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.

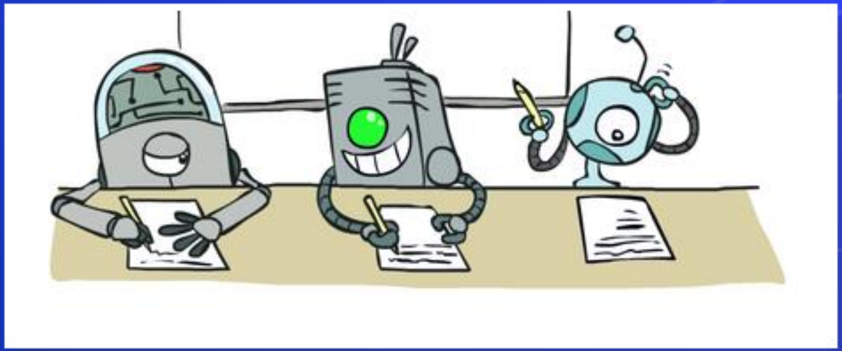
You can find me at
emanehab.ieee@gmail.com

- 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

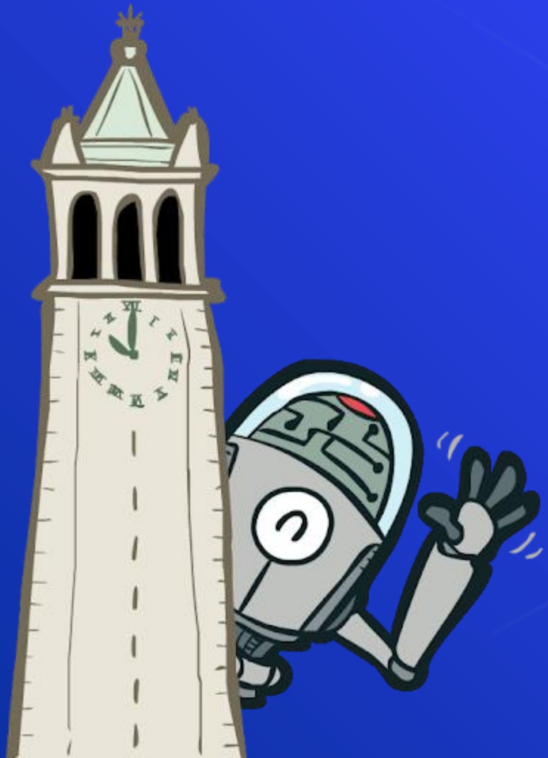
- <u>Machine Learning</u>
- Supervised Learning
  - Regression
    - Simple Linear Regression
    - Multiple Linear Regression
    - Polynomial Regression
    - Evaluating Model Performance
  - Classification
    - Logistic Regression
    - K-Nearest Neighbors (KNN)
    - Naive Bayes
    - SVM
    - 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

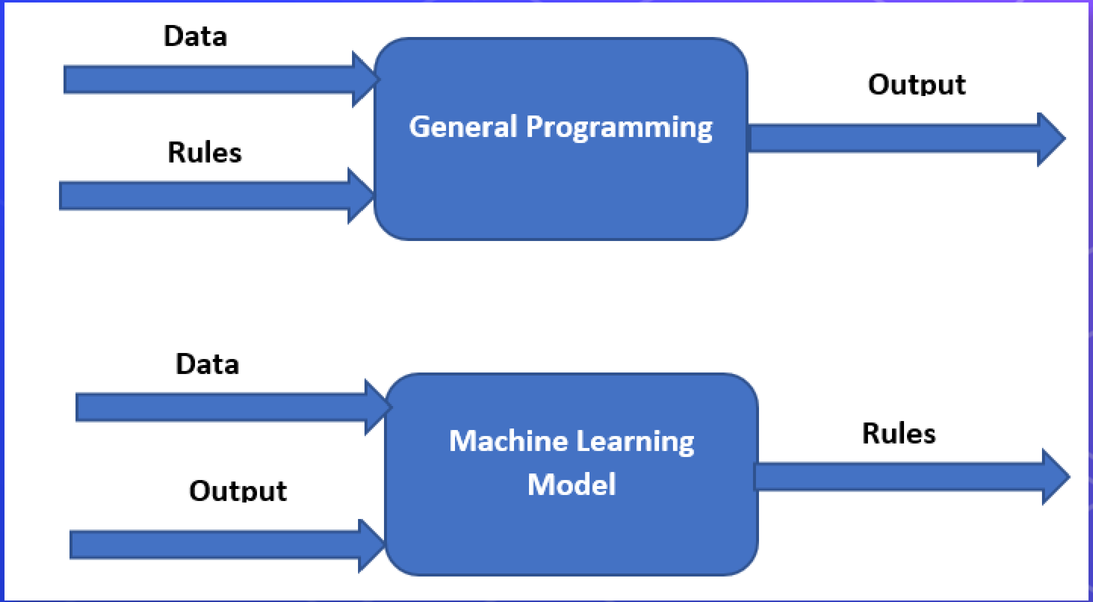
Machine Learning

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.

