Basic Machine Learning with Scikit-Learn Course v11

SINGAPORE WORKFORCE SKILLS QUALIFICATIONS

Trainer: Marcel

TINFOTECH

Website:www.tertiarycourses.com.sg
Email: enquiry@tertiaryinfotech.com

About the Trainer

- Marcel graduated with majors in Applied Mathematics and Physics from the National University of Singapore.
- His core specialisation skills are R, Python, Machine Learning, Statistical Analysis, and Data Visualisation in Tableau. His current interests include Machine Learning, Deep Learning, Artificial Intelligence, Internet of Things, Robotics and Programming.

Let's Know Each Other...

Say a bit about yourself

- Name
- What Industry you are from?
- Do you have any prior knowledge in Python and Machine Learning?
- Why do you want to learn Machine Learning or Scikit-Learn?
- What do your expect to learn from this course?

Ground Rules

- Set your mobile phone to silent mode
- Participate actively in the class. No question is stupid.
- Mutual respect. Agree to disagree.
- One conversation at one time.
- Be punctual. Back from breaks on time.
- Exit the class silently if you need to step out for phone call, toilet break etc.
- 75% attendance is required

Ground Rules for Virtual Training

- Upon entering, mute your mic and turn on the video. Use a headset if you can
- Use the 'raise hand' function to indicate when you want to speak
- Participant actively. Feel free to ask questions on the chat whenever.
- Facilitators can use breakout rooms for private sessions.

WSQ and SSG TG Application Form

Please fill up the following WSQ and SSG TG Application Form for TRACOM survey, e-cert generation and WSQ funding

https://forms.gle/pJ2WxHZ3fyRbDLVu6

Digital Attendance

- You need to take digital attendance in the AM and PM.
- Please download the mySkillsFuture apps below to take your digital attendance for WSQ and SFC courses.

Guidelines for Facilitators

1. Once all the participants are in and introduce themselves
2. Goto gallery mode, take a snapshot of the class photo - makes sure capture the date and time
3. Start the video recording (only for WSQ courses)
4. Continue the class
5. Before the class end on that day, take another snapshot of the class photo - makes sure capture the date and time
6. For NRIC verification, facilitator to create breakout room for individual participant to check (only for WSQ courses)
7. Before the assessment start, take another snapshot of the class photo - makes sure capture the date and time (only for WSQ courses)
8. For Oral Questioning assessment, facilitator to create breakout room for individual participant to OQ (only for WSQ courses)
9. End the video recording and upload to cloud (only for WSQ courses)
10. Assessor to send all the assessment records, assessment plan and photo and video to the staff (only for WSQ courses).

Prerequisite

- Basic Python is assumed in this course.
- No Machine Learning knowledge is required

Learning Outcomes

By end of the course, learners will be able to

LO1 - Learners will be able to understand and apply machine learning concepts

LO2 - Learners will be able to understand and apply classification algorithms

LO3 - Learners will be able to understand and apply regression algorithms

LO4 - Learners will be able to understand and apply clustering algorithms

LO5 - Learners will be able to understand and apply PCA algorithms

Agenda

Topic 1 Overview of Machine Learning and Scikit Learn

- Introduction to Machine Learning
- Supervised vs Unsupervised Learnings
- Machine Learning Applications and Case Studies
- What is Scikit Learn
- Installing Scikit-Learn

Topic 2 Classification

- What is Classification
- Applications of Classification
- Classification Algorithms
- Classification Workflow
- Confusion Matrix
- Classification Performance Evaluation

Agenda

Topic 3 Regression

- What is Regression
- Applications of Regression
- Regression Algorithms
- Regression Workflow
- Regression Performance Evaluation

Topic 4 Clustering

- What is Clustering
- Applications of Clustering
- Clustering Algorithms
- Clustering Workflow
- Clustering Performance Evaluation

Agenda

Topic 5 Principal Component Analysis

- Introduction to Principal Component Analysis (PCA)
- Application of PCA
- PCA Workflow

Practice Session

Final Assessment

- Written Assessment (Q&A)
- Written Assessment (Case Study)

Download Course Material

- You can download the course material from Google classroom https://classroom.google.com
- Enter the class code below to join the class on the top right.
- Goto Classwork>> Course material
- If you cannot access the Google Classroom, please inform trainer or staff.

