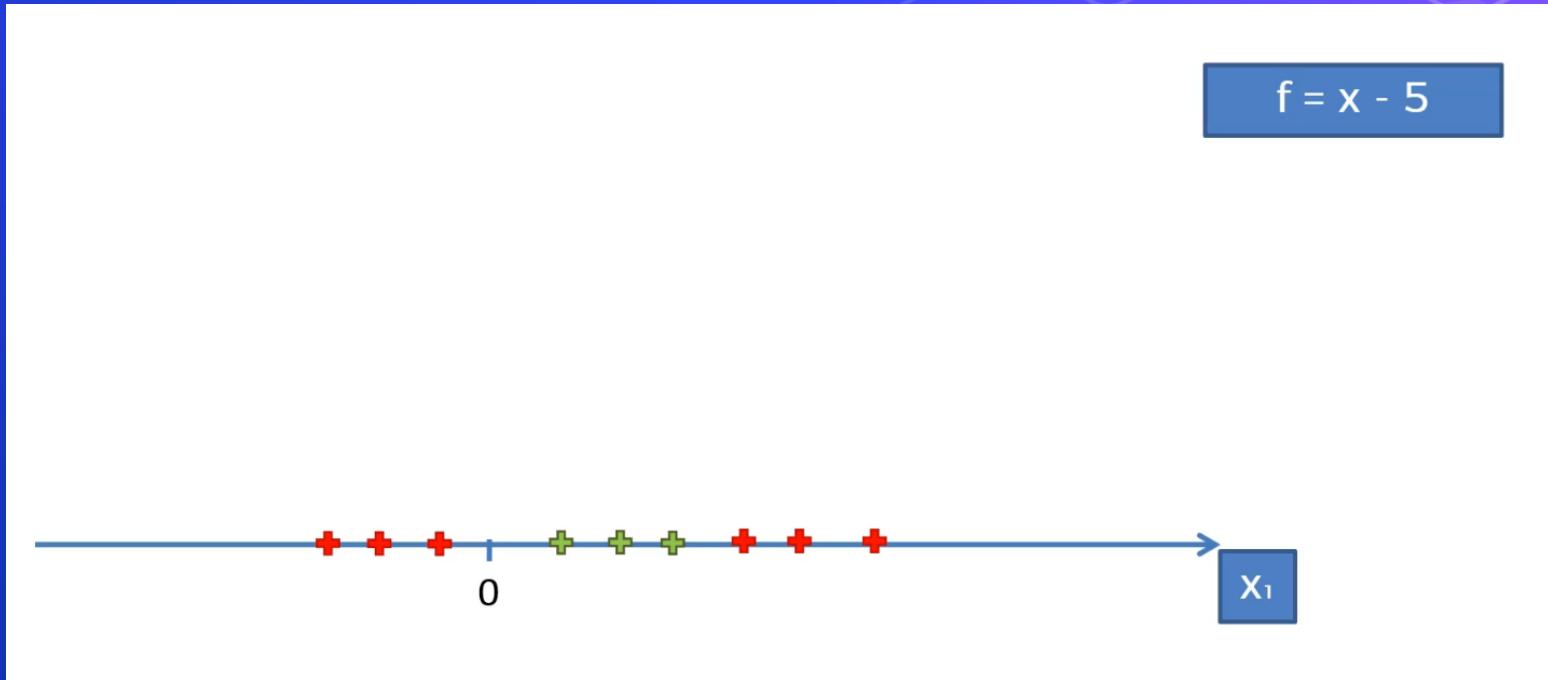
SVM (Support Vector Machine)



