



# Python Programming

Machine Learning with Scikit-Learn

Feedback is greatly appreciated!

# Objective

- What is machine learning
- Classification
- Scientific packages



# What is Machine Learning?

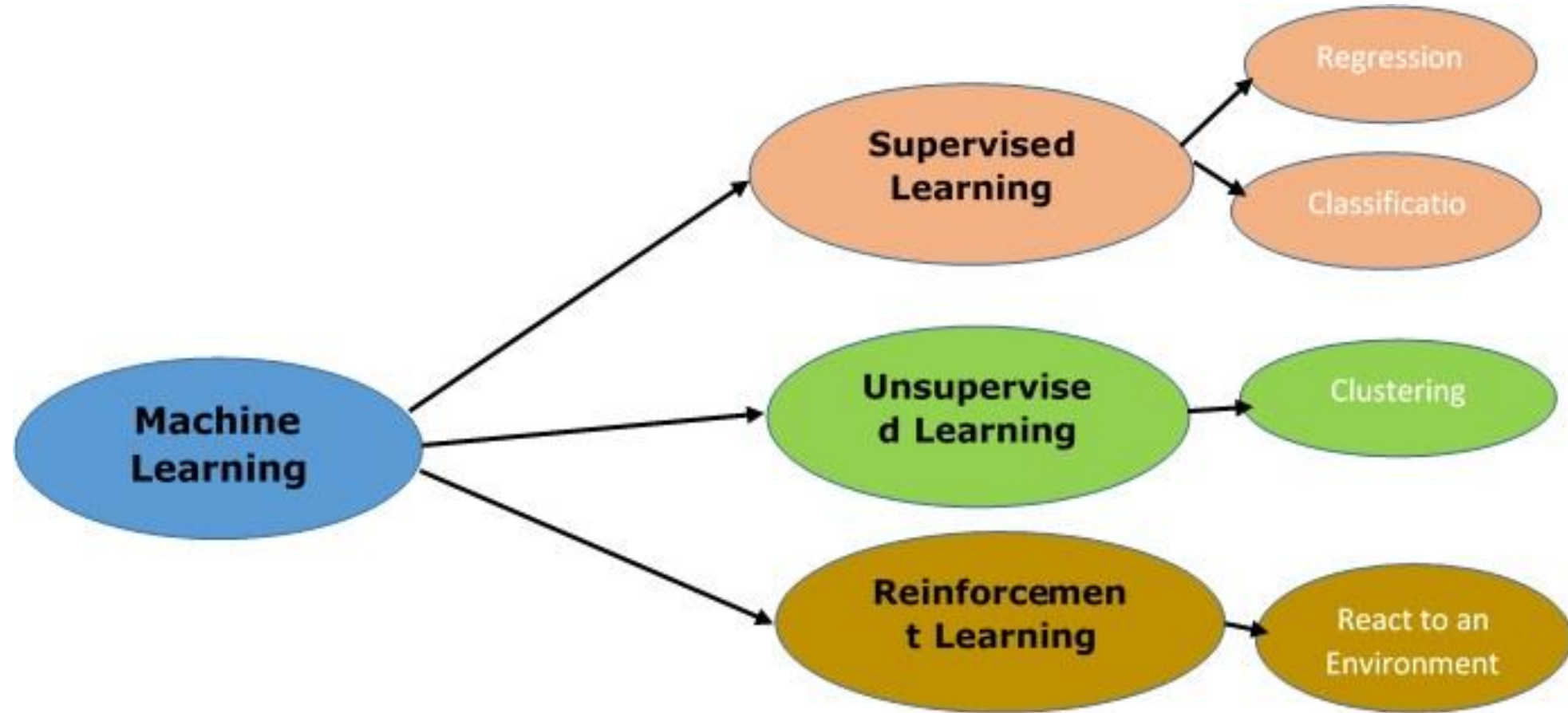
# Machine Learning

- Subfield of Artificial Intelligence (AI)
- Building blocks to make computers learn to behave more intelligently
- It is a theoretical concept. There are various techniques with various implementations

# What is it used for?

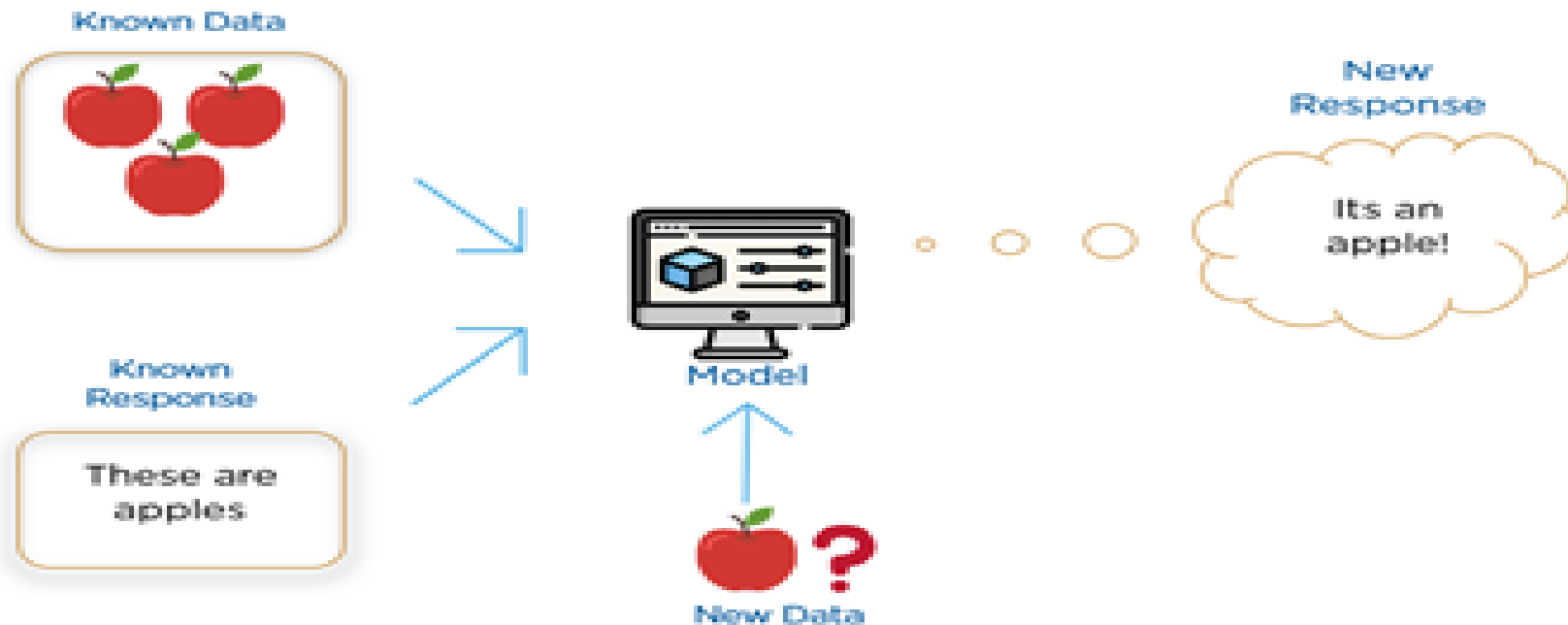
- Fraud detection.
- Web search results.
- Real-time ads on web pages
- Credit scoring and next-best offers.
- Prediction of equipment failures.
- New pricing models.
- Network intrusion detection.
- Recommendation Engines.
- Customer Segmentation.
- Text Sentiment Analysis.
- Predicting Customer Churn.
- Pattern and image recognition.
- Email spam filtering.
- Financial Modeling.

# Types of Algorithms



# Supervised Learning

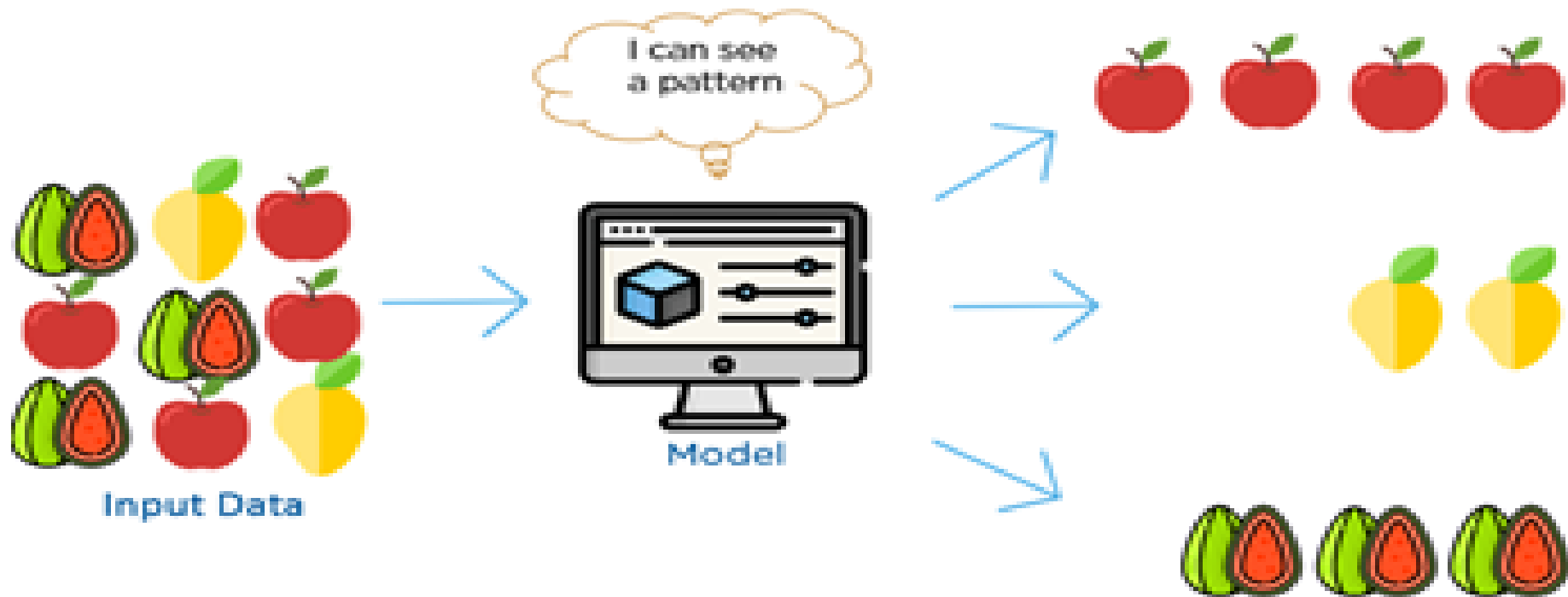
The correct classes of the training data are known