3y736nl

Sample Codes

Download the exercise file from

https://github.com/tertiarycourses/WSQ-Scikit-Learn/blob/master/scikit.ipynb

or

https://colab.research.google.com/drive/1q7g-DjbRIT5iEam9u4v8dJ5tAn-ziCN3?usp=sharing

If you have problem seeing the exercise file,, You can load this github link to the Google Colab

CERTIFICATE

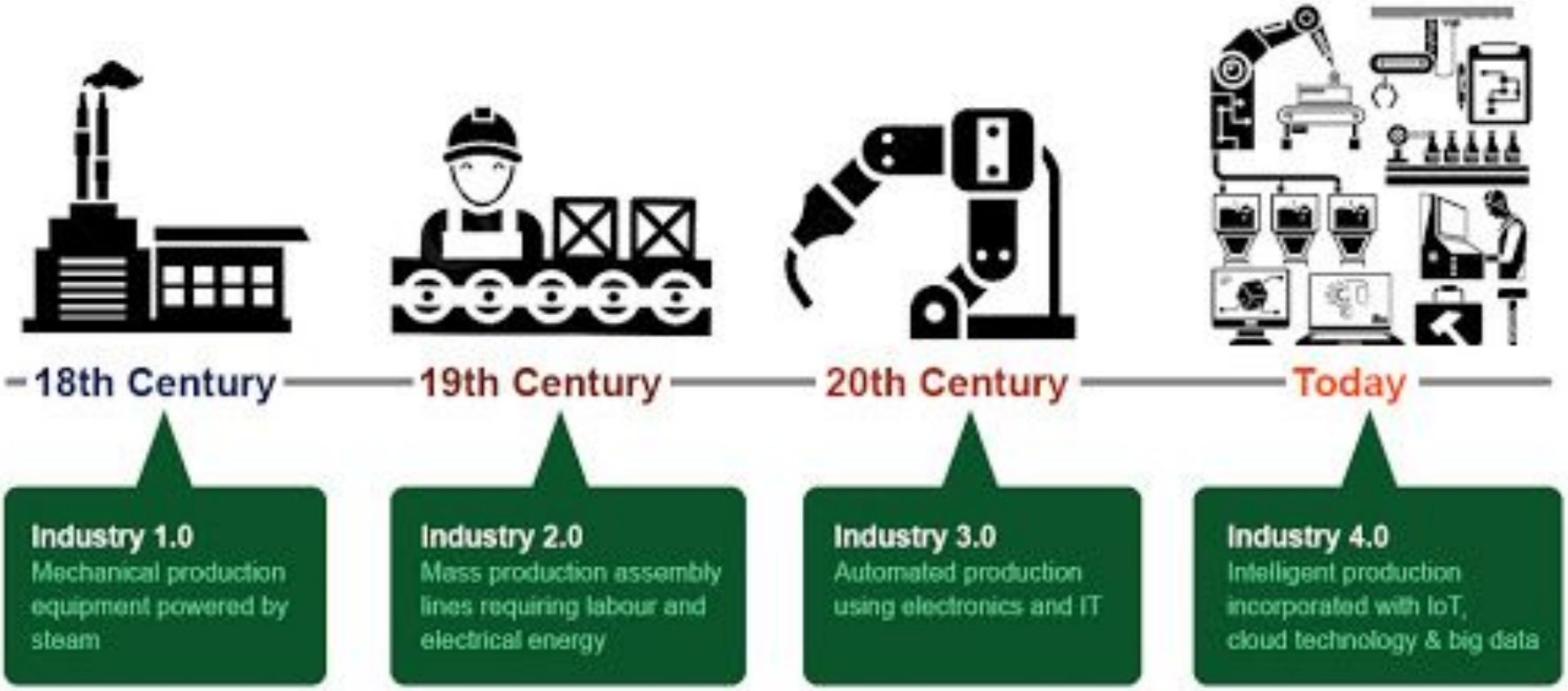
Two e-certificates will be awarded to trainees who have demonstrated competency in the WSQ assessment and achieved at least 75% attendance.

- A SkillsFuture WSQ Statement of Attainment (SOA) issued by WSG. Typically take 4 weeks
- Certification of Completion issued by Tertiary Infotech Pte Ltd, immediately after the course