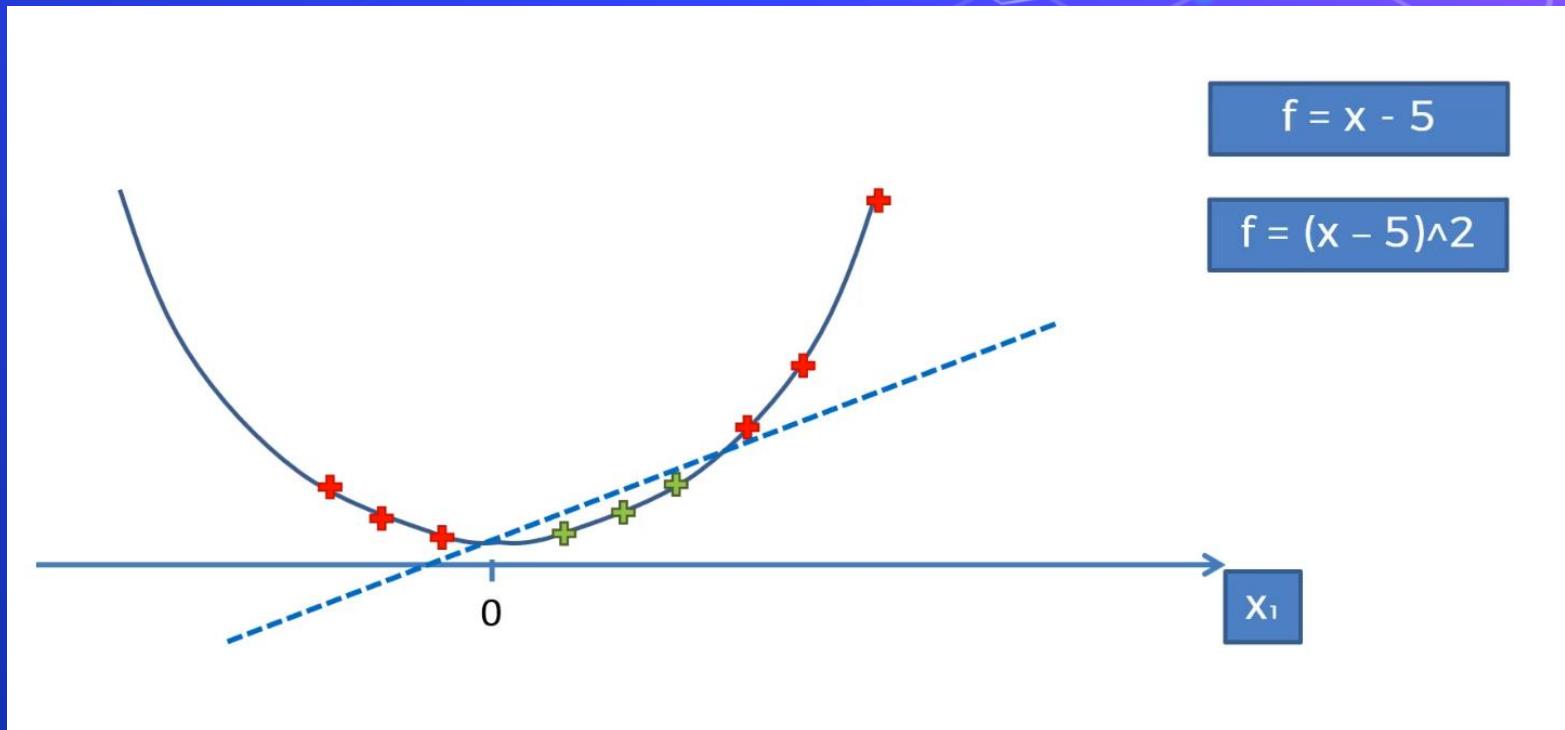
SVM (Support Vector Machine)



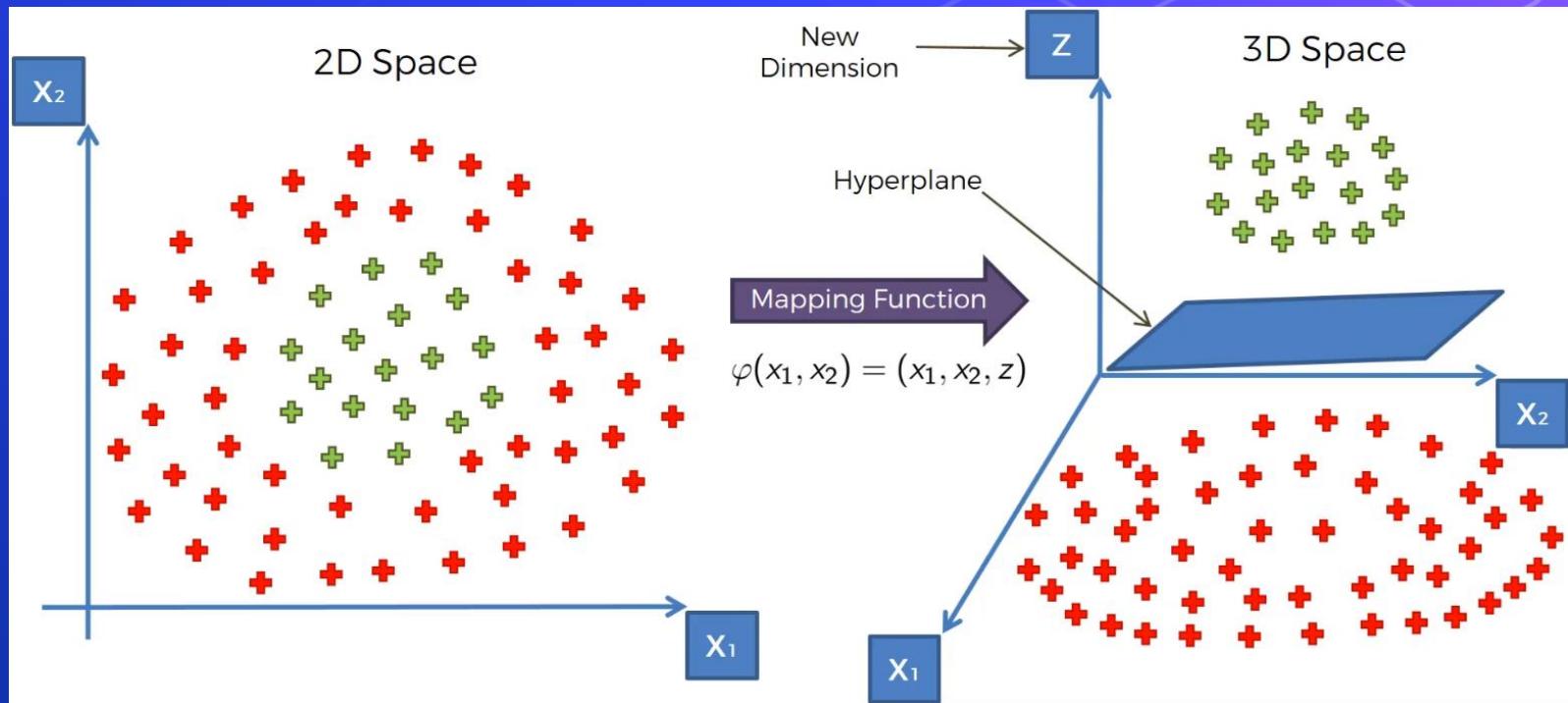
SVM (Support Vector Machine)