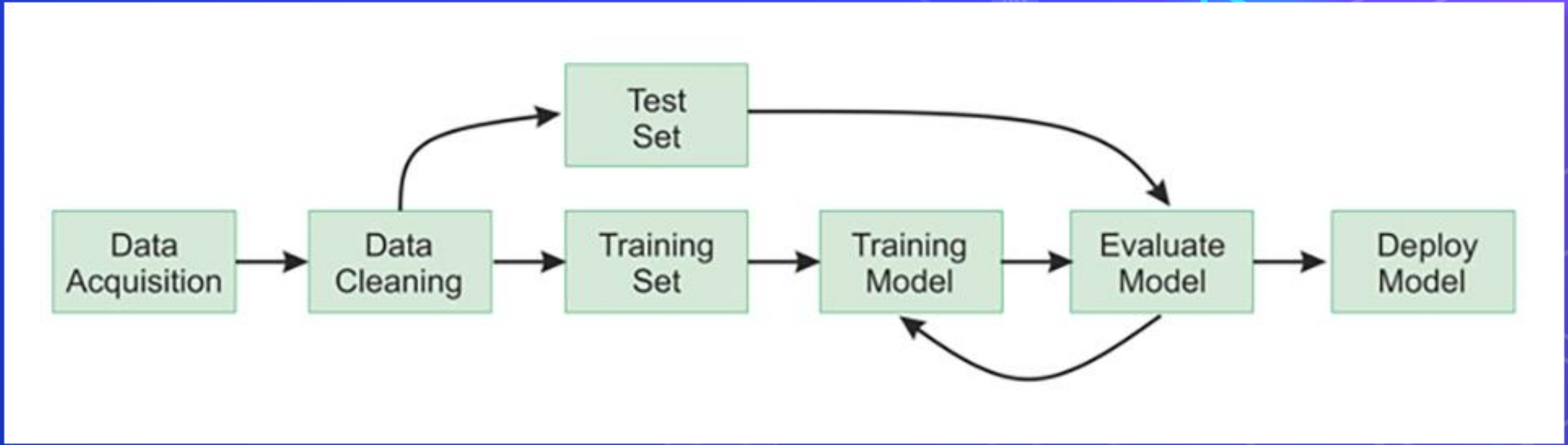


Machine Learning

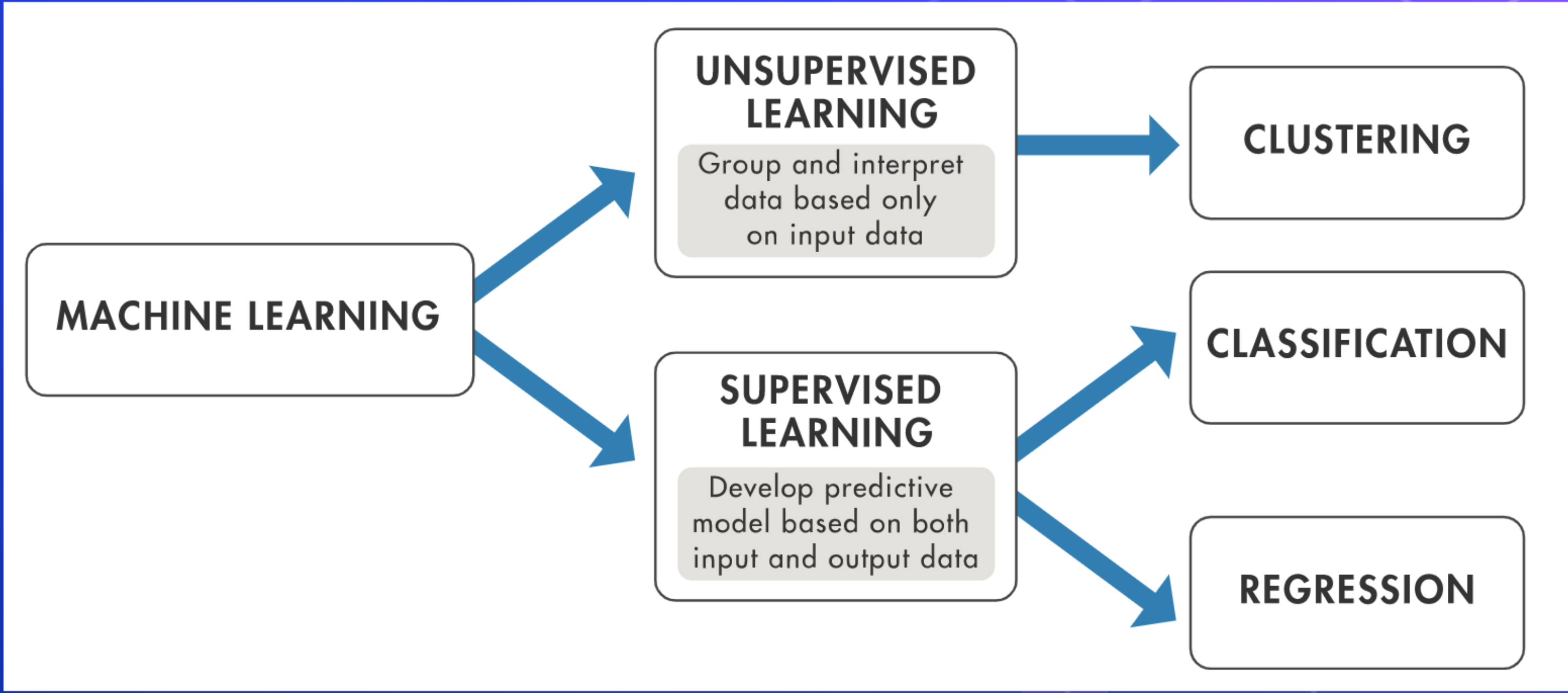




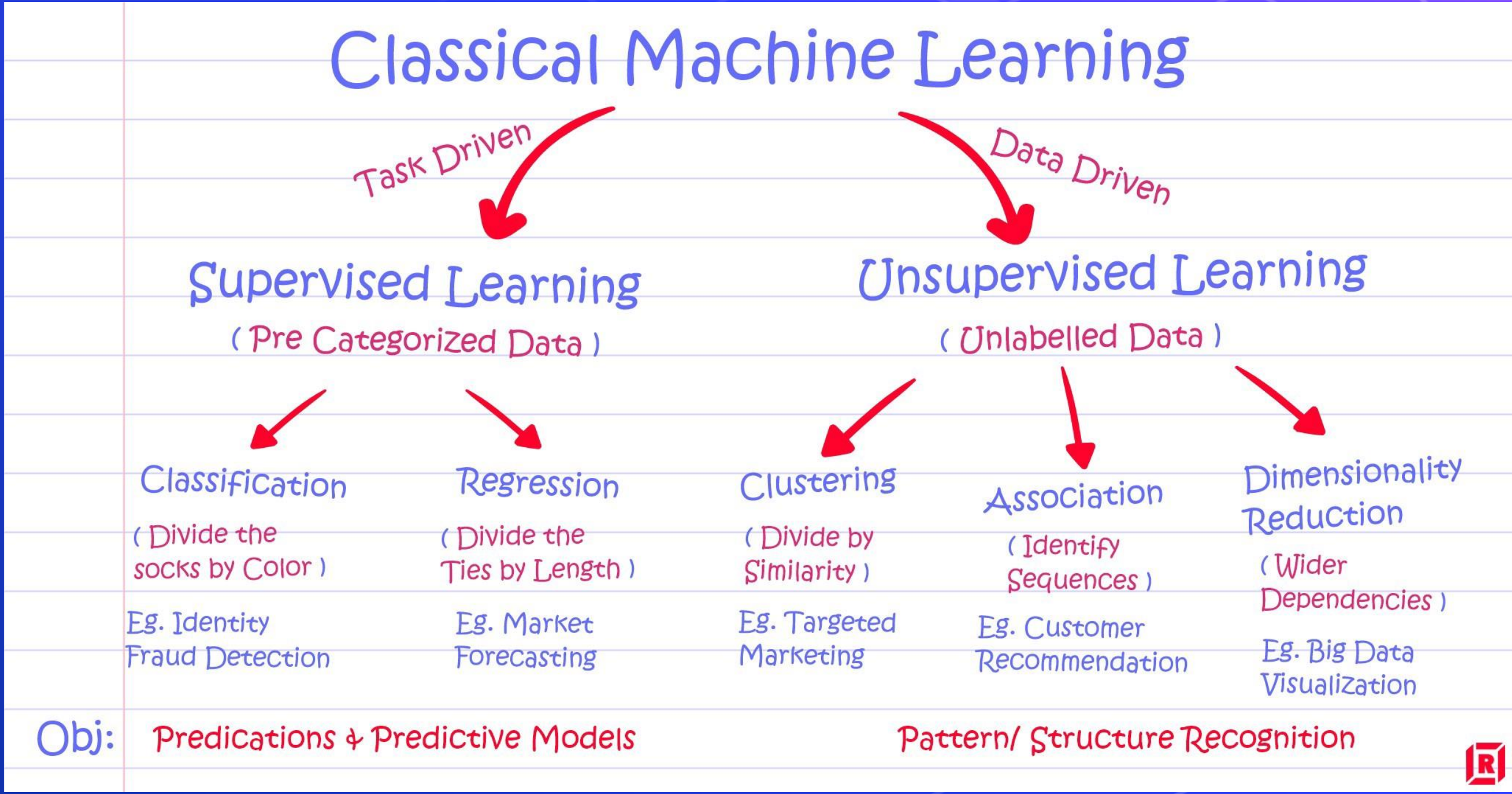
Machine Learning



Machine Learning



Machine Learning



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.

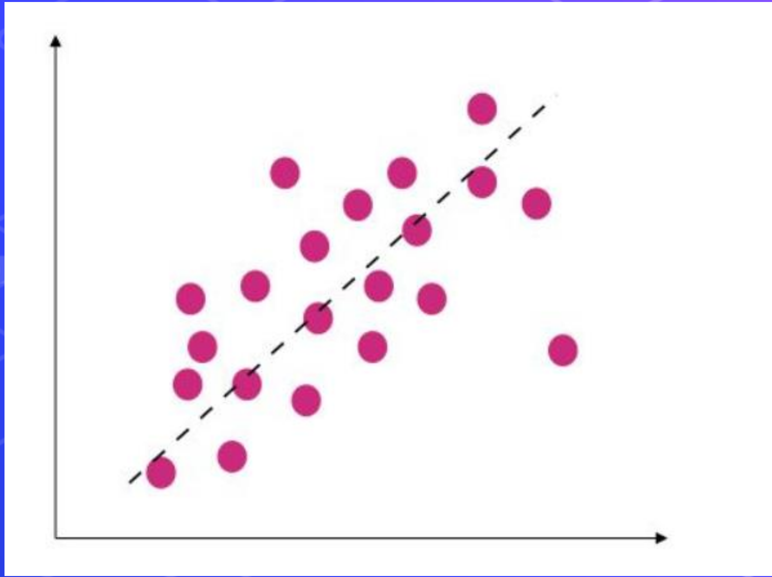
- Machine Learning
- **Supervised Learning**
  - **Regression**
    - Simple Linear Regression
    - Multiple Linear Regression
    - Polynomial Regression
    - Evaluating Model Performance
  - Classification
    - Logistic Regression
    - K-Nearest Neighbors (KNN)
    - Naive Bayes
    - SVM
    - 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.