Topic 1
Overview of Machine Learning and Scikit Learn

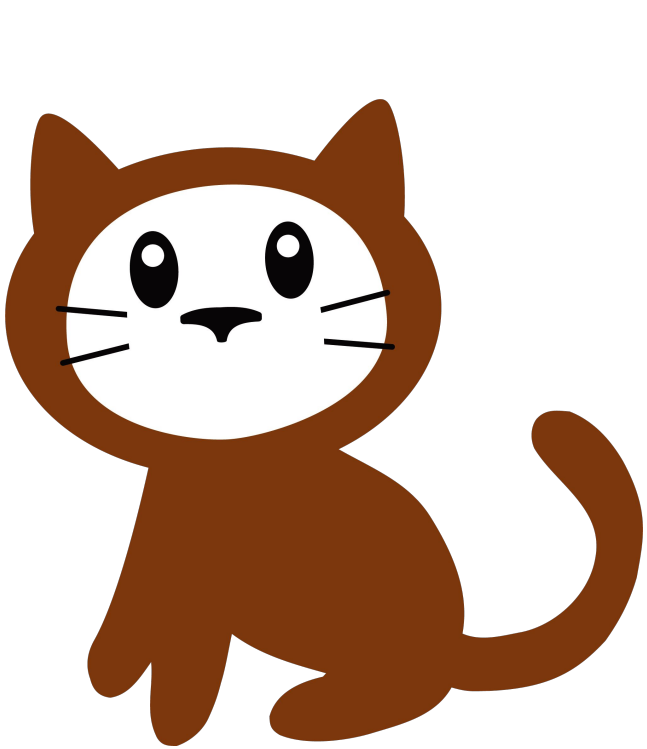
Industry 4.0 Disruption

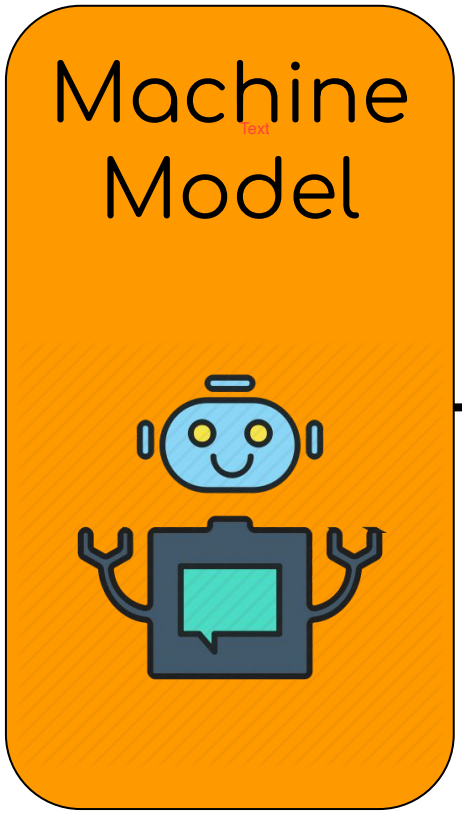
- We are at the start of industry 4.0 revolution - intelligent production (systems) powered by IoT + AI
- The development of AI and IoT will be driven by the advances of cloud computing, big data and 5G.

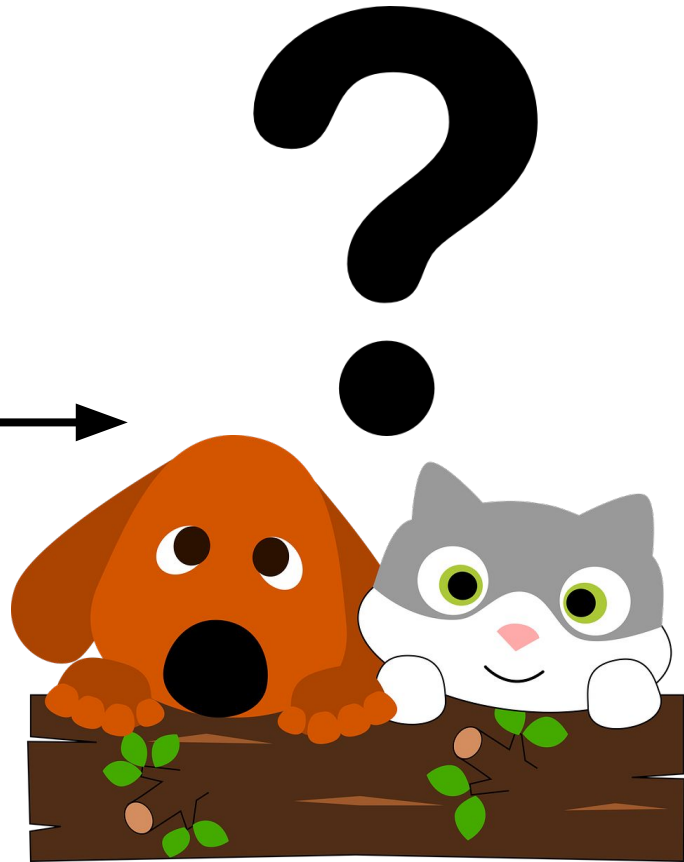


What is Machine Learning?