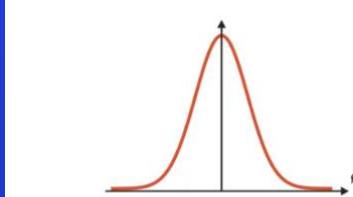
SVM (Support Vector Machine)



SVM (Support Vector Machine)

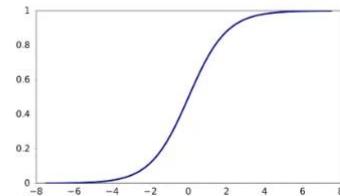


SVM (Support Vector Machine)



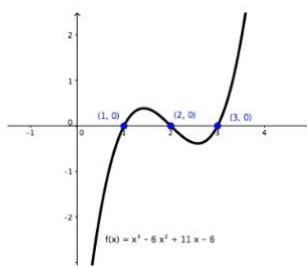
Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^i) = e^{-\frac{\|\vec{x}-\vec{l}^i\|^2}{2\sigma^2}}$$



Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$



Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

SVM (Support Vector Machine)

```
1 from sklearn.svm import SVC
2 from sklearn.metrics import classification_report
3 from sklearn.metrics import accuracy_score
4 from sklearn.metrics import confusion_matrix
5
6
7 model = SVC()
8 model.fit(x_train, y_train)
9 y_pred = model.predict(x_test)
10
11 print(confusion_matrix(y_test, y_pred))
12 print(accuracy_score(y_test, y_pred))
13 print(classification_report(y_test, y_pred))
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
- **Decision Trees**
- Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
- Evaluating Model Performance

Unsupervised Learning

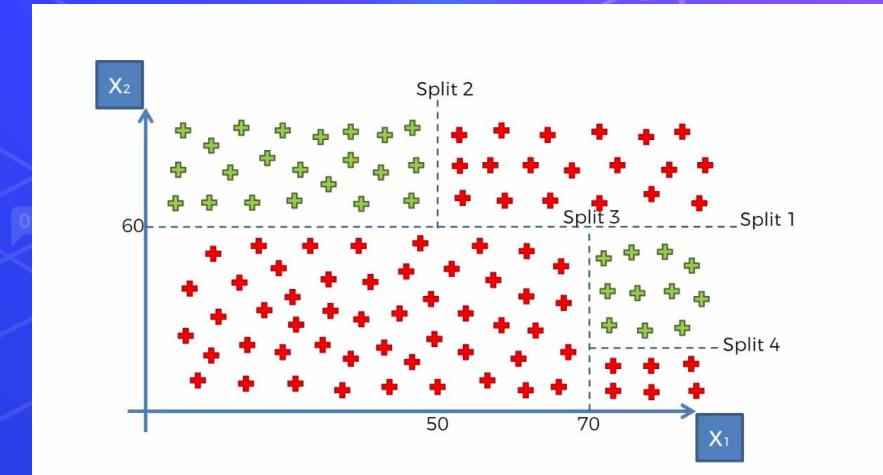
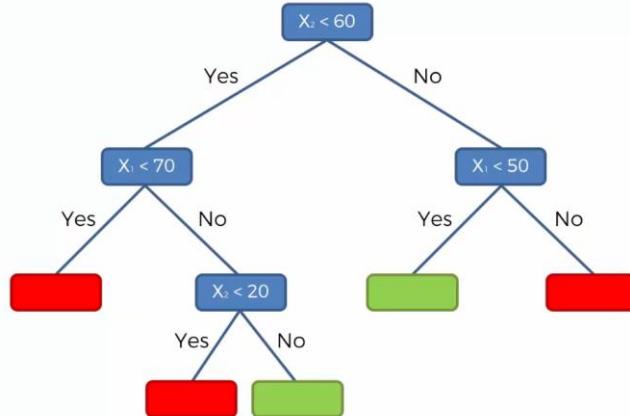
- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Decision Trees



Decision Trees