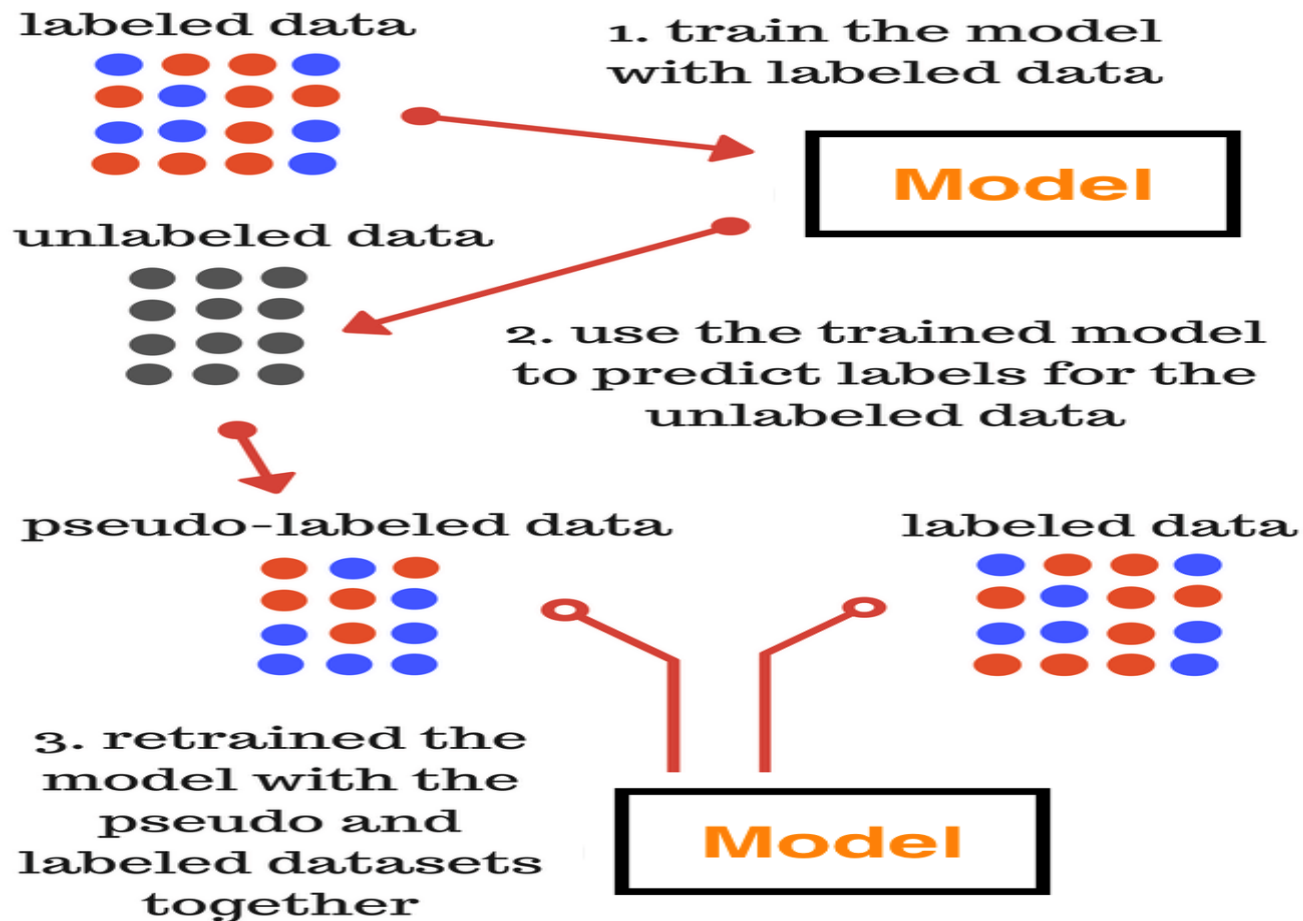
# Unsupervised Learning

The correct classes of the training data are not known



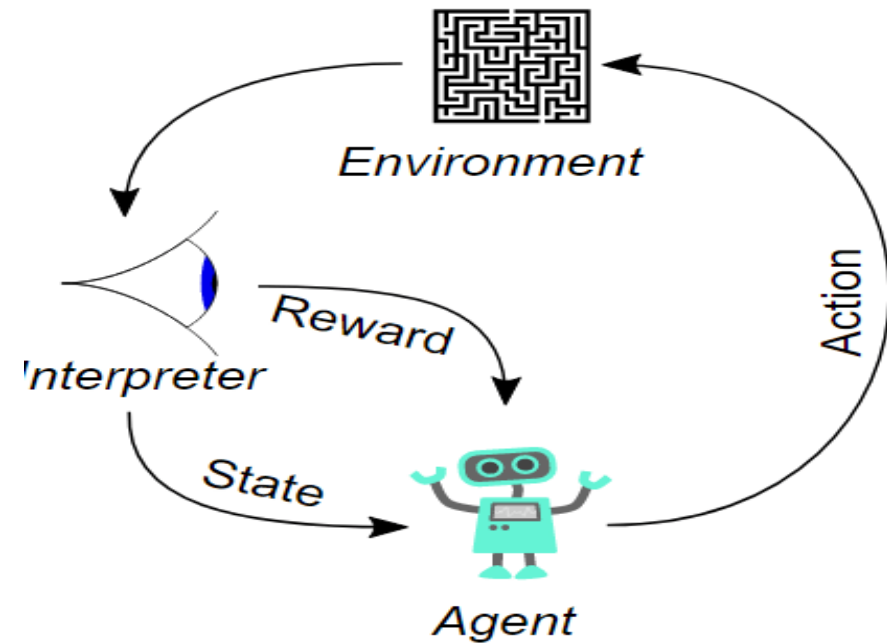
# Semi-Supervised Learning

A mix of Supervised and Unsupervised learning



# Reinforcement Learning

- Allows the machine or software agent to learn its behavior based on feedback from the environment.
- This behavior can be learnt once and for all, or keep on adapting as time goes by.



# Terminology

- Features
  - The number of features or distinct traits that can be used to describe each item in a quantitative manner.
- Samples
  - A sample is an item to process (e.g. classify). It can be a document, a picture, a sound, a video, a row in database or CSV file, or whatever you can describe with a fixed set of quantitative traits.
- Feature vector
  - an n-dimensional vector of numerical features that represent some object.
- Feature extraction
  - Preparation of feature vector .
  - Transforms the data in the high-dimensional space to a space of fewer dimensions.
- Training/Evolution set
  - Set of data to discover potentially predictive relationships.

# Sample dataset: play golf: Yes or No

Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

# Machine Learning Techniques

# Techniques

- *classification*: predict class from observations
- *clustering*: group observations into "meaningful" groups
- *regression (prediction)*: predict value from observations



# Classification



# Types of classification algorithms

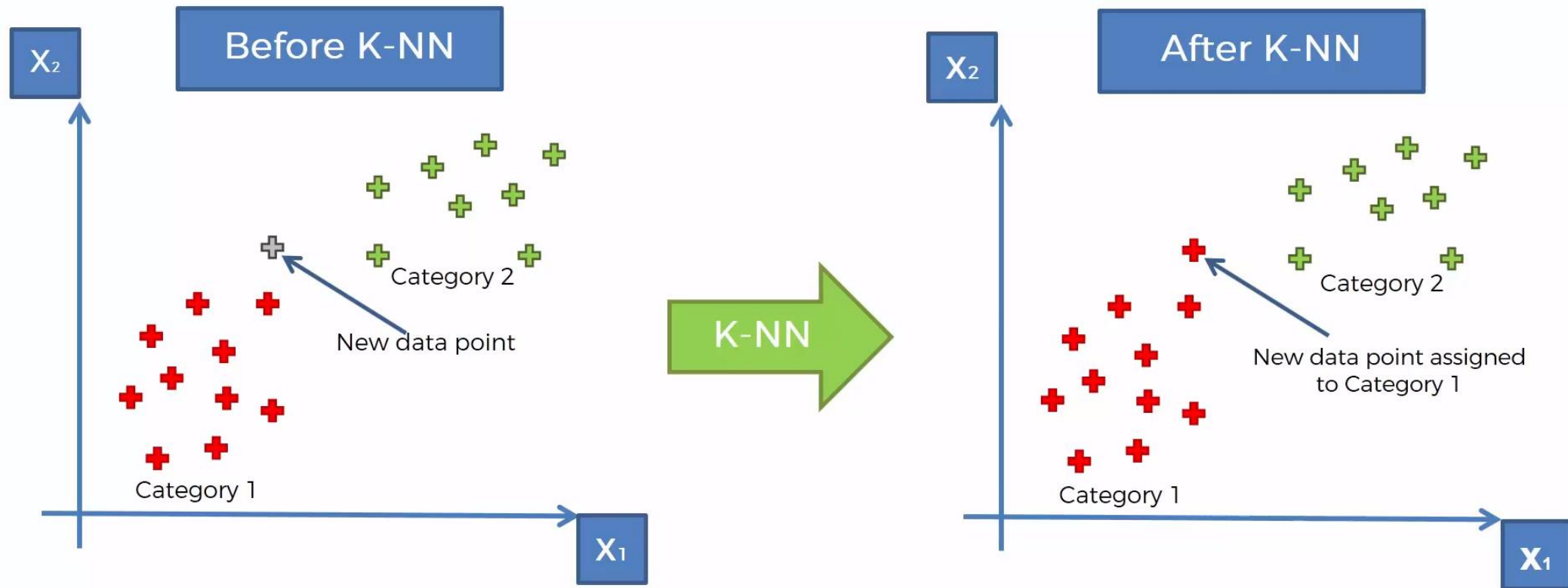
- Nearest Neighbor (k-nearest neighbors (kNN))
- Naive Bayes Classifier
- Support Vector Machines (SVM)
- Decision Trees
- Boosted Trees
- Random Forest
- Neural Networks

<https://medium.com/@sifium/machine-learning-types-of-classification-9497bd4f2e14>

<https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>

k-nearest neighbors (k-NN)

# k-nearest neighbors (k-NN)



# k-nearest neighbors (k-NN)

STEP 1: Choose the number K of neighbors



STEP 2: Take the K nearest neighbors of the new data point, according to the Euclidean distance



STEP 3: Among these K neighbors, count the number of data points in each category



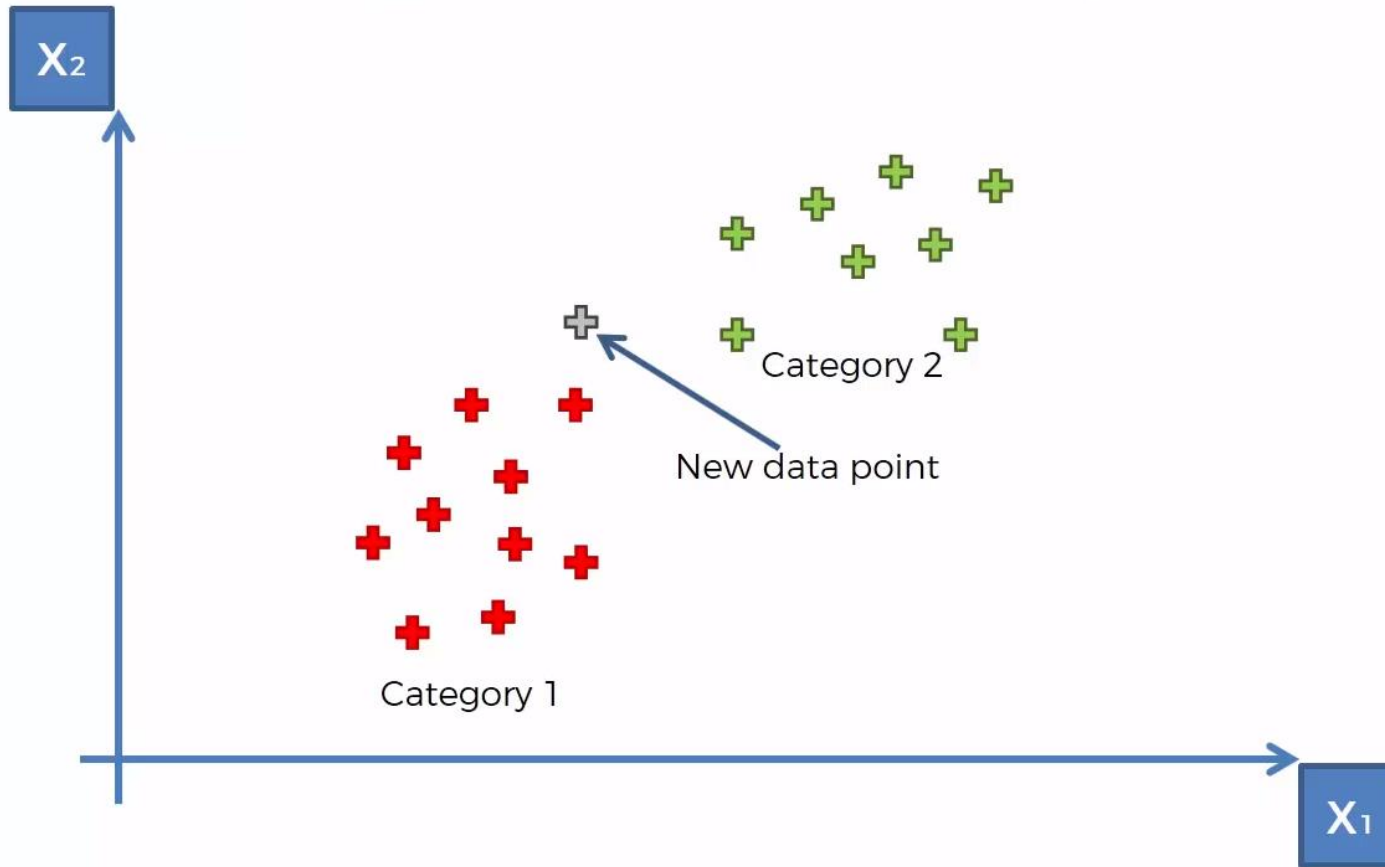
STEP 4: Assign the new data point to the category where you counted the most neighbors