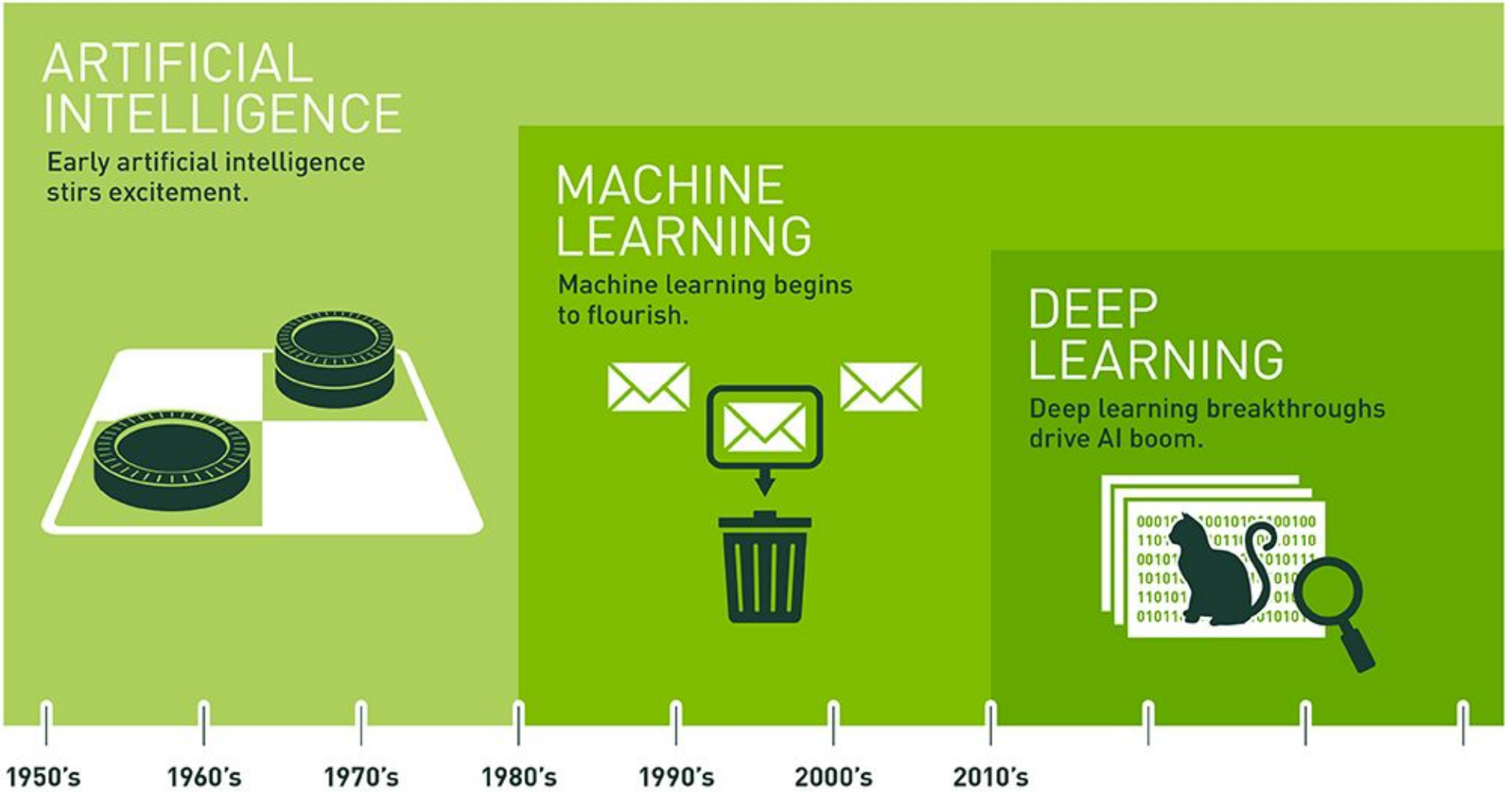
After you trained the machine to recognize dog or cat, then machine is able to tell your the answer when you give it an image