```
1 from sklearn.tree import DecisionTreeClassifier  
2 from sklearn.metrics import classification_report  
3 from sklearn.metrics import accuracy_score  
4 from sklearn.metrics import confusion_matrix  
5  
6  
7 model = DecisionTreeClassifier()  
8 model.fit(x_train, y_train)  
9 y_pred = model.predict(x_test)  
10  
11 print(confusion_matrix(y_test, y_pred))  
12 print(accuracy_score(y_test, y_pred))  
13 print(classification_report(y_test, y_pred))
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
- Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
- Evaluating Model Performance

Unsupervised Learning

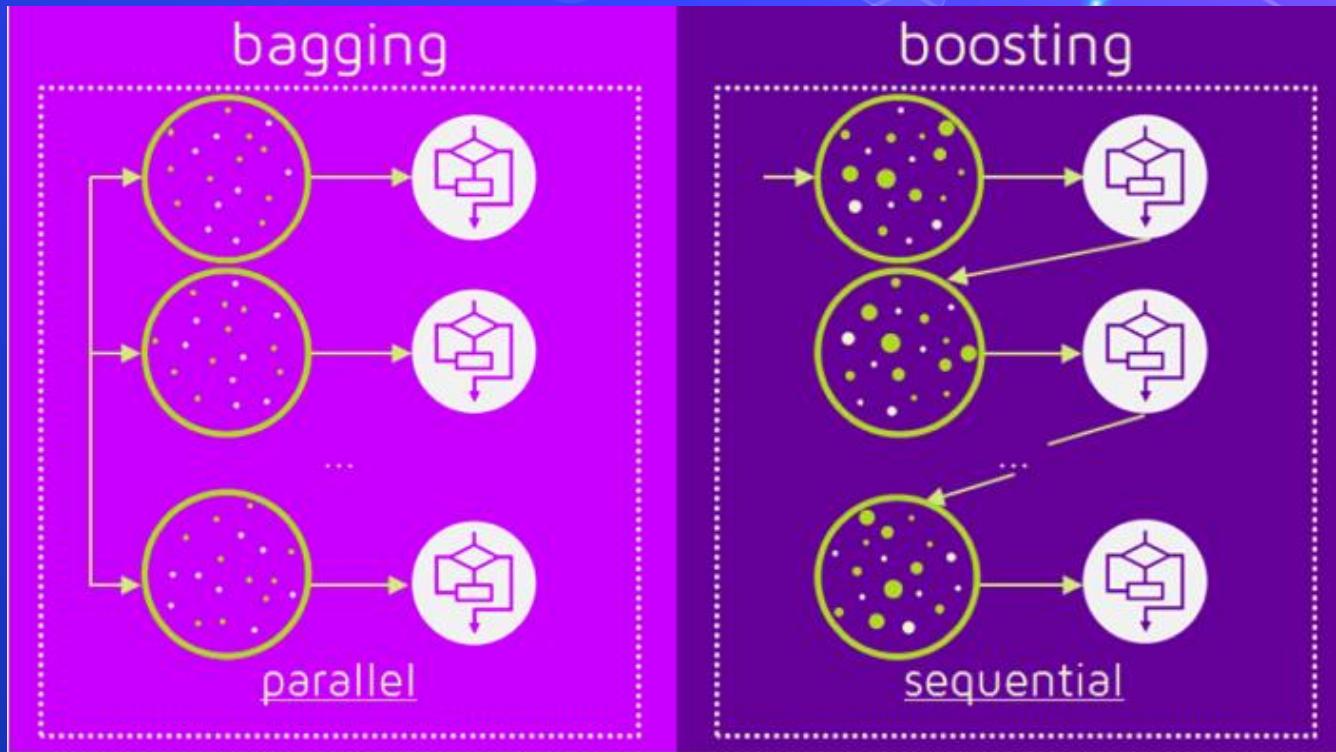
- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