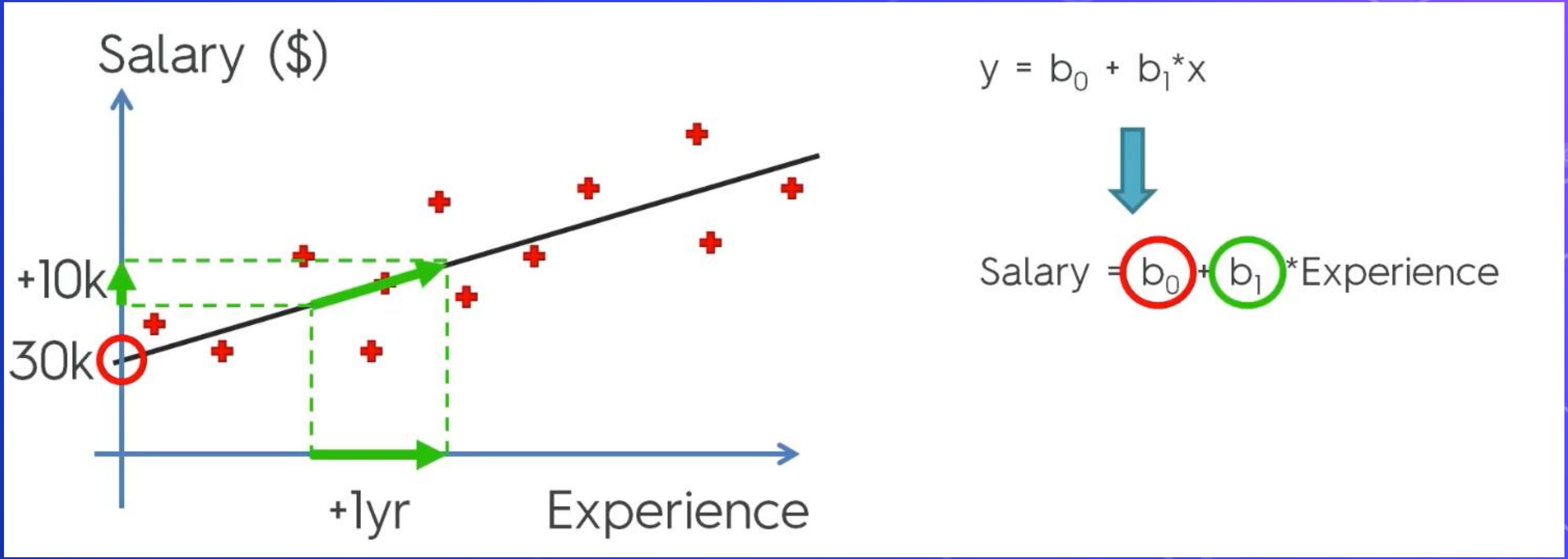
For example,
it can be used to quantify the relative impacts of age, gender, and diet (the predictors) on height (the output or dependent).

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



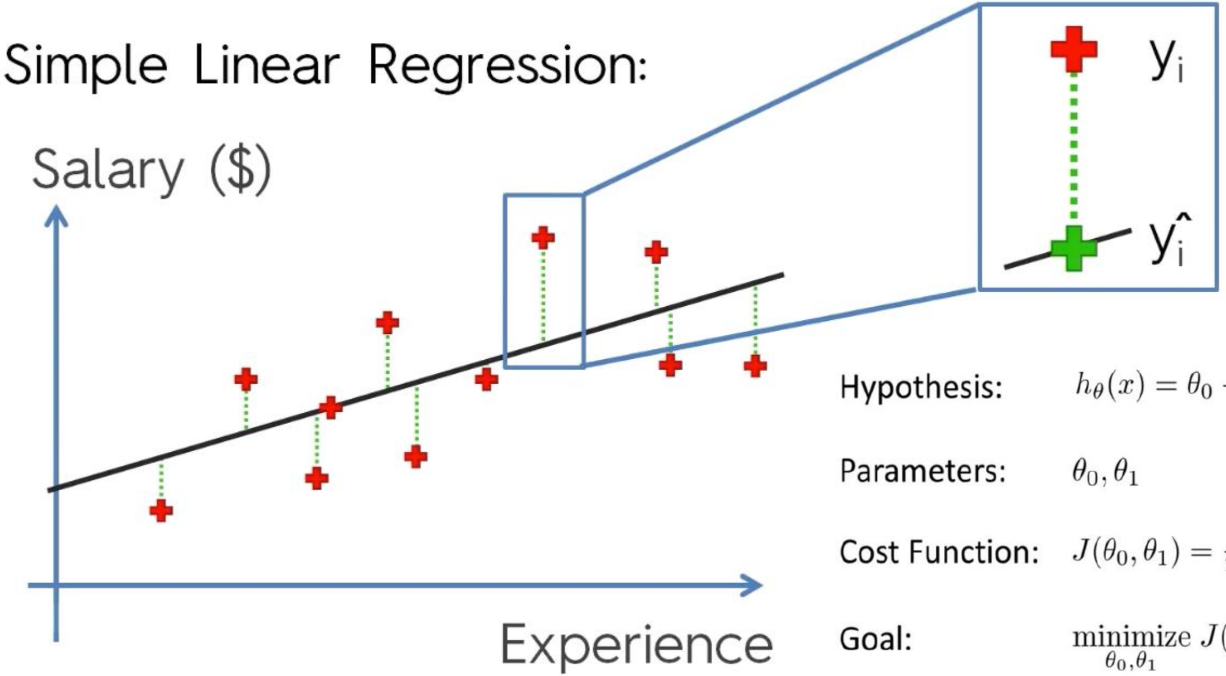
- Machine Learning
- **Supervised Learning**
  - **Regression**
    - **Simple Linear Regression**
    - Multiple Linear Regression
    - Polynomial Regression
    - Evaluating Model Performance
  - Classification
    - Logistic Regression
    - K-Nearest Neighbors (KNN)
    - Naive Bayes
    - SVM
    - 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)



Simple Linear Regression (Calculate Cost Function)

Simple Linear Regression:



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

Parameters: θ_0, θ_1

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

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

Simple Linear Regression (Example)