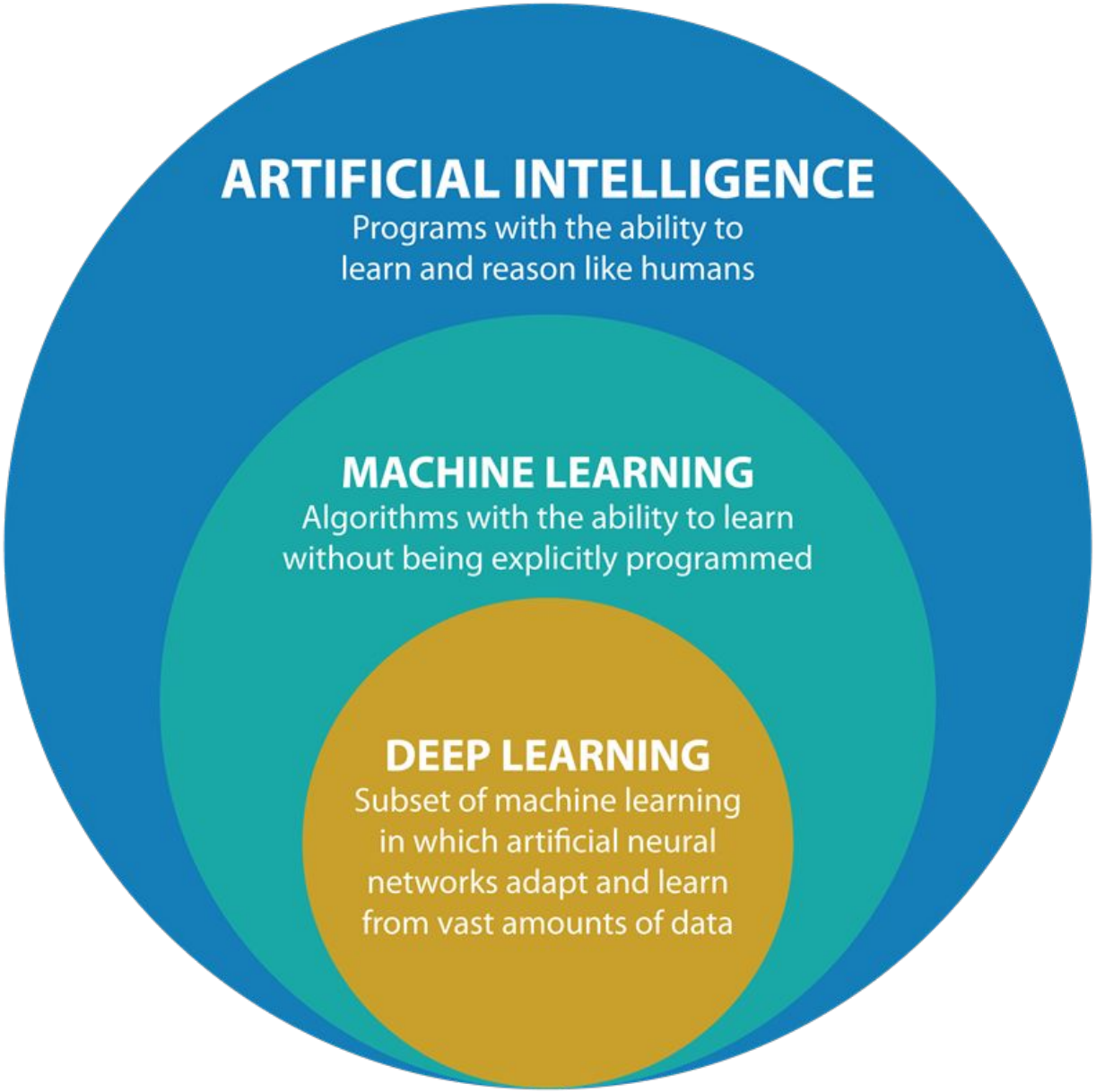




Evolution of AI

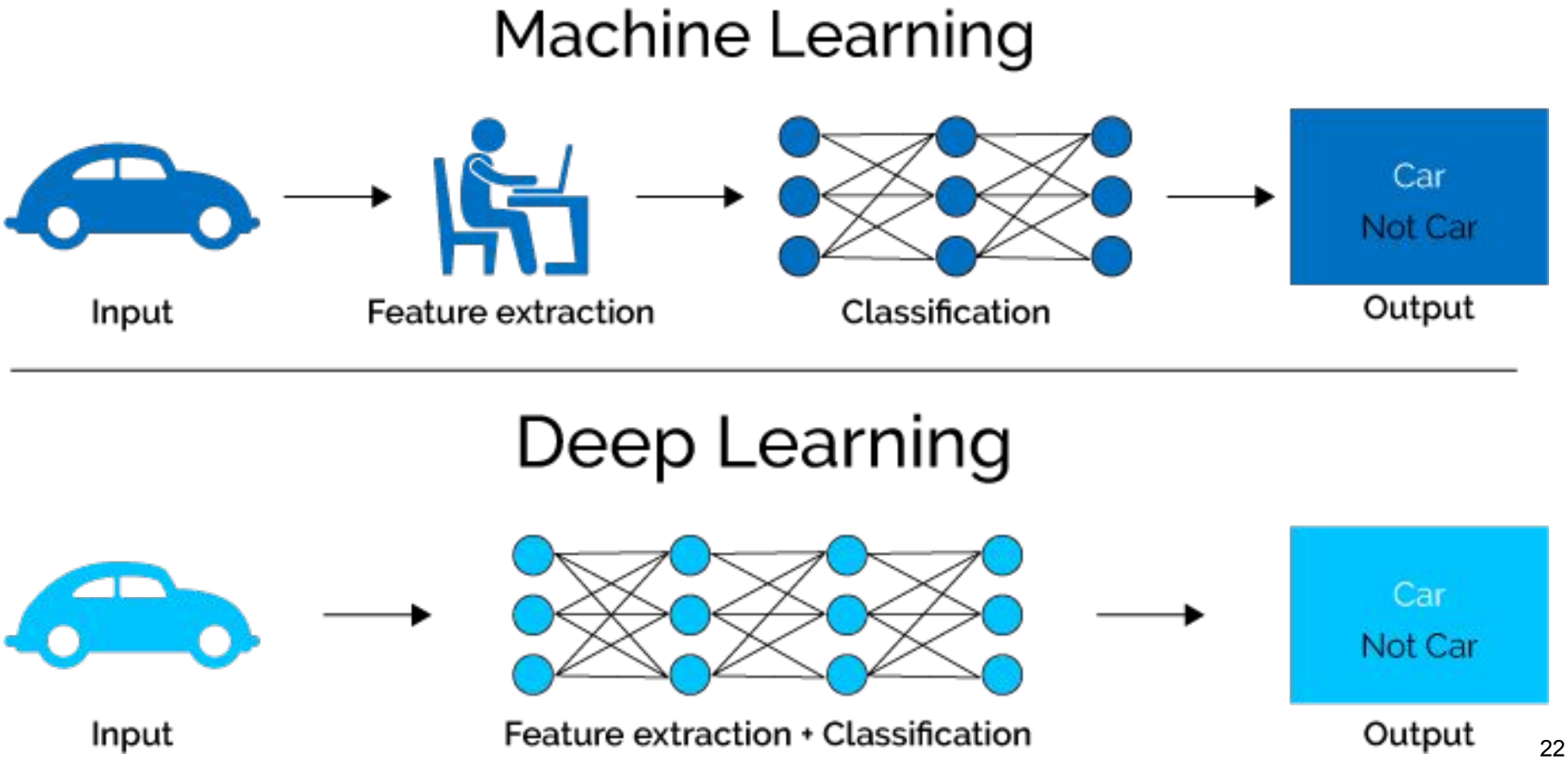


AI vs ML vs DL


ARTIFICIAL INTELLIGENCE
Programs with the ability to learn and reason like humans
MACHINE LEARNING
Algorithms with the ability to learn without being explicitly programmed
DEEP LEARNING
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data

Machine Learning vs Deep Learning

Deep Learning are neural network based and consists of millions of parameters to train.