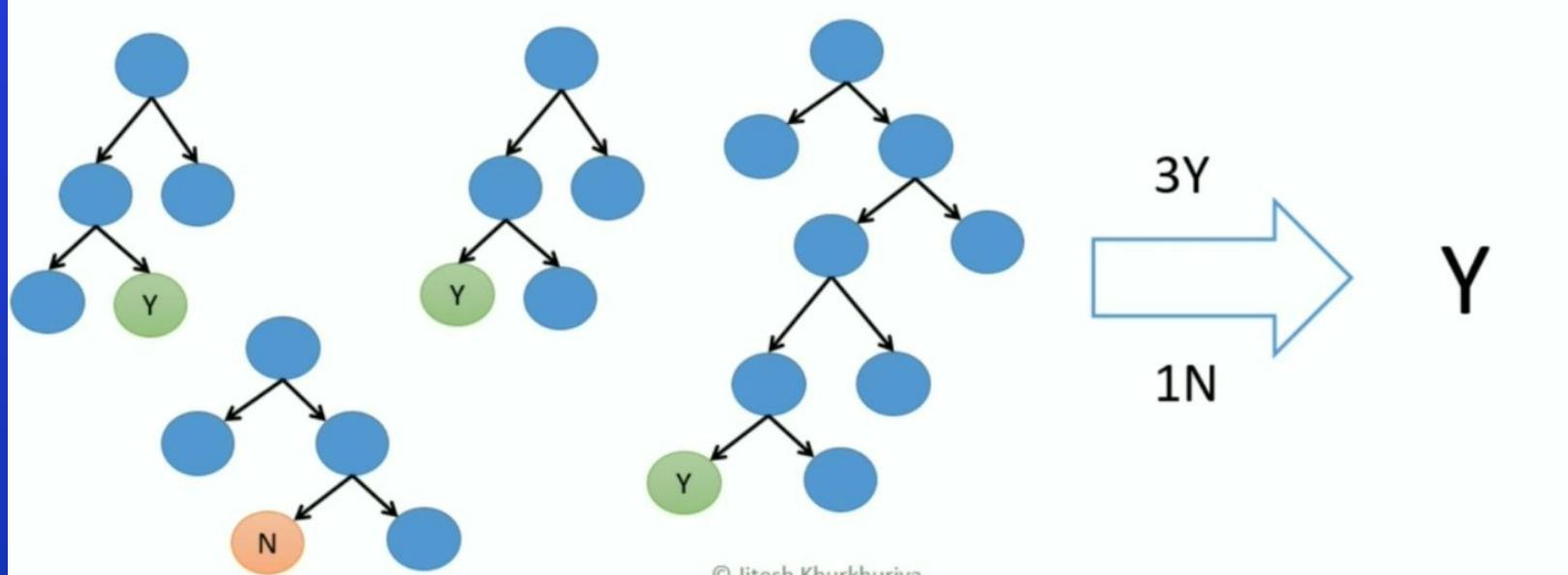
- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