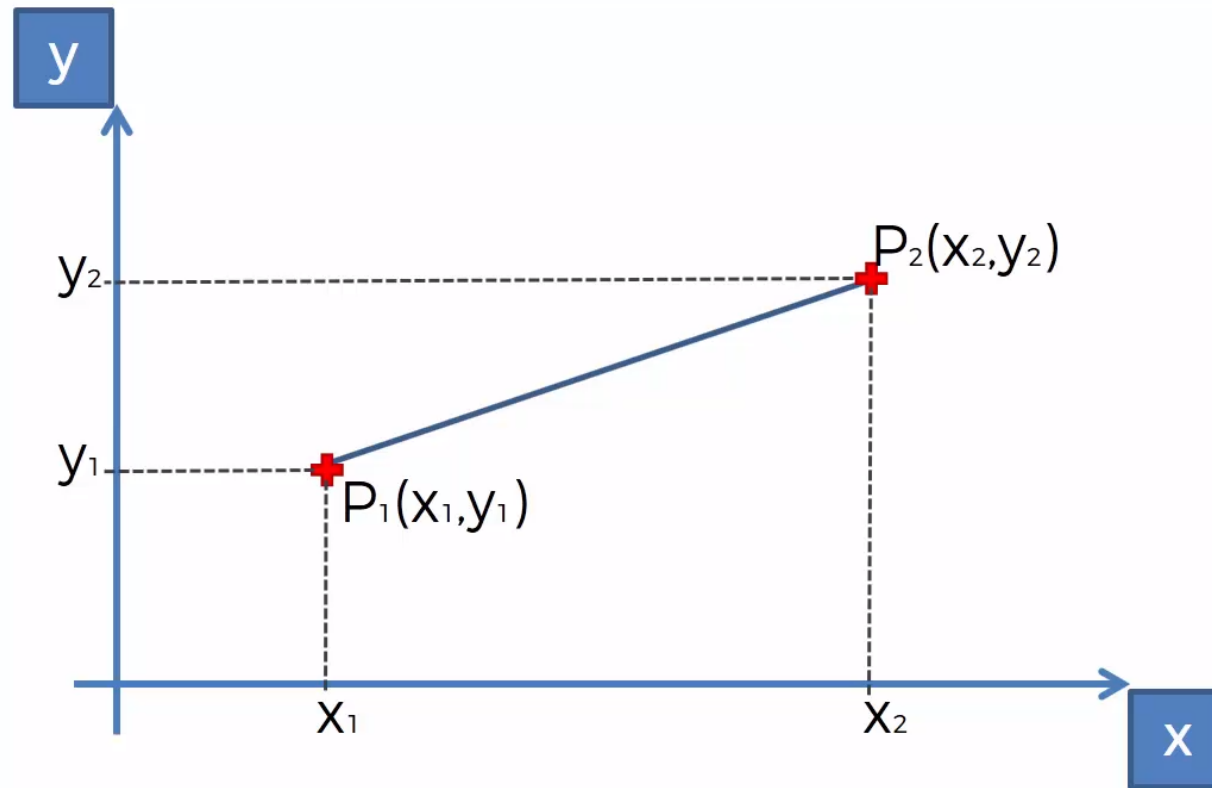
Your Model is Ready

# k-nearest neighbors (k-NN)

STEP 1: Choose the number K of neighbors:  $K = 5$



# k-nearest neighbors (k-NN)



$$\text{Euclidean Distance between } P_1 \text{ and } P_2 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

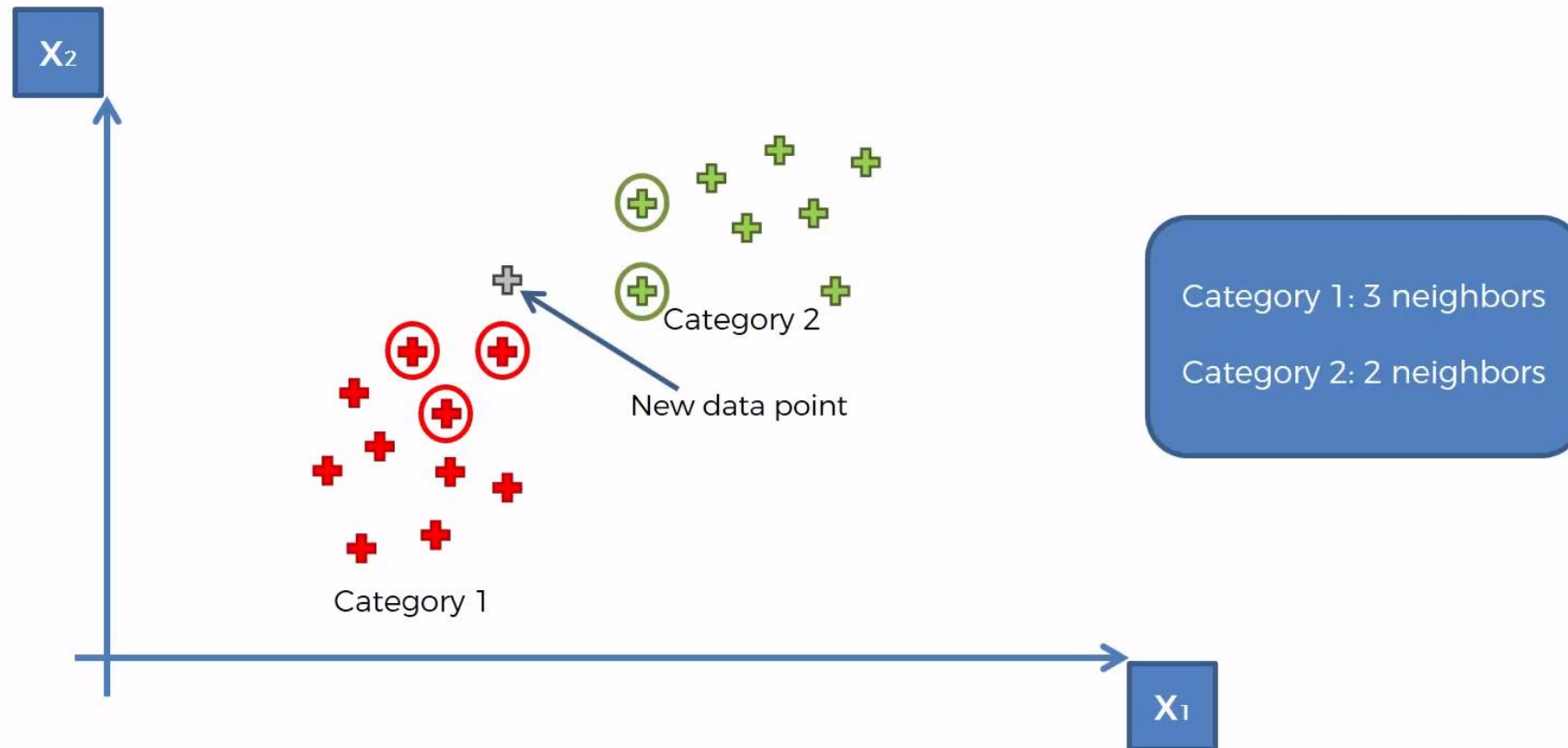
# k-nearest neighbors (k-NN)

STEP 2: Take the  $K = 5$  nearest neighbors of the new data point, according to the Euclidean distance



# k-nearest neighbors (k-NN)

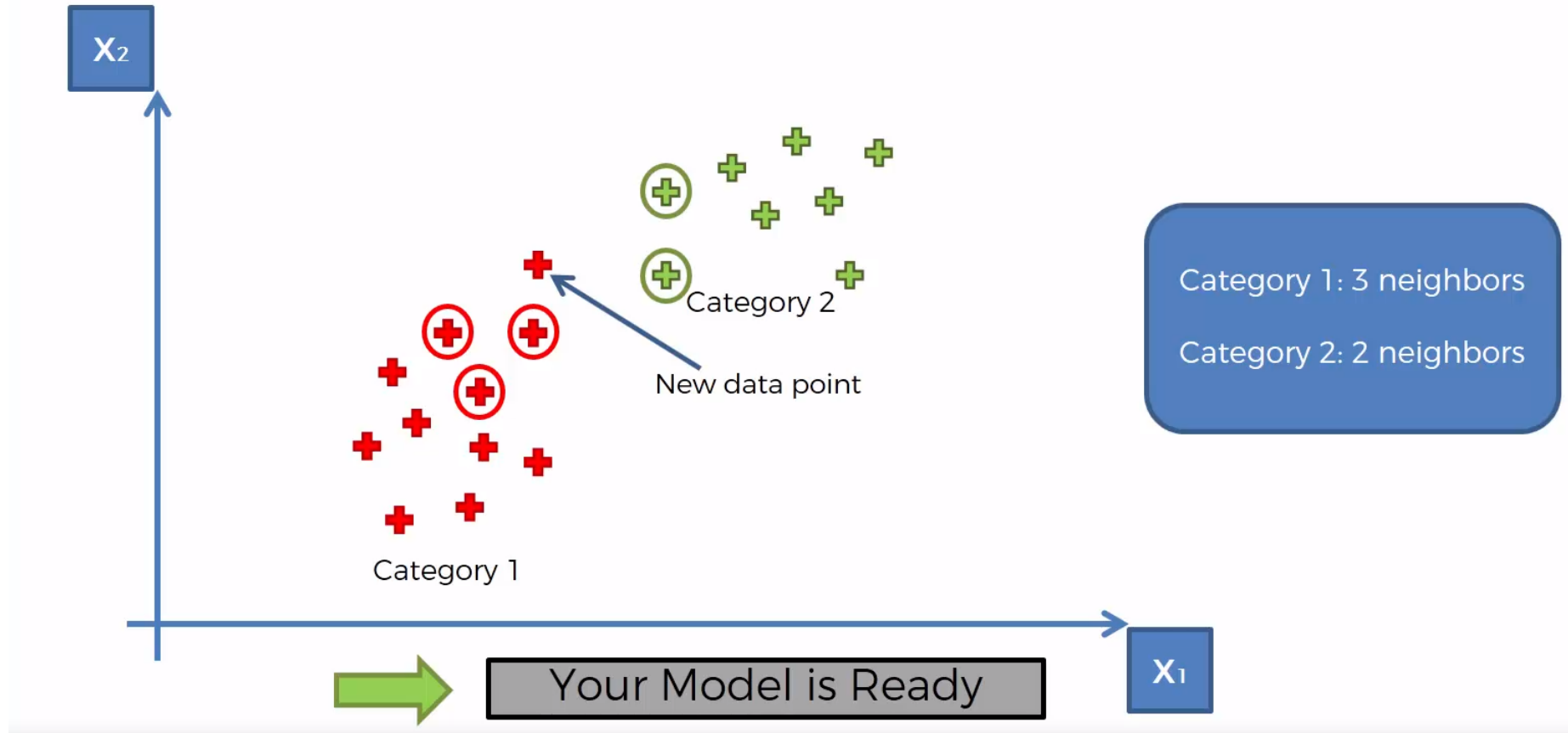
STEP 3: Among these K neighbors, count the number of data points in each category





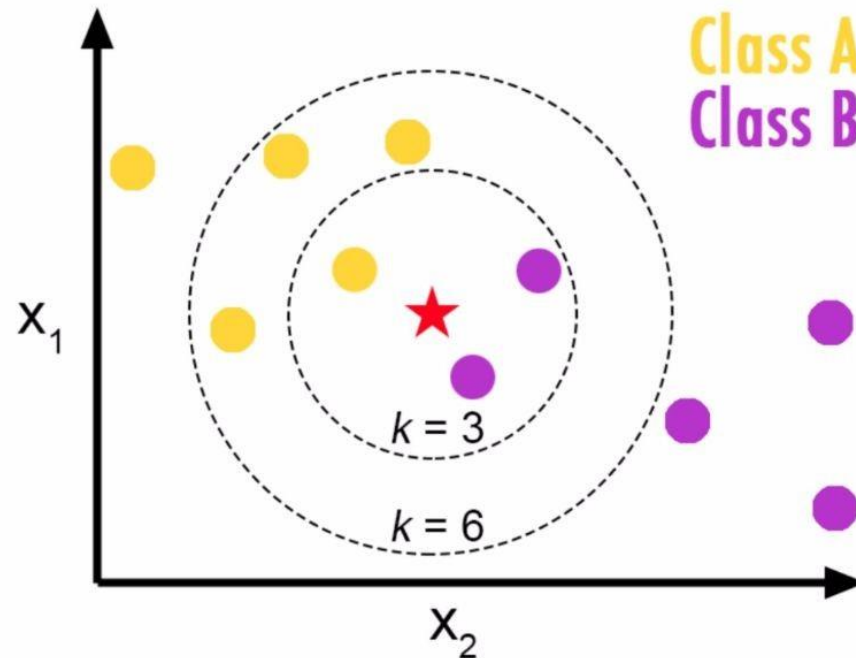
# k-nearest neighbors (k-NN)

STEP 4: Assign the new data point to the category where you counted the most neighbors



# k-nearest neighbors (k-NN)

Choosing a K will affect what class a new point is assigned to:



# Naive Bayes Classifier

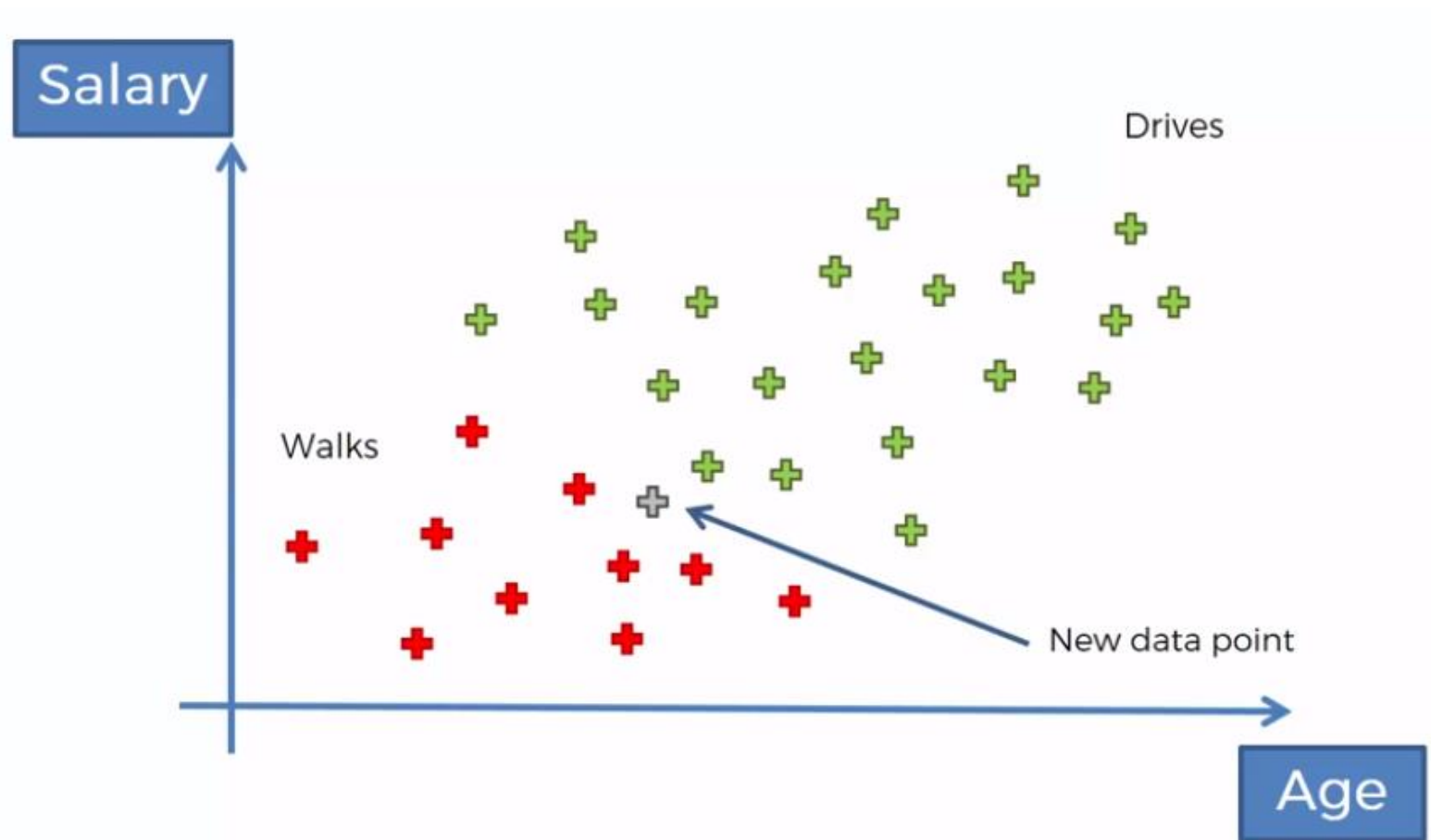
# Naive Bayes Classifier

- It is a classification technique based on **Bayes' Theorem**
- assumption of independence among predictors.
- In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature

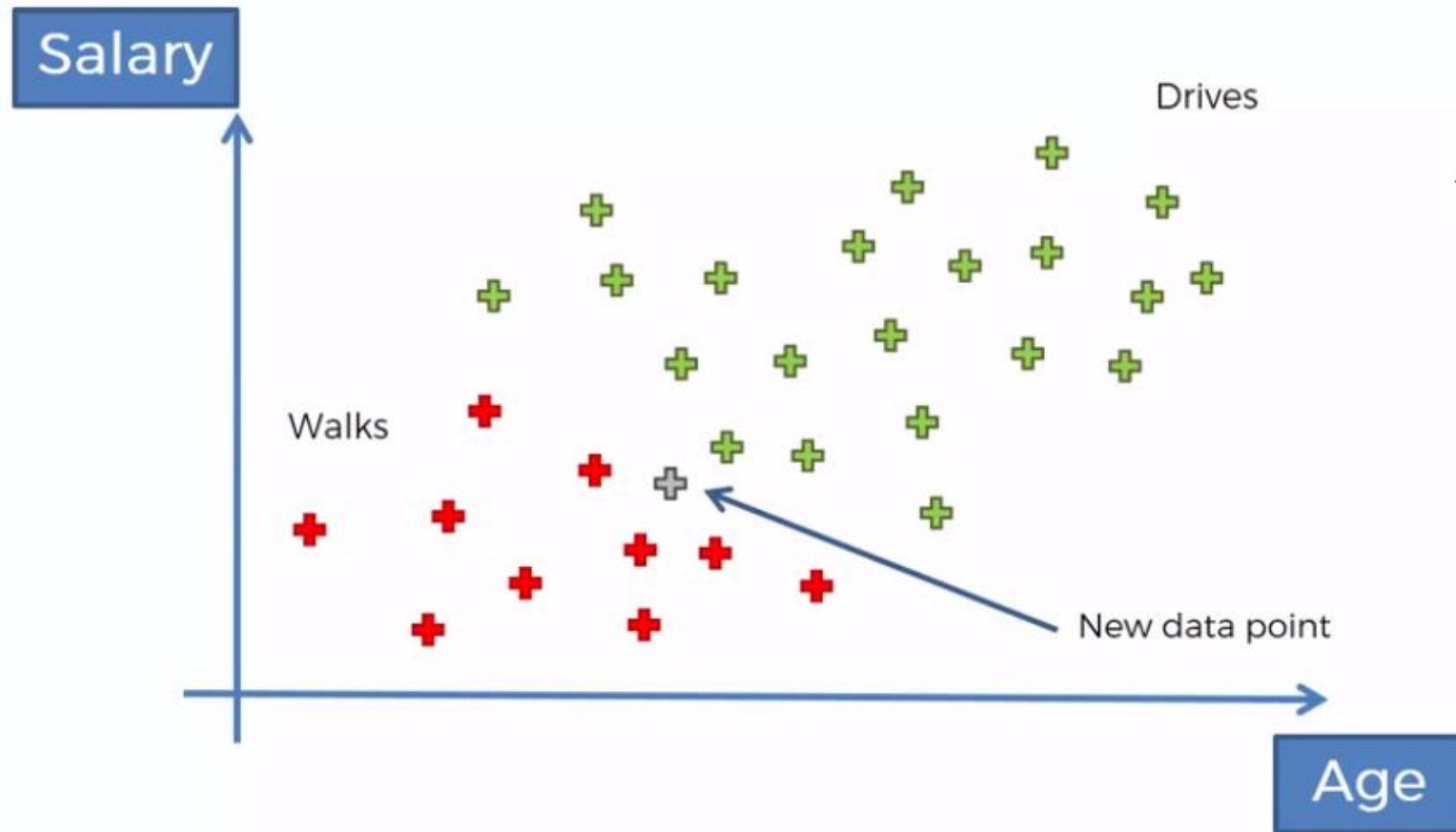
$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

*Bayes' Theorem*

# Naive Bayes Classifier



# Naive Bayes Classifier



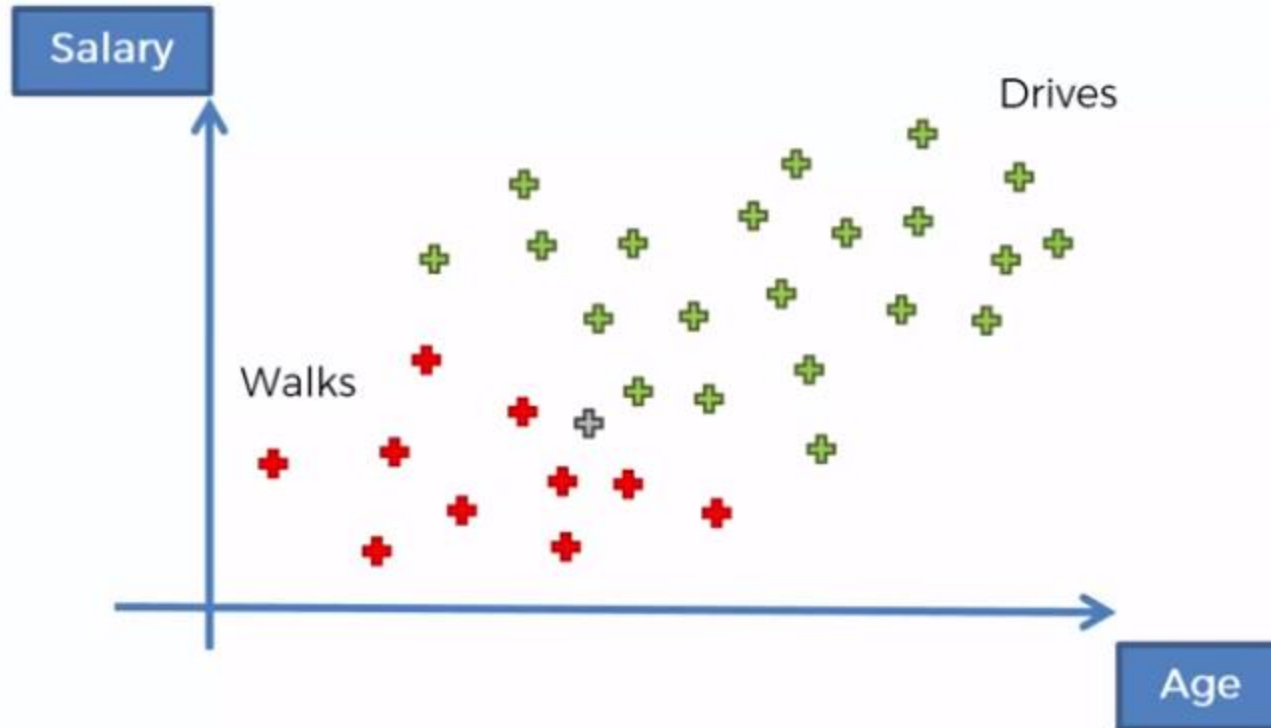
$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