Theta0 = 5 , theta 1 = 2 Equation $h(x) = 5 + 2x$

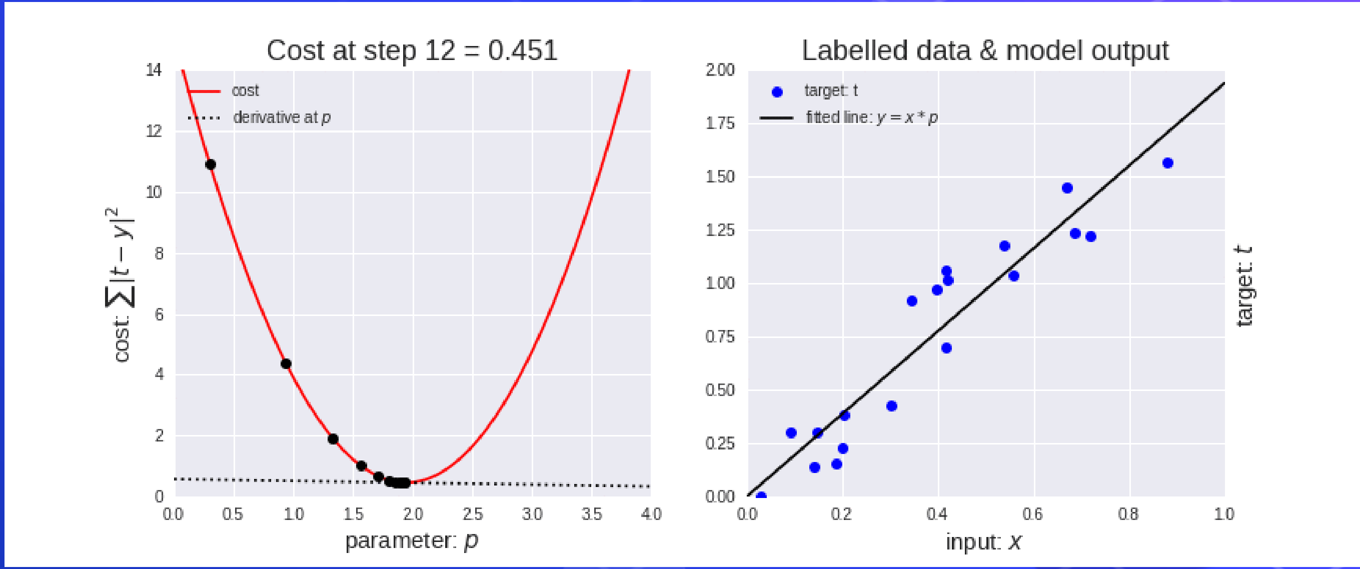
X	Y	h(x)	h(x) - y	$(h(x) - y)^2$
1	7	7	0	0
2	8	9	1	1
2	7	9	2	4
3	9	11	2	4
4	11	13	2	4
5	10	15	5	25
5	12	15	3	9

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

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

J = 47/14 = 3.3

Simple Linear Regression (Minimize cost using Gradient Descent)



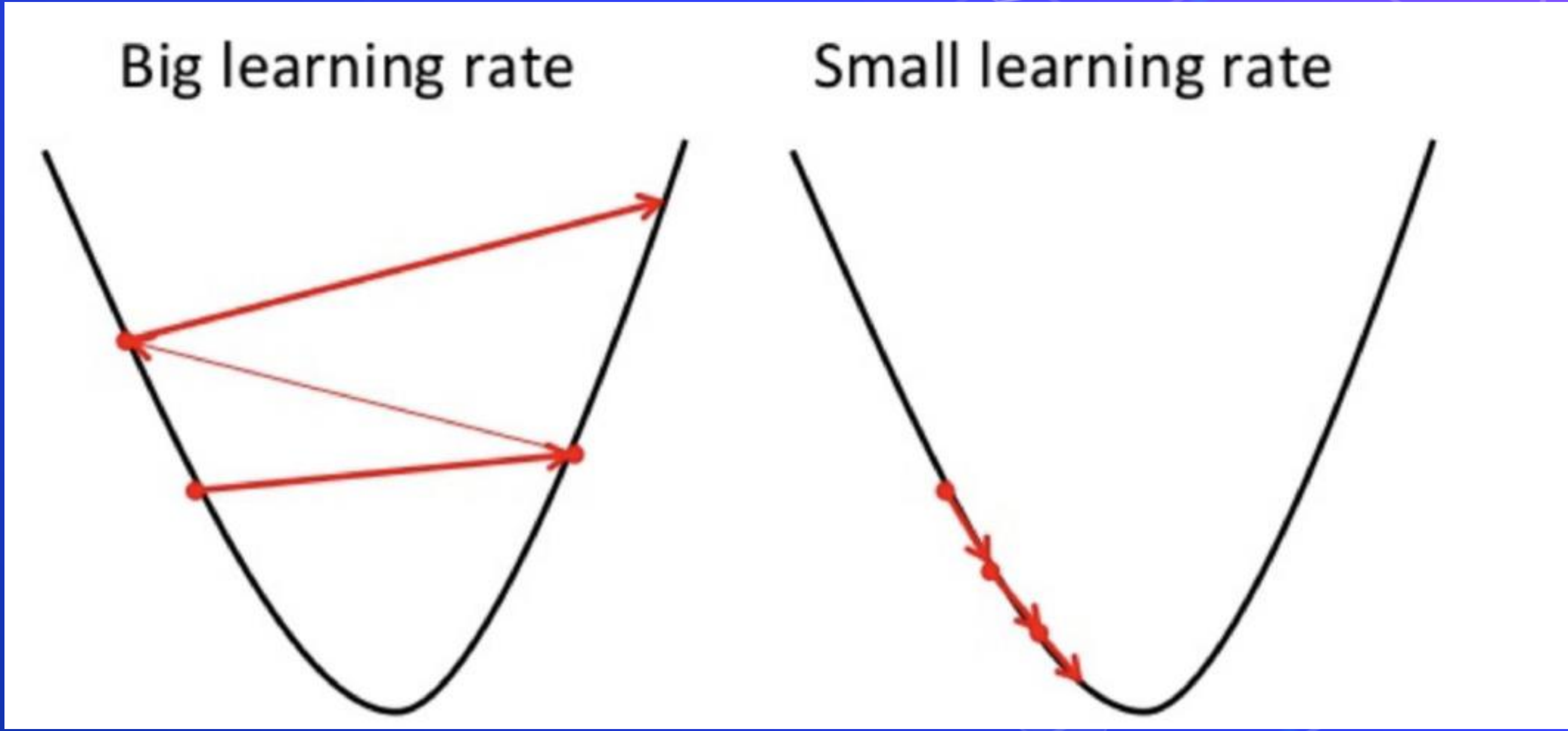
repeat until convergence:

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

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

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

Simple Linear Regression (Alpha constant)



Simple Linear Regression (Code)

```python
from sklearn.linear_model import LinearRegression

# make model object
model = LinearRegression()

# train
model.fit(x_train, y_train)

# test
model.predict(x_test)

# calculate R2 score on training data
model.score(x_train, y_train)

# calculate R2 score on testing data
model.score(x_test, y_test)

# get model parameters
model.coef_
model.intercept_
```

- Machine Learning
- **Supervised Learning**
  - **Regression**
    - Simple Linear Regression
    - **<u>Multiple Linear Regression</u>**
    - 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

Multiple Linear Regression (Update theta values)

repeat until convergence: {

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

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

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

$\cdots$

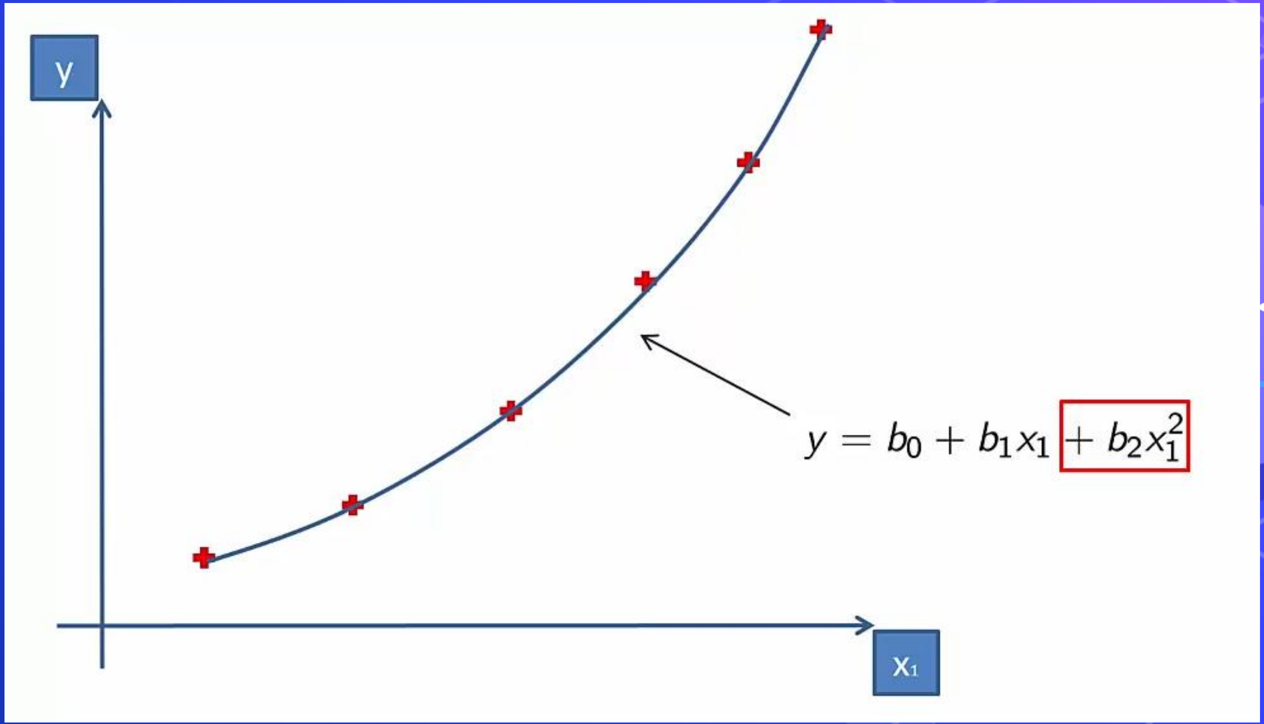
}

Multiple Linear Regression (Code)

```python
from sklearn.linear_model import LinearRegression

# make model object
model = LinearRegression()

# train
model.fit(x_train, y_train)

# test
model.predict(x_test)

# calculate R2 score on training data
model.score(x_train, y_train)

# calculate R2 score on testing data
model.score(x_test, y_test)

# get model parameters
model.coef_
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

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

Polynomial Regression (Code)

```python
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# create poly features for a feature
poly = PolynomialFeatures(degree=2)
x_train_poly = poly.fit_transform(x_train)
x_test_poly = poly.fit_transform(x_test)