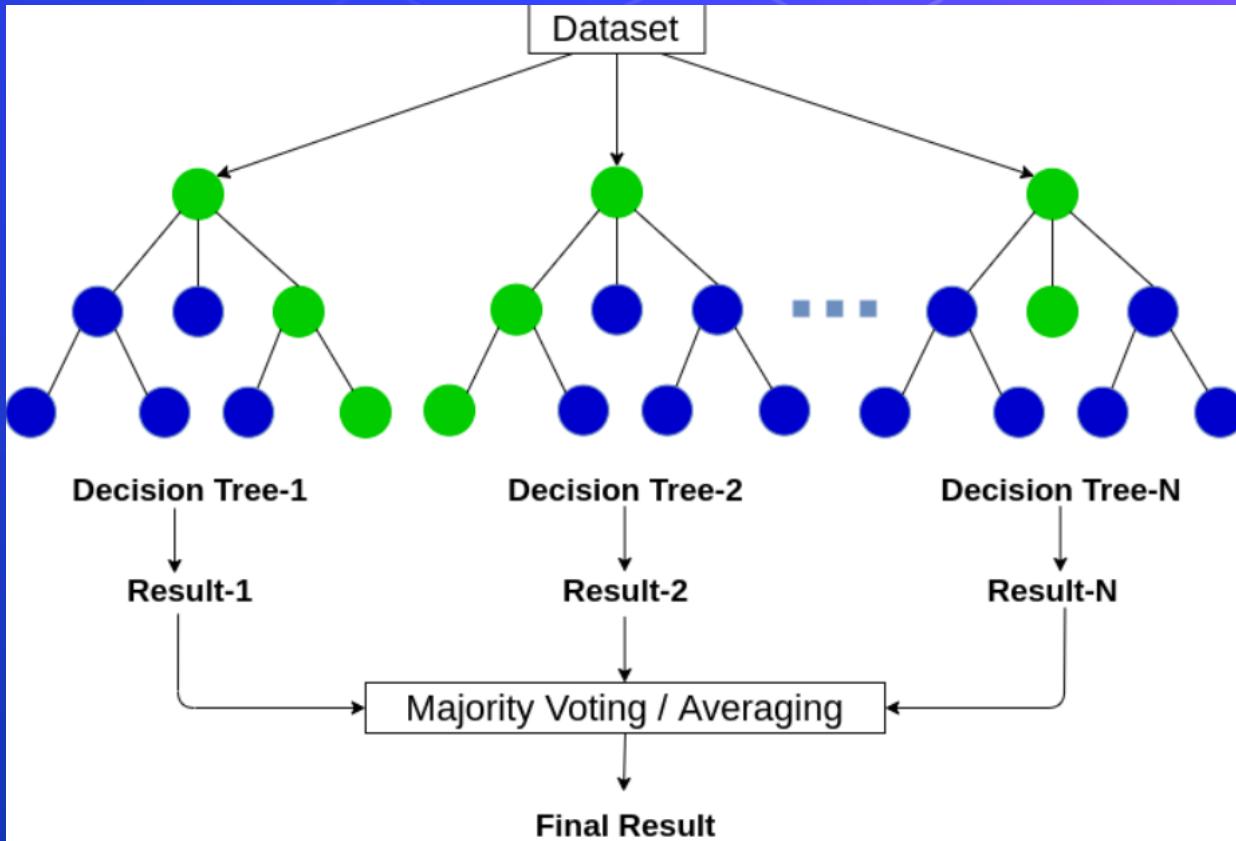
What is Bagging & Boosting



What is Bagging & Boosting



Random Forests

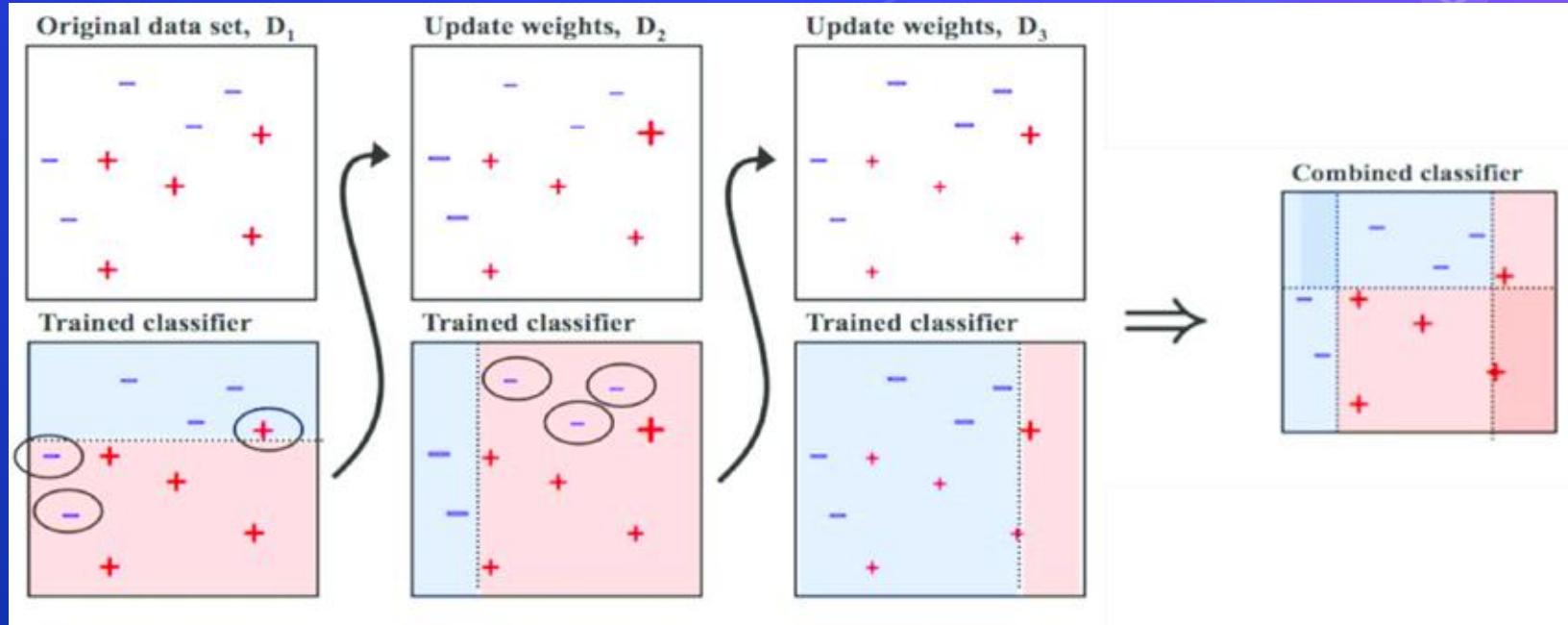


Random Forests

```
1 from sklearn.ensemble import RandomForestClassifier  
2 from sklearn.metrics import classification_report  
3 from sklearn.metrics import accuracy_score  
4 from sklearn.metrics import confusion_matrix  
5  
6  
7 model = RandomForestClassifier()  
8 model.fit(x_train, y_train)  
9 y_pred = model.predict(x_test)  
10  
11 print(confusion_matrix(y_test, y_pred))  
12 print(accuracy_score(y_test, y_pred))  
13 print(classification_report(y_test, y_pred))
```



What is Bagging & Boosting



XGBoost

```
1 import xgboost as xgb
2 from sklearn.metrics import classification_report
3 from sklearn.metrics import accuracy_score
4 from sklearn.metrics import confusion_matrix
5
6
7 model = xgb.XGBClassifier()
8 model.fit(x_train, y_train)
9 y_pred = model.predict(x_test)
10
11 print(confusion_matrix(y_test, y_pred))
12 print(accuracy_score(y_test, y_pred))
13 print(classification_report(y_test, y_pred))
```



Machine Learning

Supervised Learning

- Regression
 - Simple Linear Regression
 - Multiple Linear Regression
 - Polynomial Regression
 - Evaluating Model Performance
- Classification
 - Logistic Regression
 - K-Nearest Neighbors (KNN)
 - Naive Bayes
 - SVM
 - Decision Trees
 - Ensemble Methods
 - What is Bagging & Boosting
 - Random Forests
 - XGBoost
- Evaluating Model Performance

Unsupervised Learning

- Clustering
 - KMeans
 - Hierarchical Clustering
 - Density Based Clustering - DBSCAN
- Association rule mining
 - Apriori
- Dimension Reduction
 - PCA
 - LDA
- Evaluating Model Performance

Model Selection & Evaluation