$P(Walks|X)$  v.s.  $P(Drives|X)$

$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)}$$

$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)}$$

# Naive Bayes Classifier

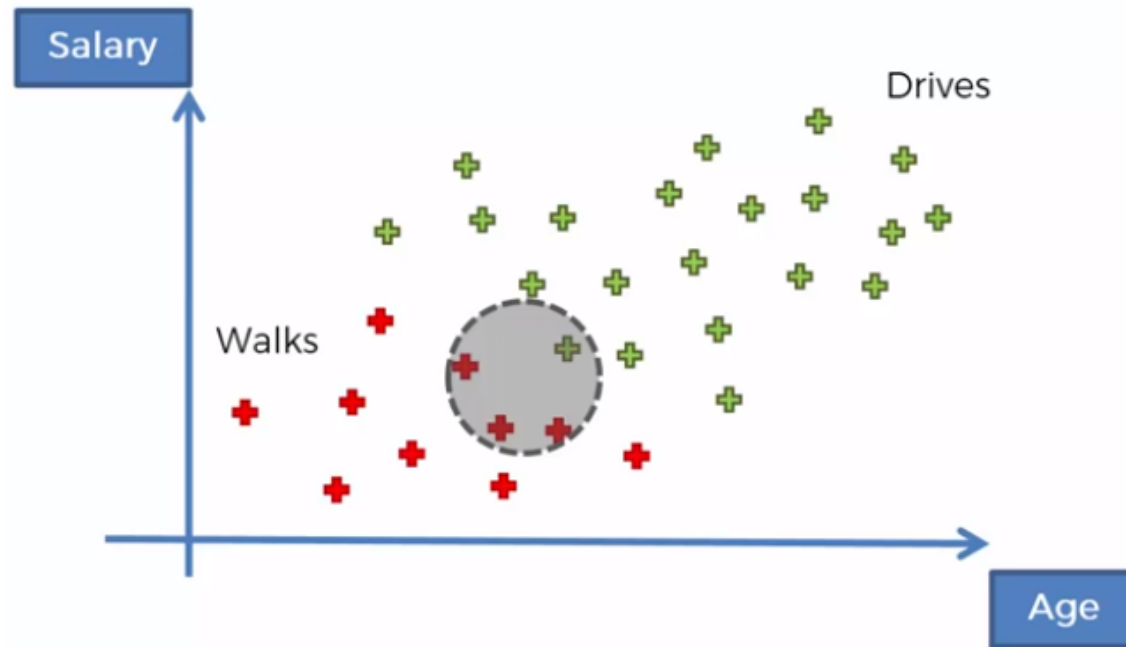


## #1. $P(\text{Walks})$

$$P(\text{Walks}) = \frac{\text{Number of Walkers}}{\text{Total Observations}}$$

$$P(\text{Walks}) = \frac{10}{30}$$

# Naive Bayes Classifier



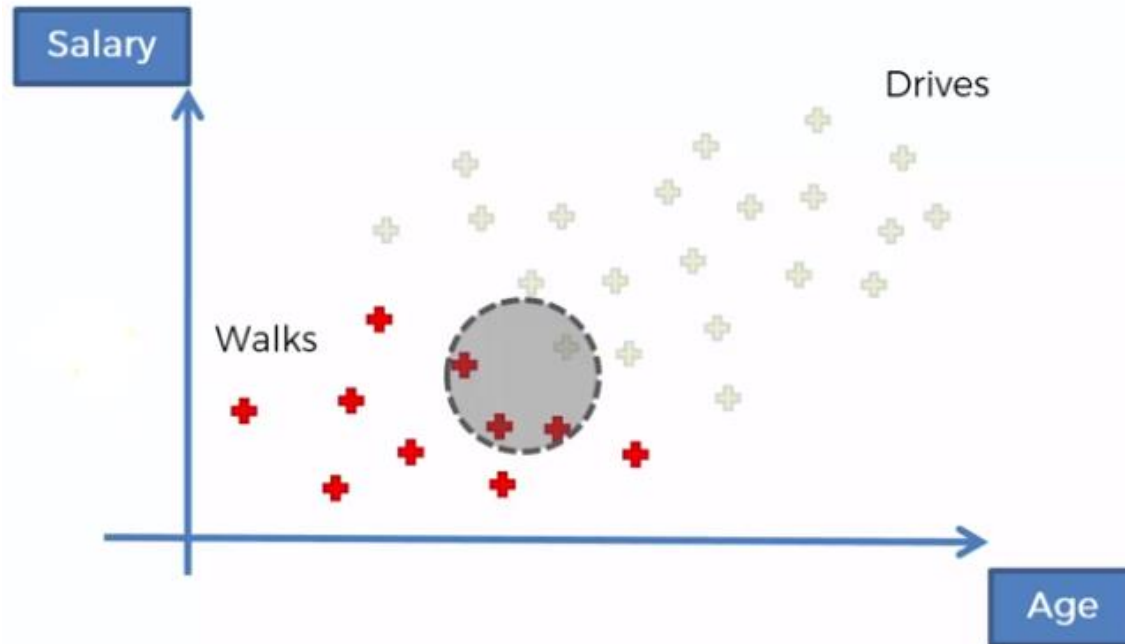
## #2. $P(X)$

$$P(X) = \frac{\text{Number of Similar Observations}}{\text{Total Observations}}$$

$$P(X) = \frac{4}{30}$$



# Naive Bayes Classifier



## #3. $P(X|Walks)$

*Number of Similar Observations*

$$P(X|Walks) = \frac{\text{Among those who Walk}}{\text{Total number of Walkers}}$$
$$P(X|Walks) = \frac{3}{10}$$

# Naive Bayes Classifier

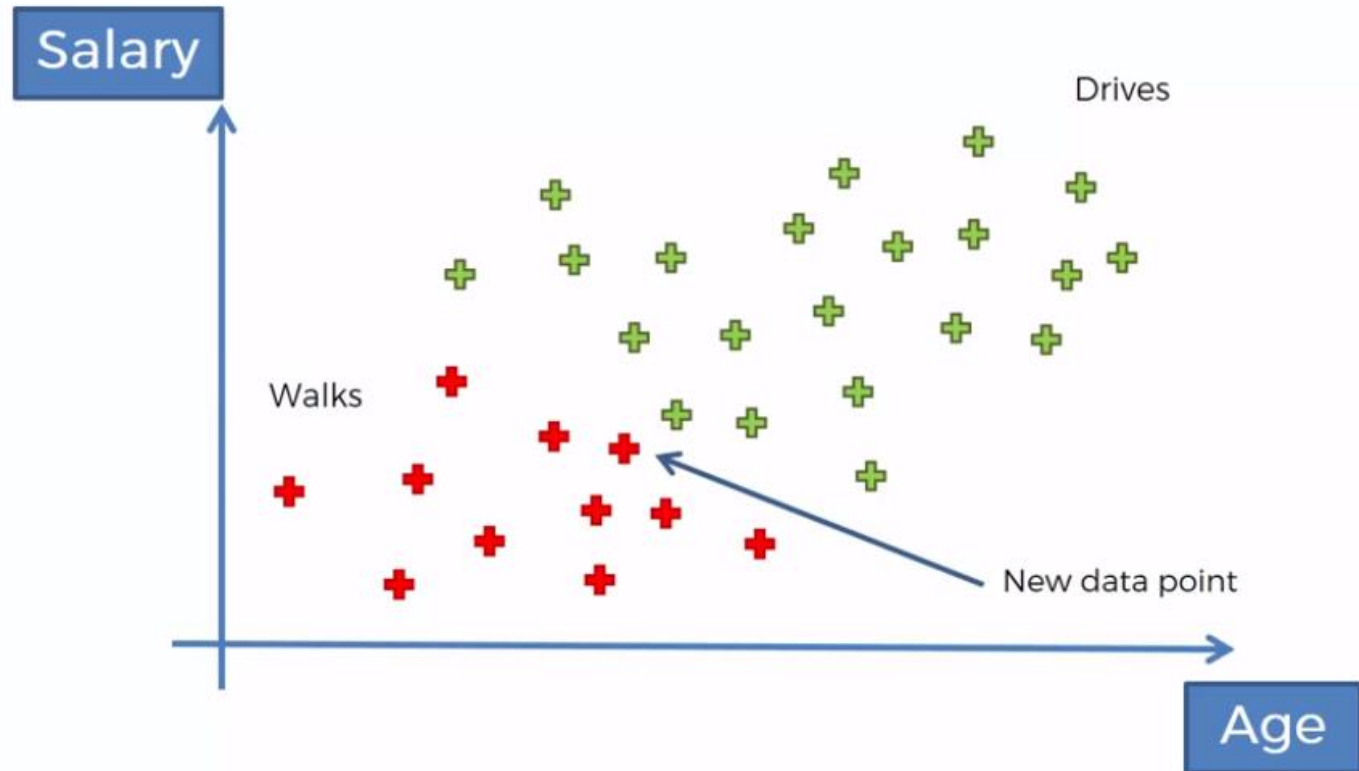
$$P(Walks|X) = \frac{P(X|Walks) * P(Walks)}{P(X)} \implies P(Walks|X) = \frac{\frac{3}{10} * \frac{10}{30}}{\frac{4}{30}} = 0.75$$

$$P(Drives|X) = \frac{P(X|Drives) * P(Drives)}{P(X)} \implies P(Drives|X) = \frac{\frac{1}{20} * \frac{20}{30}}{\frac{4}{30}} = 0.25$$

$$P(Walks|X) \text{ v.s. } P(Drives|X) \implies 0.75 \text{ v.s. } 0.25$$

$$0.75 > 0.25 \implies \boxed{P(Walks|X) > P(Drives|X)}$$

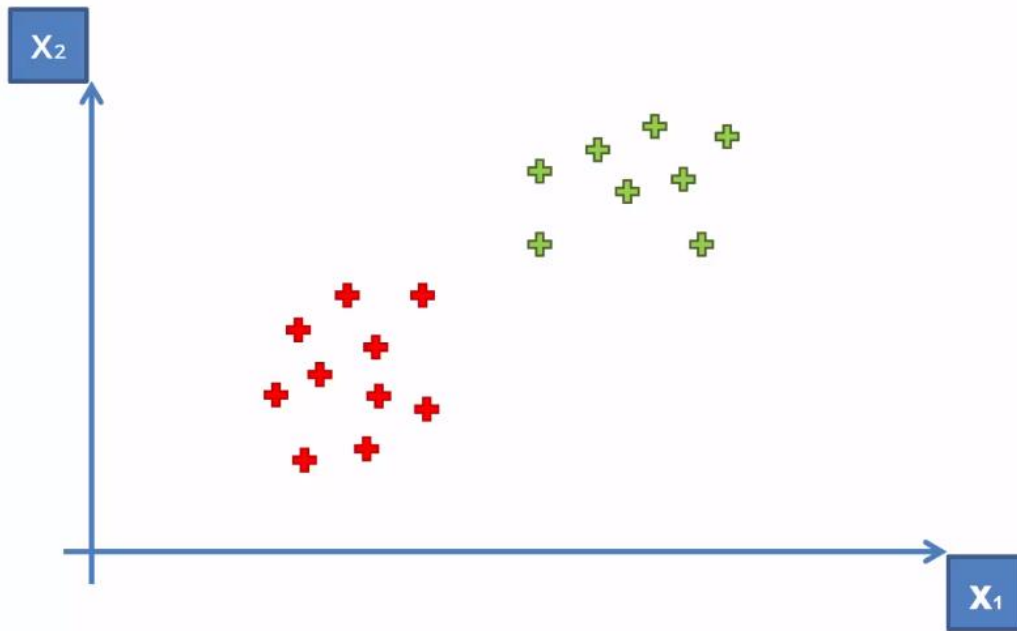
# Naive Bayes Classifier



# Support Vector Machines (SVM)

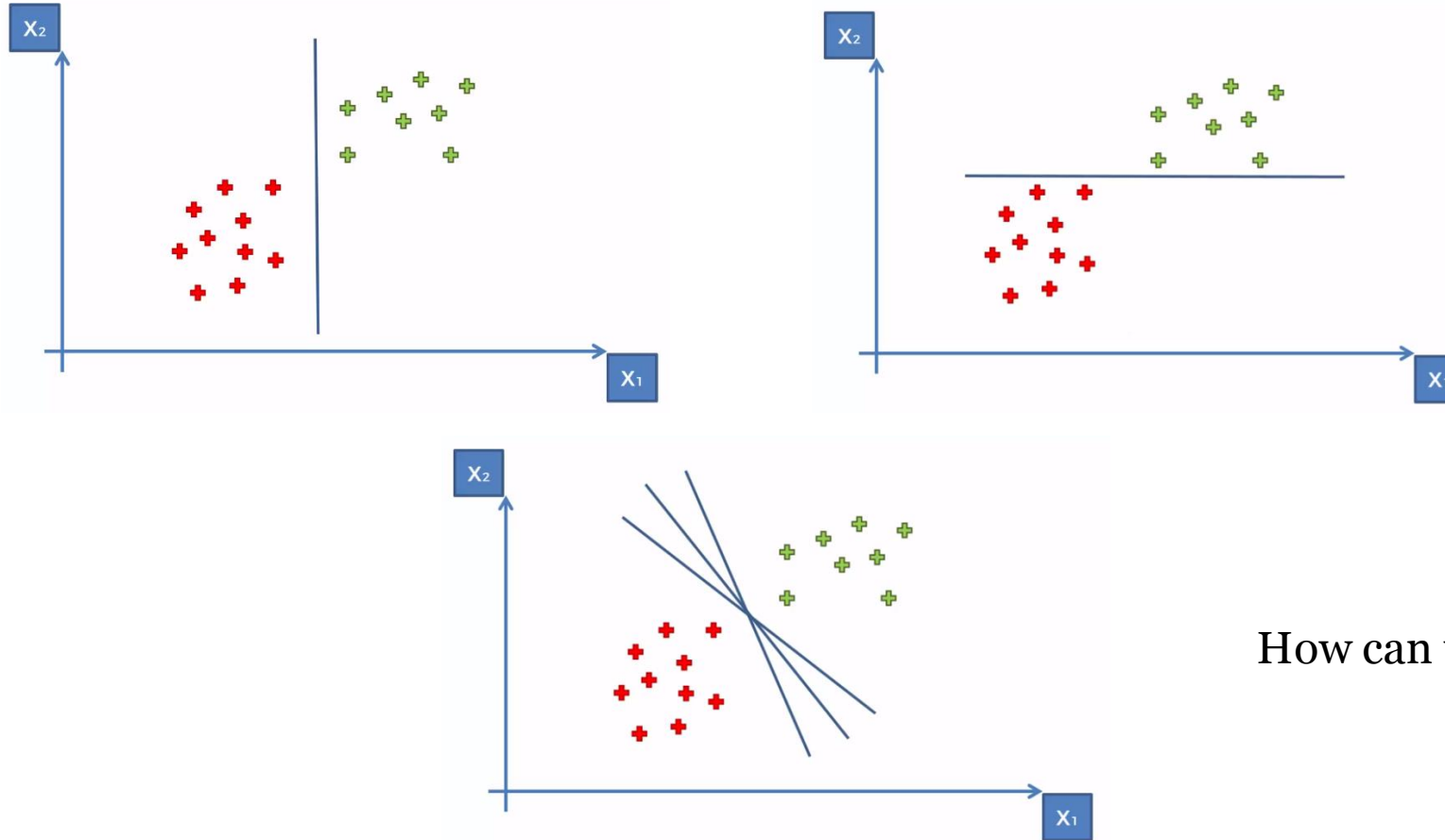
# Support Vector Machines (SVM)