# make model object
model = LinearRegression()

# train
model.fit(x_train_poly, y_train)

# test
model.predict(x_test_poly)

# calculate R2 score on training data
model.score(x_train_poly, y_train)

# calculate R2 score on testing data
model.score(x_train_poly, y_train)

# get model parameters
model.coef_
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	

$$R^2 = SSR/SST$$

$$= 1844.12/3028.77$$

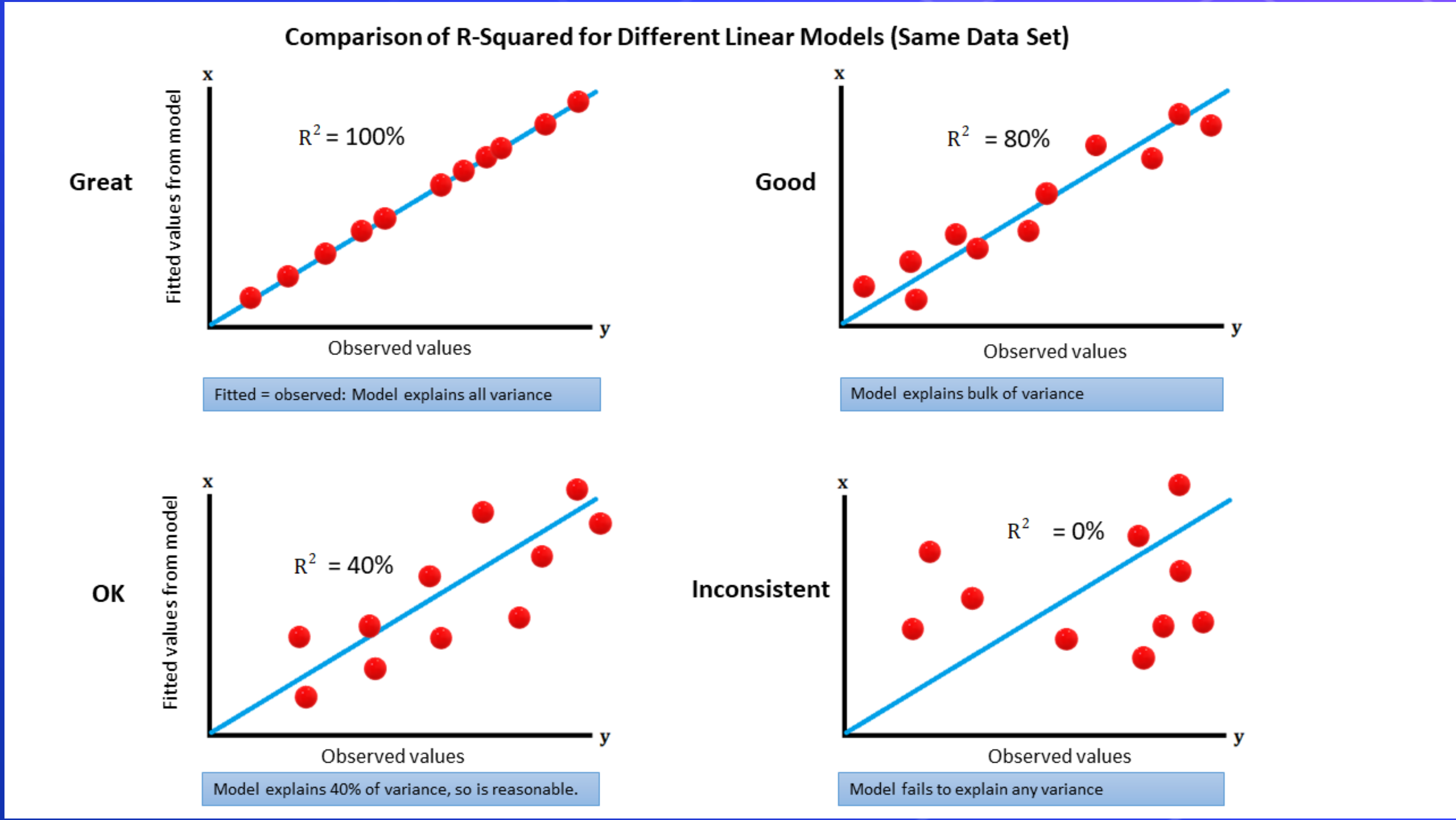
$$= 0.60886$$

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

$\hat{y} \to y$ SSR → SST $R^2 \approx 1$

$(Y - \bar{Y})$^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 (R2)

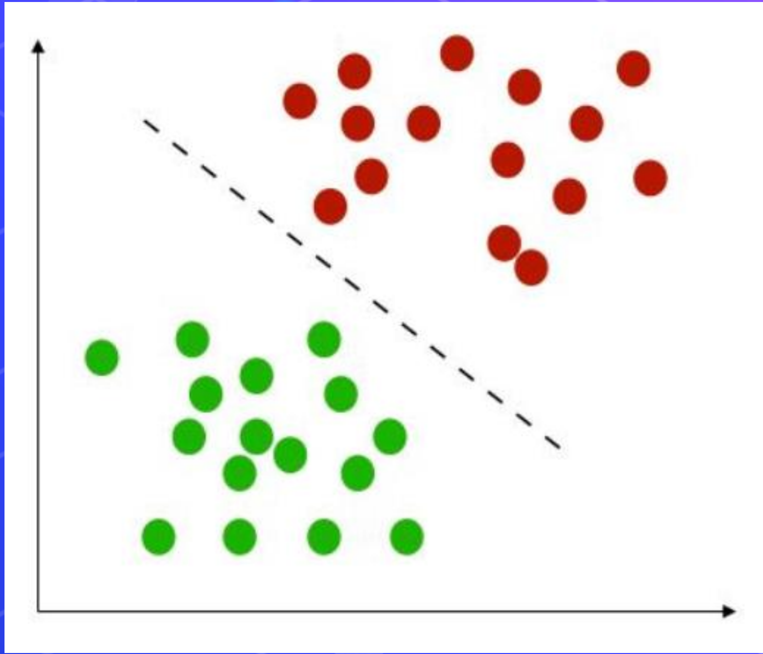


- Machine Learning
- Supervised Learning
  - Regression
    - Simple Linear Regression
    - Multiple Linear Regression
    - Polynomial Regression
    - Evaluating Model Performance
  - Classification
    - Logistic Regression
    - K-Nearest Neighbors (KNN)
    - Naive Bayes
    - SVM
    - Decision 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