- Cross Validation
- Hyperparameter Tuning
 - Grid Search
 - Randomized Search

Recommendation Systems

Evaluating Model Performance (Confusion Matrix)

		Actual Value	
		Positive	Negative
Predicted Value	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

- True Positive (TP) : Observation is positive, and is predicted to be positive.
- False Negative (FN) : Observation is positive, but is predicted negative.
- True Negative (TN) : Observation is negative, and is predicted to be negative.
- False Positive (FP) : Observation is negative, but is predicted positive.

Evaluating Model Performance (Confusion Matrix)

Actual Values			
		1	0
Predicted Values	1	TRUE POSITIVE  TRUE NEGATIVE 	FALSE POSITIVE  FALSE NEGATIVE 
	0	TYPE 1 ERROR	TYPE 2 ERROR

True positives (TP): actuals are positives and are predicted as positives.

You predicted that a woman is pregnant and she actually is.

False positives (FP) - Type 1 Error: actuals are negatives and are predicted as positives.

You predicted that a man is pregnant but he actually is not.

False negatives (FN) - Type 2 Error: actuals are positives and are predicted as negatives.

You predicted that a woman is not pregnant but she actually is.

True negatives (TN): actuals are negatives and are predicted as positives.

You predicted that a man is not pregnant and he actually is not.

Evaluating Model Performance (Confusion Matrix)

```
1 from sklearn.metrics import confusion_matrix  
2  
3  
4 confusion_matrix(y_pred, y_true)  
5  
6 """  
7 array([[43,  2],  
8        [11, 87]], dtype=int64)  
9 """
```



Evaluating Model Performance (Accuracy)

Accuracy

- It's a measure of how good the model is.
- It's the ratio of the true predicted values over all the values.