Purpose of the classifier is how we can create a boundary between two classes



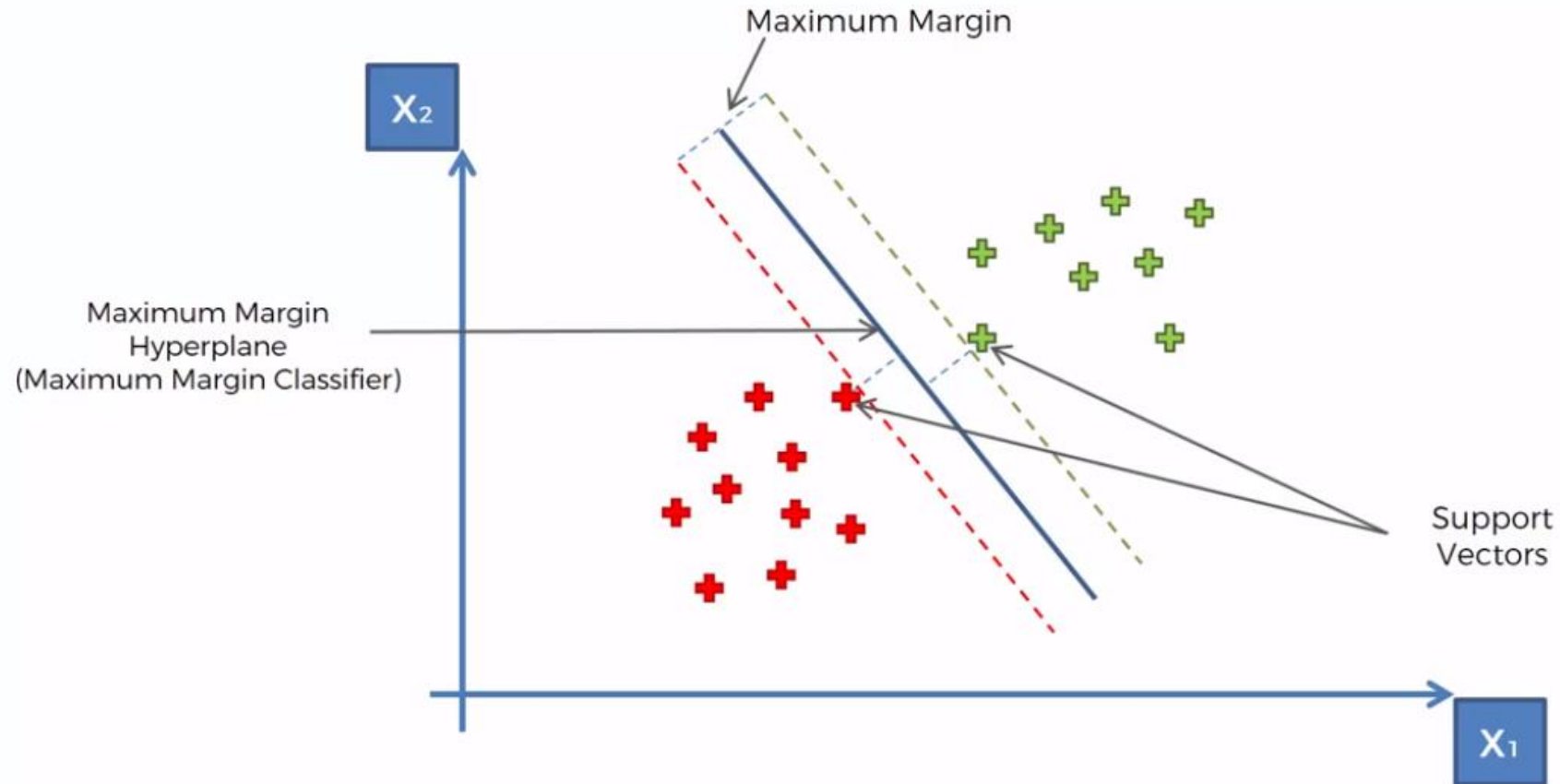
How can we separate these points?

# Support Vector Machines (SVM)



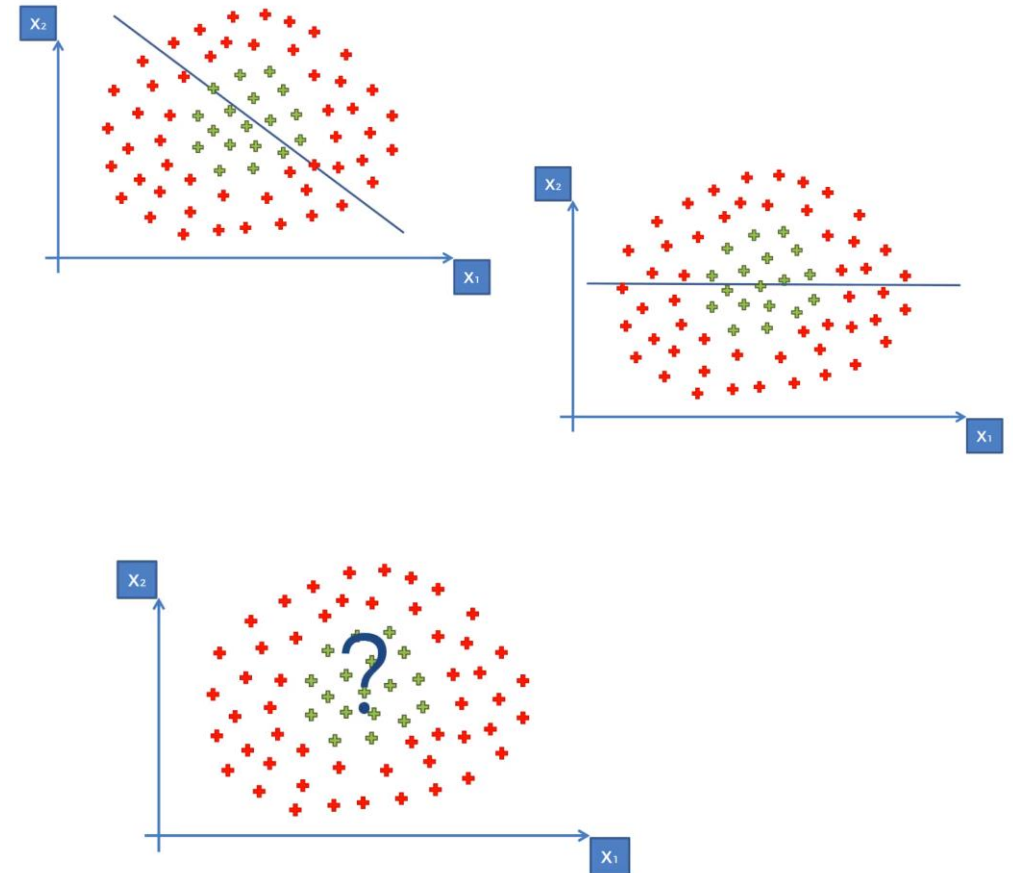
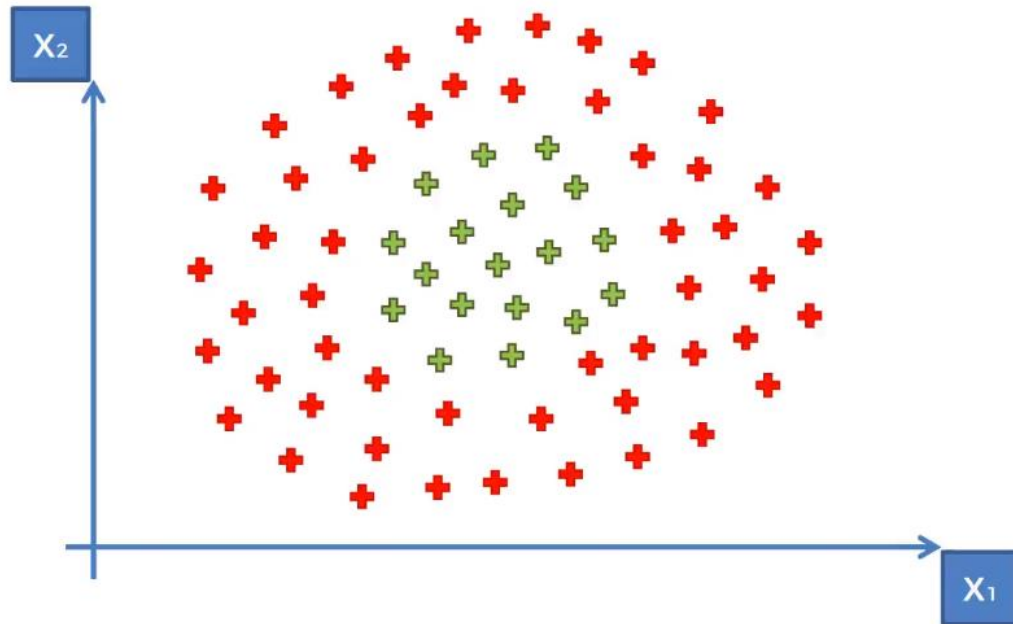
How can we separate these points?

# Support Vector Machines (SVM)



# Kernel SVM

What about these points?



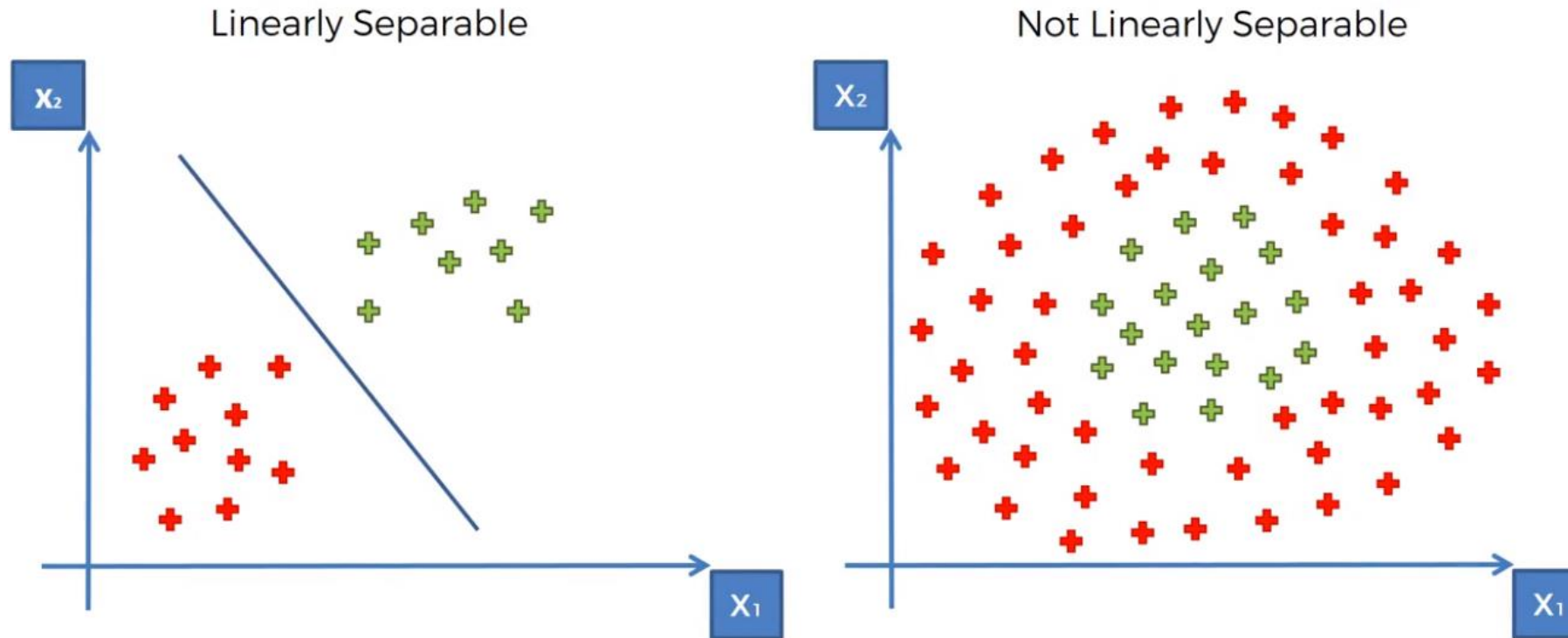
<https://www.udemy.com/machinelearning/learn/v4/content>

<https://medium.com/@sifium/machine-learning-types-of-classification-9497bd4f2e14>