Feature extraction done by DL instead of programmer

Machine Learning in our Daily LIfe

Spam Filtering

Web Search

Postal Mail Routing

Fraud Detection

Movie Recommendations

Vehicle Driver Assistance

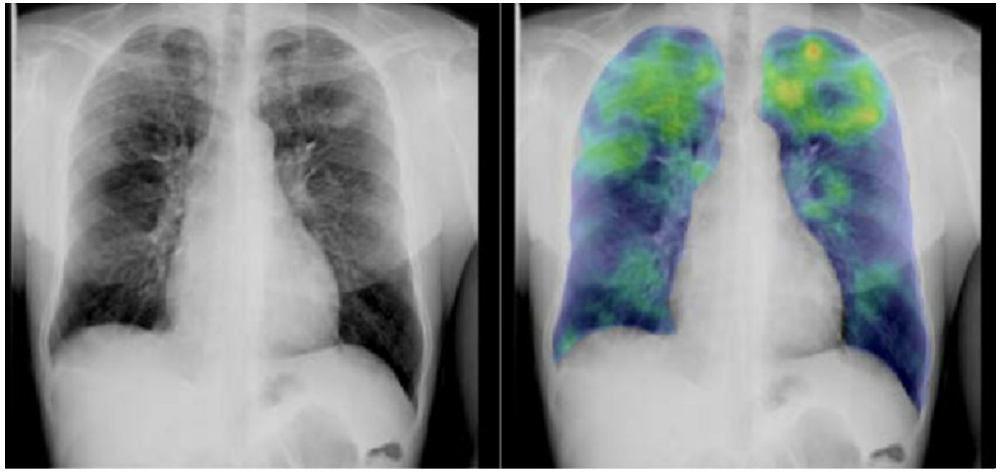
Web Advertisement

Social Networks

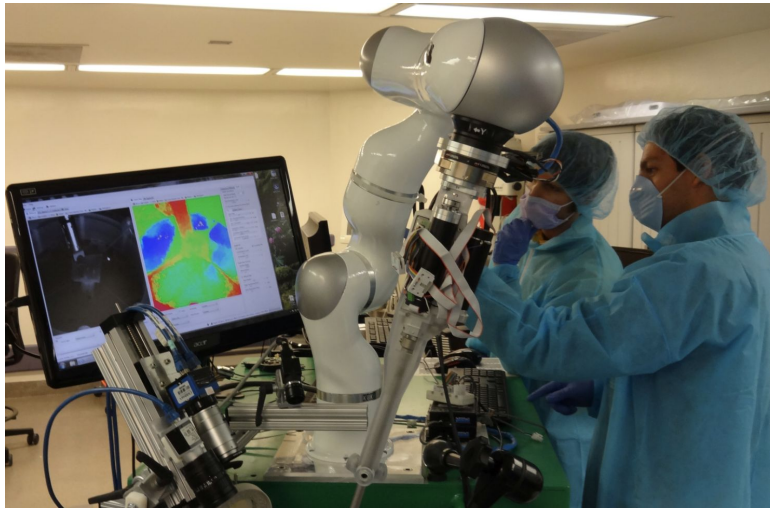
Speech Recognition

AI Applications on Health Care

Identifying Tuberculosis

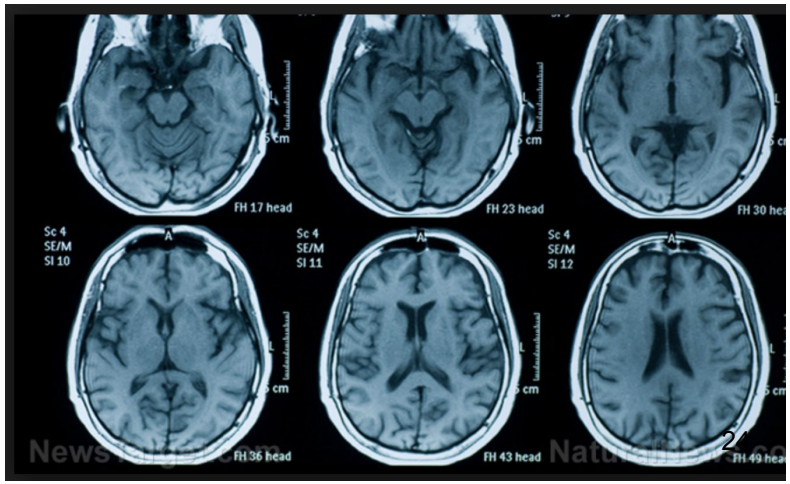


Robotics-Assisted Surgery



Detecting Brain Bleeds

Detecting Alzheimer

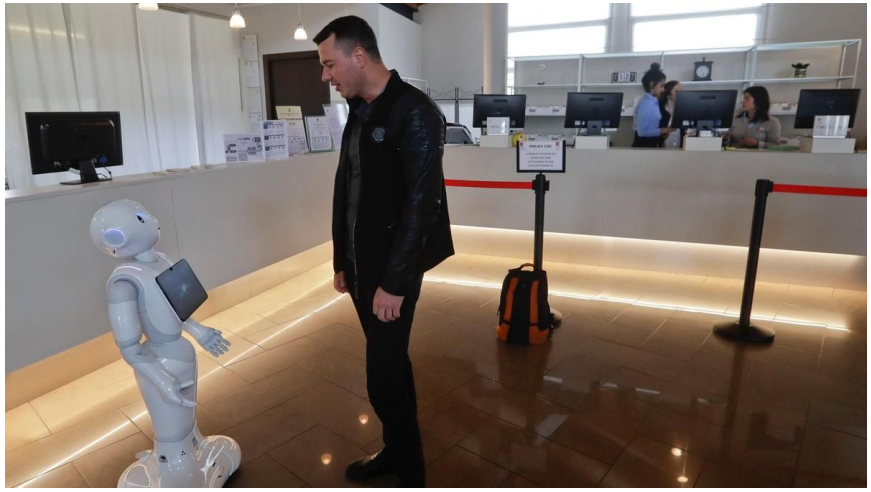