Accuracy:

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

Recall:

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

Precision:

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

F₁ score:

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

Evaluating Model Performance (Accuracy)

```
1 from sklearn.metrics import accuracy_score  
2  
3  
4 accuracy_score(y_pred, y_true)  
5  
6 """  
7 0.9090909090909091  
8 """
```



Evaluating Model Performance (Recall or Sensitivity)

- TP will be that the COVID 19 residents diagnosed with COVID-19.
- TN will be that healthy residents are with good health.
- FP will be that those actually healthy residents are predicted as COVID 19 residents.
- FN will be that those actual COVID 19 residents are predicted as the healthy residents.

which scenario do you think will have the highest cost?

Imagine that if we predict COVID-19 residents as healthy patients and they do not need to quarantine, there would be a massive number of COVID-19 infections. The cost of false negatives is much higher than the cost of false positives.

Case 1

COVID 19 = 1
Healthy = 0

Cost of FN > Cost of FP

Actual

Healthy predicted as sick

Sick predicted as healthy

Predict	COVID 19 (1)	Diagnosed COVID 19 (1)	Diagnosed Healthy (0)
Covid 19 (1)	TP	FP	
Healthy (0)	FN	TN	

Evaluating Model Performance (Recall or Sensitivity)

Recall (Sensitivity)

- It's a ratio of True Positives to all the positives in your data.
- Low recall: the more False Negatives the model predicts, the lower the recall.

Accuracy:

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

Recall:

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

Precision:

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

F₁ score:

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

Evaluating Model Performance (Recall or Sensitivity)

```
1 from sklearn.metrics import recall_score  
2  
3  
4 recall_score(y_pred, y_true)  
5  
6 """  
7 0.9090909090909091  
8 """
```



Evaluating Model Performance (Precision)

- TP will be that spam emails are placed in the spam folder.
- TN will be that important emails are received.
- FP will be that important emails are placed in the spam folder.
- FN will be that spam emails are received.

which scenario do you think will have the highest cost?

Well, since missing important emails will clearly be more of a problem than receiving spam, we can say that in this case, FP will have a higher cost than FN.

Case 2

Spam = 1
Not Spam = 0

Cost of FP > Cost of FN

Actual

Not spam predicted as spam

Predict

	Spam (1)	Not Spam (0)
Spam (1)	TP	FP
Not Spam (0)	FN	TN

Spam predicted as not spam

Evaluating Model Performance (Precision)

Precision

- It's a ratio of True Positives to all the positives predicted by the model.
- Low precision: the more False positives the model predicts, the lower the precision.

Accuracy:

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

Recall:

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

Precision:

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

F₁ score:

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

Evaluating Model Performance (Precision)

```
1 from sklearn.metrics import precision_score  
2  
3  
4 precision_score(y_pred, y_true)  
5  
6 """  
7 0.9090909090909091  
8 """
```



Evaluating Model Performance

Case 1



COVID 19/ Healthy

Case 2



Spam/Not Spam

Cost of **FN** > Cost of **FP**

Cost of **FP** > Cost of **FN**

Recall

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

Precision

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Evaluating Model Performance (F-1 Score)

F-1 Score

- F-Score called the Harmonic mean of precision and recall.
- F-Score provides a single score that balances both the concerns of precision and recall in one number.
- A good F1 score means that you have low false positives and low false negatives.
- An F1 score is considered perfect when it's 1, while the model is a total failure when it's 0.

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

Accuracy:

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

Recall:

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

Precision:

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

F₁ score:

$$F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

Evaluating Model Performance (F-1 Score)

```
1 from sklearn.metrics import f1_score  
2  
3  
4 f1_score(y_pred, y_true)  
5  
6 """  
7 0.9090909090909091  
8 """
```



Evaluating Model Performance (Classification Report)

```
1 from sklearn.metrics import classification_report  
2  
3  
4 classification_report(y_true, y_pred)  
5  
6 """  
7         precision    recall    f1-score    support  
8     class 0       0.50      1.00      0.67      1  
9     class 1       0.00      0.00      0.00      1  
10    class 2       1.00      0.67      0.80      3  
11 """
```

Evaluating Model Performance (F-beta Score)

F-beta Score

- Generalization of F-Score where beta is a parameter to weight the result of the precision or recall.
- Beta = 0.5 to focus more on precision, It's called F-0.5 Score.
- Beta = 2 to focus more on recall, It's called F-2 Score.
- Beta value should be used depending on your domain knowledge business case.

$$F_{\beta} = (1 + \beta^2) \cdot \frac{\text{precision} \cdot \text{recall}}{(\beta^2 \cdot \text{precision}) + \text{recall}}.$$



Evaluating Model Performance (F-beta Score)

```
1 from sklearn.metrics import fbeta_score
2
3
4 fbeta_score(y_true, y_pred, beta=0.5)
5
6 """
7 0.9090909090909091
8 """
```



Questions ?!



Thanks!

>_ Live long and prosper