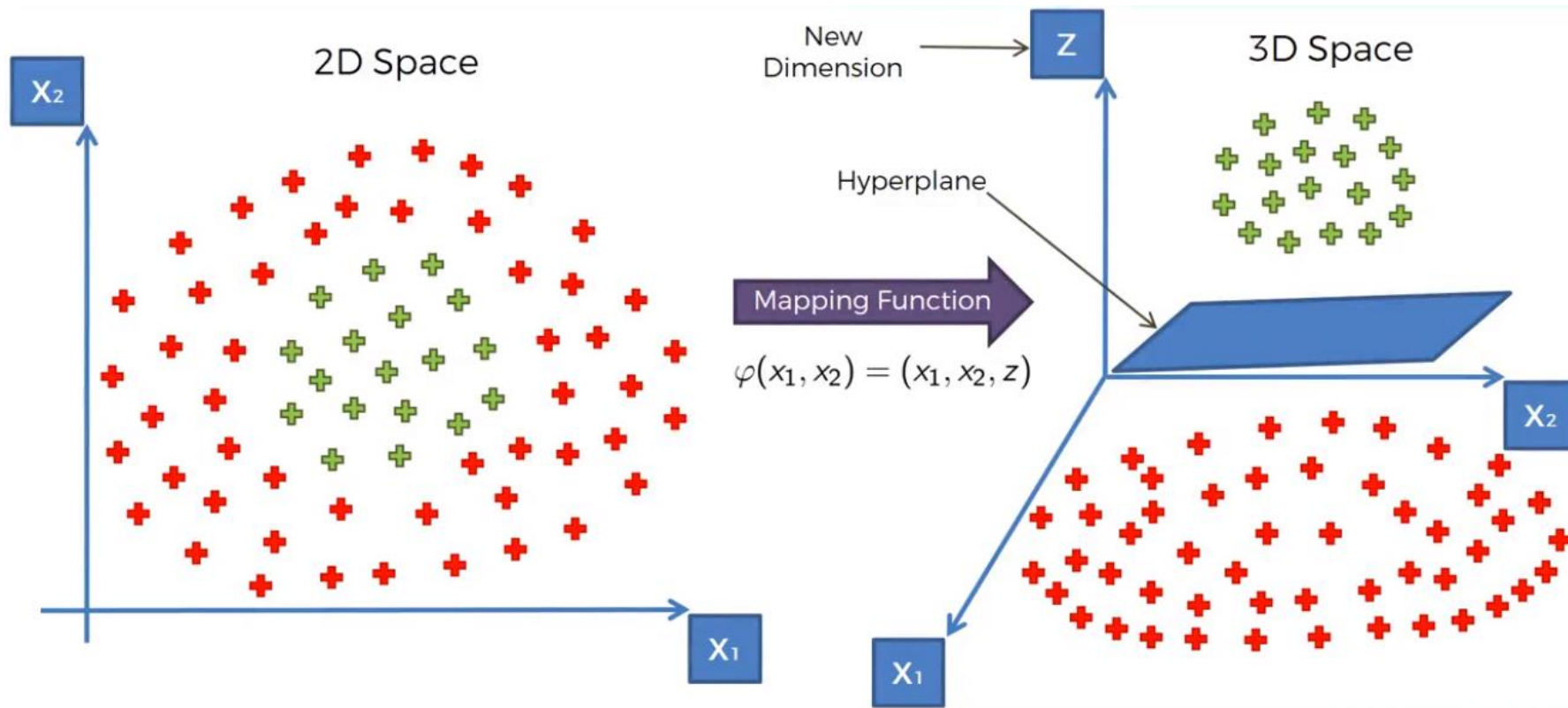
# Kernel SVM

## Linear Separability



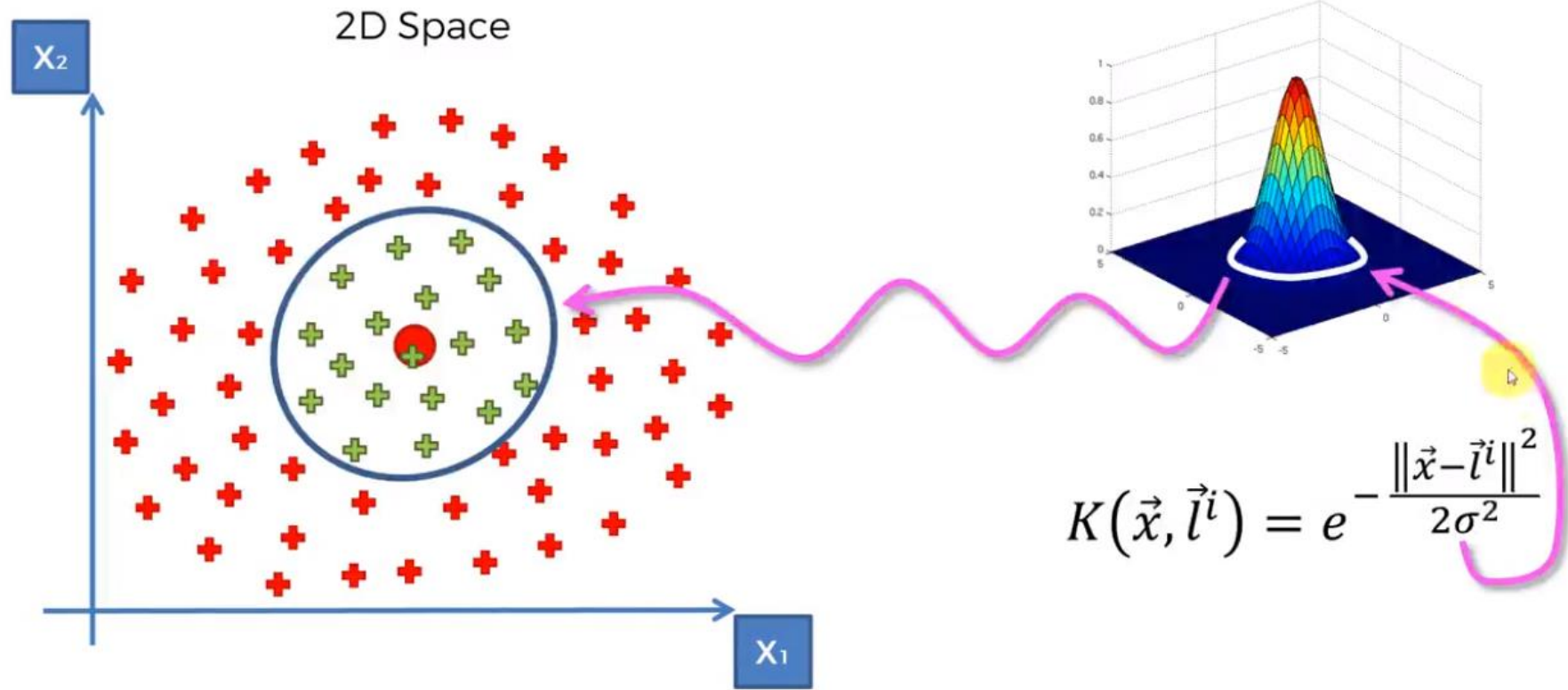
# Kernel SVM

## Mapping to higher dimension



# Kernel SVM

## The Gaussian RBF Kernel





How to know our model is good

When we have an input  $x$  and we apply a function  $f$  on the input  $x$  to predict an output  $y$ . Difference between the actual output and predicted output is the error. Our goal with machine learning algorithm is to generate a model which minimizes the error of the test dataset.

# Models are assessed based on the **Mean Squared Error** on a new test dataset

$$L(x, y) = \sum_{i=1}^N (y - f(x))$$

Error = sum of all (Actual output – Predicted Output)

# Classification

- False Positives & False Negatives
- Confusion Matrix
- Accuracy
- Precision
- Recall
- F1 score



# False Positives & False Negatives

False Positive



False Negative



		Reality	
		True	False
Measured or Perceived	True	Correct 😊	<b>Type 1 error</b> False Positive
	False	<b>Type 2 error</b> False Negative	Correct 😊

- A **false positive** is where you receive a positive result for a test, when you should have received a negative results. A false positive is usually called a **Type I error**.
- A **false negative** is where you receive a negative test result, but you should have got a positive test result . A false negative is usually called a **Type II error**.



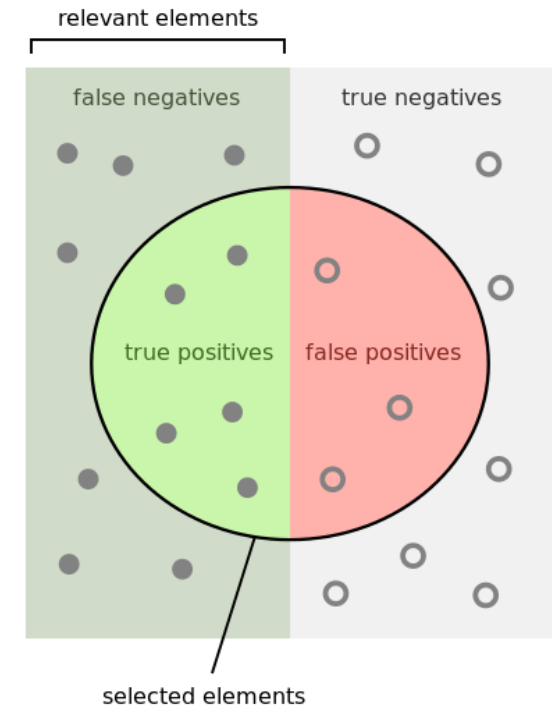
# Metrics

accuracy =  
 $\text{true positives} + \text{true negatives} / \text{total}$

precision =  
 $\text{true positives} / (\text{true positives} + \text{false positives})$

recall =  
 $\text{true positives} / (\text{false negatives} + \text{true positives})$

F1 score =  
 $2 * ((\text{precision} * \text{recall}) / (\text{precision} + \text{recall}))$



How many selected items are relevant?

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

How many relevant items are selected?

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

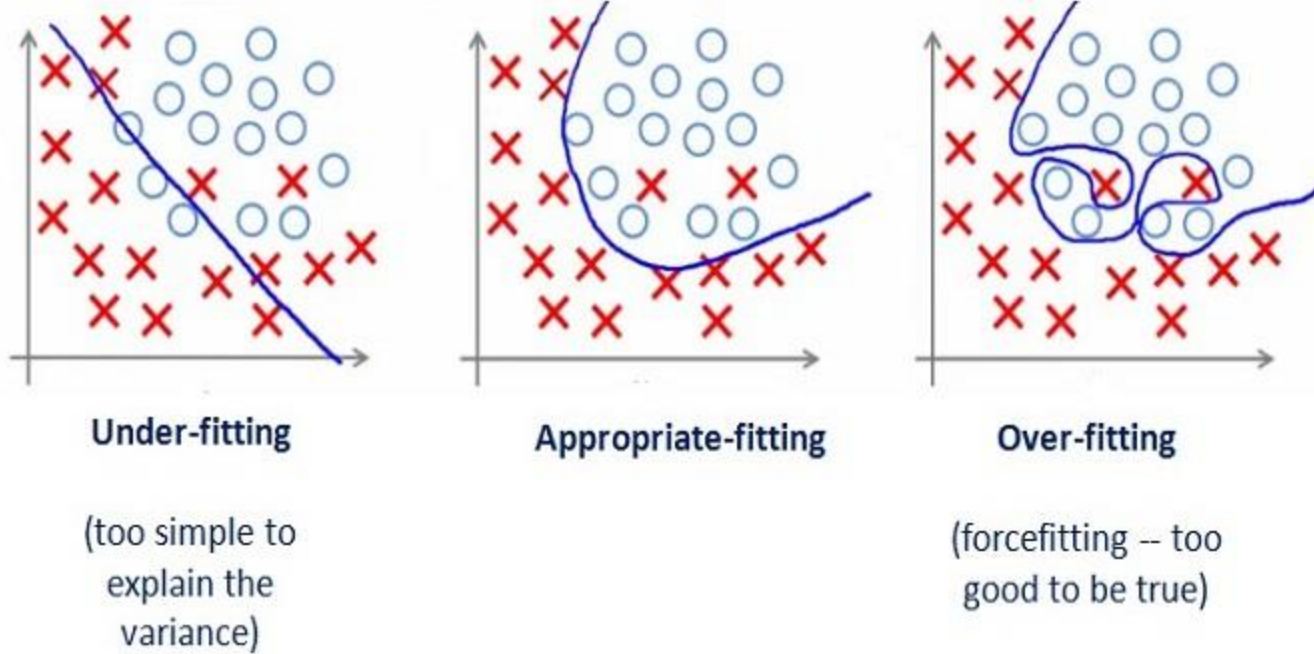
# Bias Error

Bias is how far are the predicted values from the actual values. If the **average predicted values are far off from the actual values then the bias is high**


# Variance Error

- Variance is the amount that the estimate of the target function will change if different training data was used.
- Variance measures how far a data set is spread out. The technical definition is “The average of the squared differences from the mean”
  - The data set 12, 12, 12, 12, 12 has a var. of zero (the numbers are identical).
  - The data set 12, 12, 12, 12, 13 has a var. of 0.167; a small change in the numbers equals a very small var.
  - The data set 12, 12, 12, 12, 13,013 has a var. of 28171000; a large change in the numbers equals a very large number.