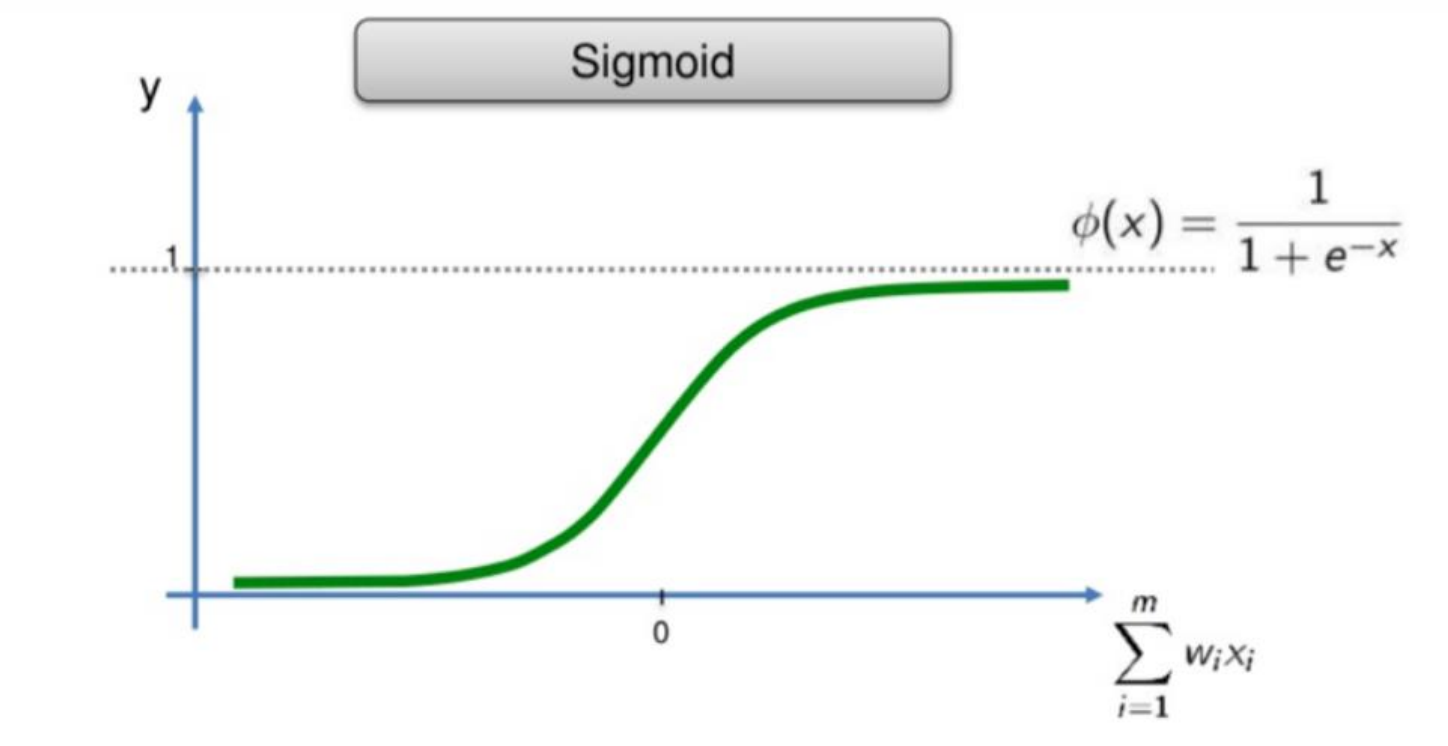
The Logistic Function

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

where

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

Logistic Regression



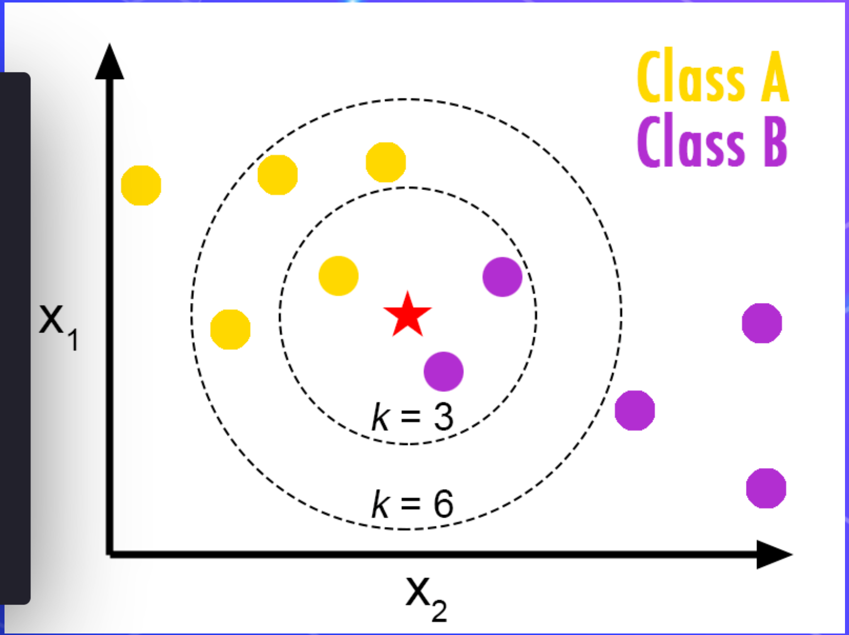
Logistic Regression

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
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

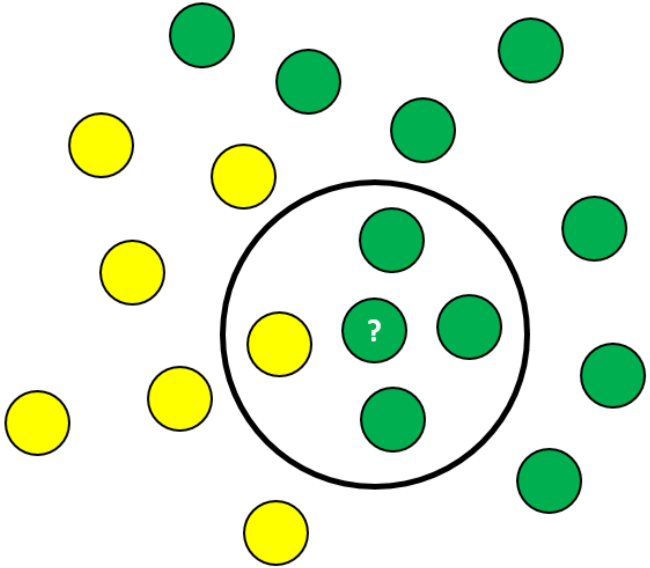
K-Nearest Neighbors (KNN)

```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = KNeighborsClassifier()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
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(yellow) = $\frac{7}{17}$

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

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

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

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

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

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

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

Naive Bayes

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix


model = MultinomialNB( )
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

print(confusion_matrix(y_test, y_pred))
print(accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Questions ?!

Thanks!

>_ Live long and prosper