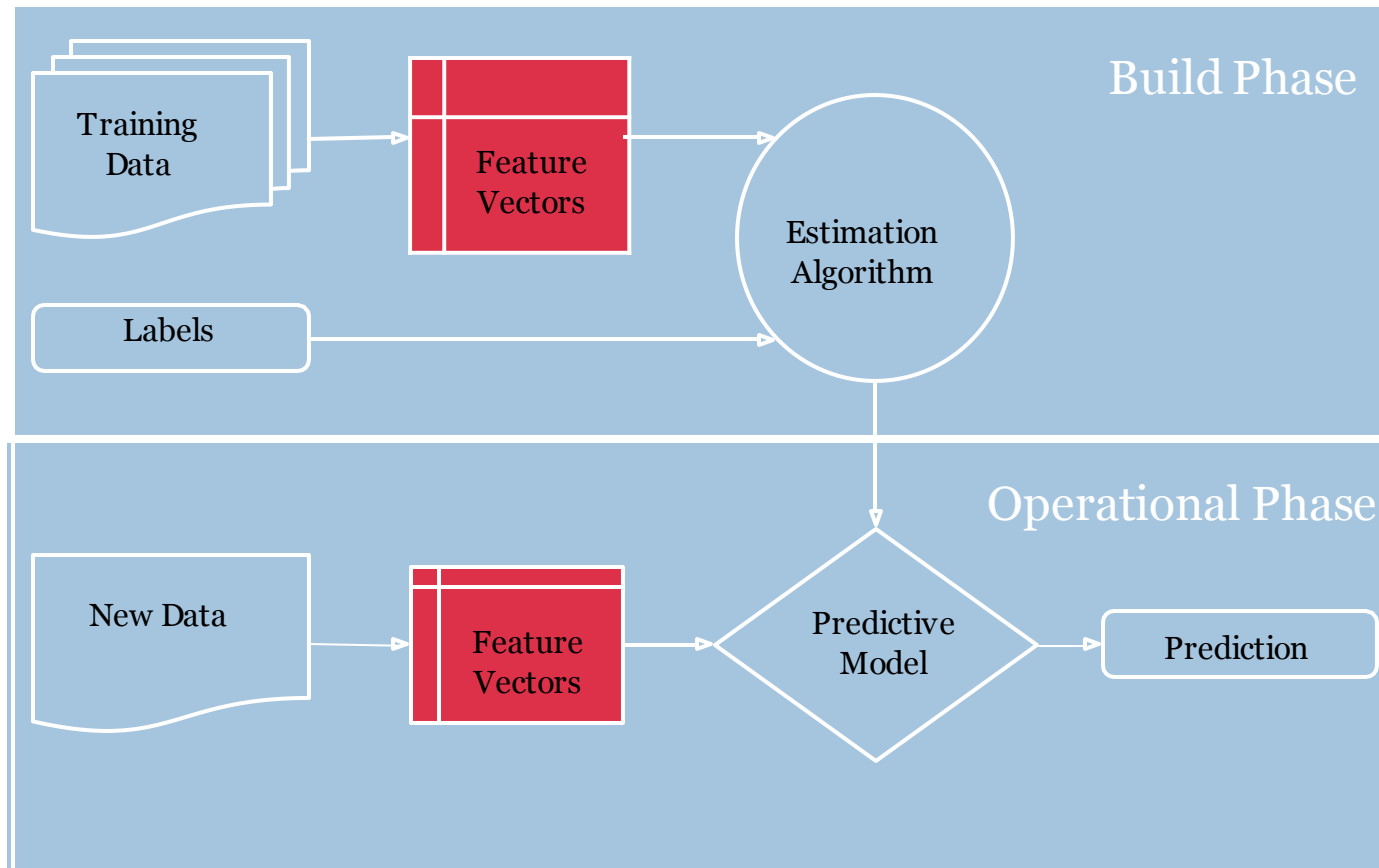
# Underfitting and Overfitting



- Underfitting occurs if the model or algorithm shows low variance but high bias
- Overfitting occurs if the model or algorithm shows low bias but high variance.



# An architecture for operationalizing Machine Learning algorithms



Architecture of Machine Learning Operations

# Your Task

Given a data set of instances of size  $N$ , create a model that is fit from the data (built) by extracting features and dimensions. Then use that model to predict outcomes ...

- Data Wrangling (normalization, standardization, imputing)
- Feature Analysis/Extraction
- Model Selection/Building
- Model Evaluation

# Evaluation Procedure: Train/Test split

- Split the dataset into two sets
- Train the model on the Training set
- Test the model on the Testing Set, and evaluate how well we did



# Titanic Survival Challenge

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Title	FamilySize	IsAlone
0	892	3	0	34	0	0	7.8292	Q	1	1	1
1	893	3	1	47	1	0	7.0000	S	3	2	0
2	894	2	0	62	0	0	9.6875	Q	1	1	1
3	895	3	0	27	0	0	8.6625	S	1	1	1
4	896	3	1	22	1	1	12.2875	S	3	3	0

**Objective:** Predict who survived the sinking of the Titanic

**-Is there a specific outcome for the objective?** Yes, survived or not survived

**-Is it a Numerical outcome or Class outcome?** Class (survived or not survived)

**-What type of machine learning is this:** Supervised classification

**-Suitable models for this type of learning** KNN, SVM (see cheat sheet)

**-Do we have the data or is data scraping needed:** Already have data

**-Load data into Pandas using** `pd.read_csv('')`

# Import the libraries

# Getting to know your data

```
train_df = pd.read_csv('../input/train.csv')  
test_df = pd.read_csv('../input/test.csv')  
combine = [train_df, test_df]
```

# EDA

How many classes are there in the target variable?

```
print(train_df['Survived'].value_counts(dropna=False))
```

# Analyze by describing data

Which features are available in the dataset?

Which features are categorical?

Which features are numerical?

Which features are mixed data types?

Which features may contain errors or typos?

Which features contain blank, null or empty values?

What are the data types for various features?

# Analyze by describing data

```
train_df.info()  
test_df.info()
```

RangeIndex: 891 entries, 0 to 890 Data columns (total 12 columns):  
PassengerId 891 non-null int64  
Survived 891 non-null int64  
Pclass 891 non-null int64  
Name 891 non-null object  
Sex 891 non-null object  
Age 714 non-null float64  
SibSp 891 non-null int64  
Parch 891 non-null int64  
Ticket 891 non-null object  
Fare 891 non-null float64  
Cabin 204 non-null object  
Embarked 889 non-null object  
dtypes: float64(2), int64(5), object(5)

# What is the distribution of numerical feature values across the samples?

Total samples are 891 or 40% of the actual number of passengers on board the Titanic (2,224).

Survived is a categorical feature with 0 or 1 values.

Around 38% samples survived representative of the actual survival rate at 32%.

Most passengers ( $> 75\%$ ) did not travel with parents or children.

Nearly 30% of the passengers had siblings and/or spouse aboard.

Fares varied significantly with few passengers ( $<1\%$ ) paying as high as \$512.

Few elderly passengers ( $<1\%$ ) within age range 65-80.



# Fixing null values

```
print(train_df.isnull().sum())
```

What will be your decision after seeing this data

PassengerId	0
Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687
Embarked	2

# Result of null in Embarked Column

```
train_df["Embarked"] = train_df["Embarked"].fillna("S")
```

```
train_df.drop("Cabin", axis=1, inplace=True)
```

S	644
C	168
Q	77
NaN	2

# Analyze by pivoting features

- we can quickly analyze our feature correlations by pivoting features against each other
- We can only do so at this stage for features which do not have any empty values.

It also makes sense doing so only for features which are categorical (Sex), ordinal (Pclass) or discrete (SibSp, Parch) type

**Pclass:** We observe significant correlation ( $>0.5$ ) among Pclass=1 and Survived

We decide to include this feature in our model.

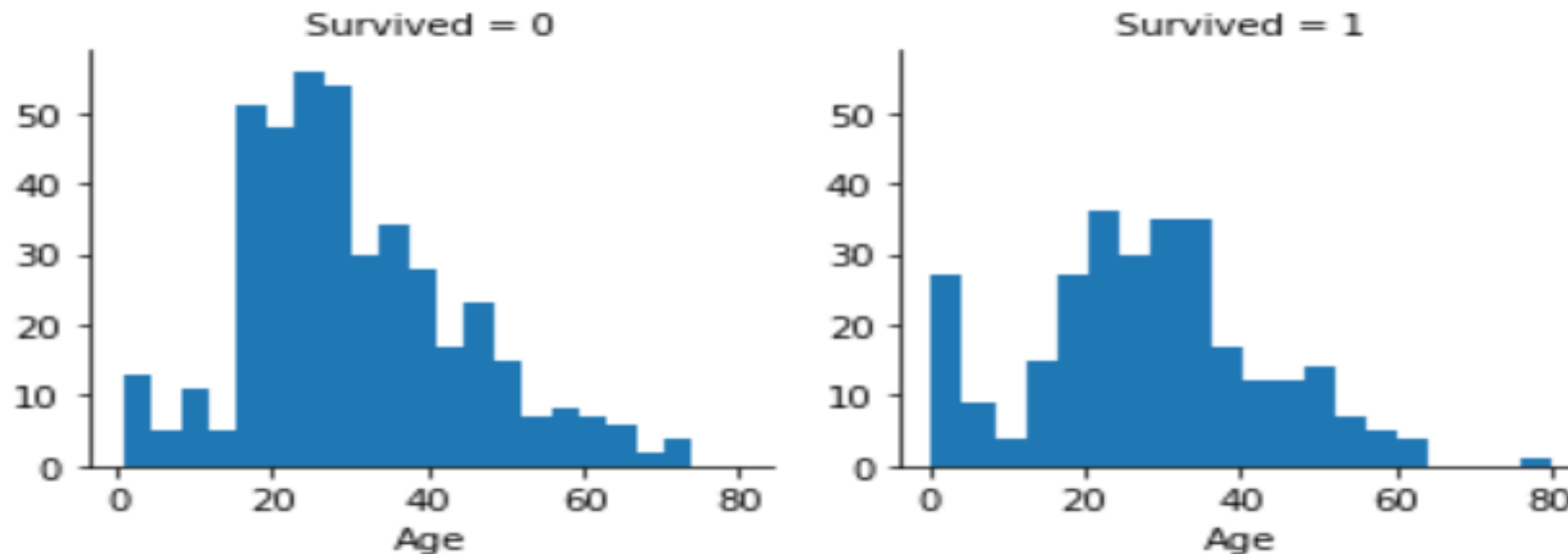
```
train_df[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean()  
.sort_values(by='Survived', ascending=False)
```

Pclass	Survived	
0	1	0.629630
1	2	0.472826
2	3	0.242363

# Analyze by visualizing data

A histogram chart is useful for analyzing **continuous numerical variables like Age** where banding or ranges will help identify useful patterns.

```
g = sns.FacetGrid(train_df, col='Survived')  
g.map(plt.hist, 'Age', bins=20)
```



# Analyze by visualizing data

## Observations:

- Infants (Age  $\leq 4$ ) had high survival rate.
- Oldest passengers (Age = 80) survived.
- Large number of 15-25 year olds did not survive.
- Most passengers are in 15-35 age range.

## Decisions:

- This simple analysis confirms our assumptions as decisions for subsequent workflow stages.
- We should **consider Age** in our model training.
- Complete the **Age feature for null values**.
- We should band age groups.

# Correcting by dropping features

- By dropping features we are dealing with fewer data points.
- Speeds up our notebook and eases the analysis.
- Based on our assumptions and decisions we want to drop the Cabin and Ticket features.
- Note that where applicable we perform operations on both training and testing datasets together to stay consistent.

```
train_df = train_df.drop(['Ticket', 'Cabin'], axis=1)
test_df = test_df.drop(['Ticket', 'Cabin'], axis=1)
combine = [train_df, test_df]
```

# Converting a categorical feature

Let us start by converting Sex feature to a new feature called Gender where female=1 and male=0

```
for dataset in combine: dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(int)
```

# Reading data

```
train_df = pd.read_csv('./train_preprocessed.csv')
test_df = pd.read_csv('./test_preprocessed.csv')
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test = test_df.drop("PassengerId", axis=1).copy()
print(train_df[train_df.isnull().any(axis=1)])
```



# Train the model and evaluation (SVM)

```
svc = SVC()  
svc.fit(X_train, Y_train)  
Y_pred = svc.predict(X_test)  
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)  
print("svm accuracy is:", acc_svc)
```

# Train the model and evaluation (KNN)

*##KNN*

```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
print("KNN accuracy is:", acc_knn)
```

# References

- <https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>