AI Applications on Retail/Logistics

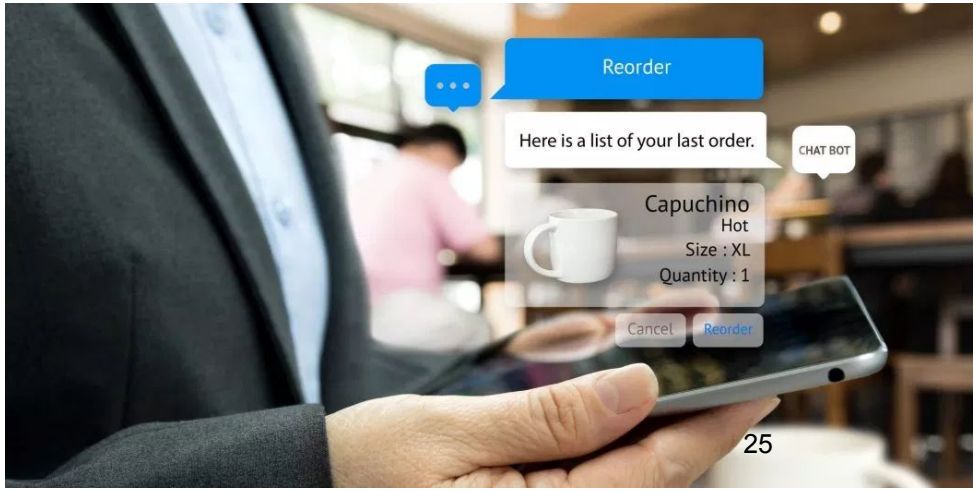
Unmanned Store: Amazon Go

Unmanned store

Frontdesk Robots



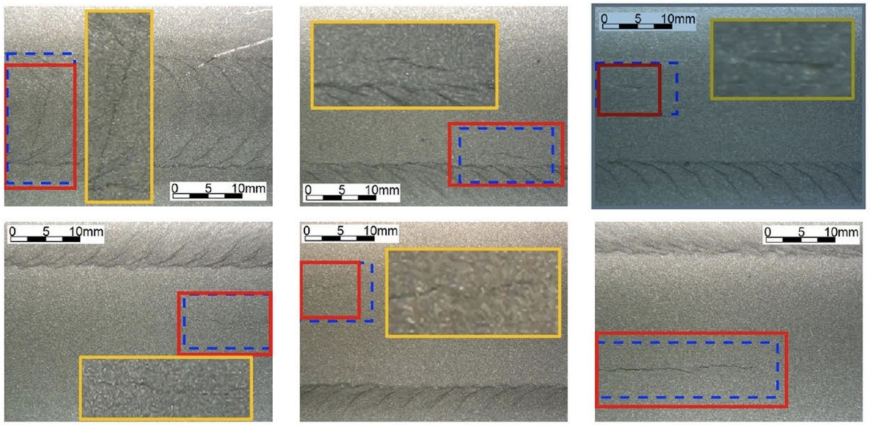
Chatbot for retail/services



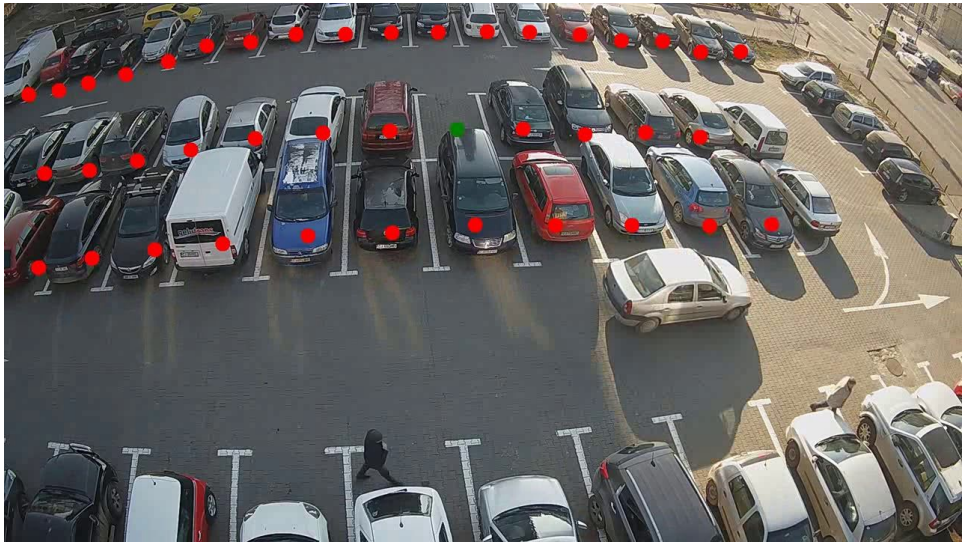
AI Applications on Security/Mfg

Face recognition

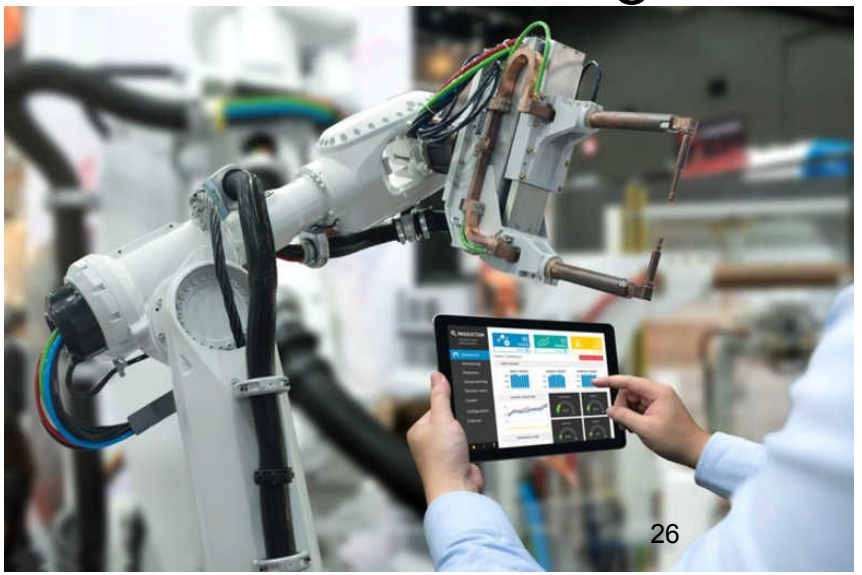
Defect Detection



Parking Spotter



Robots in Mfg



What's Dog?

https://www.what-dog.net/

Supervised & Unsupervised Learning

Supervised Learning

Data Points have known outcome

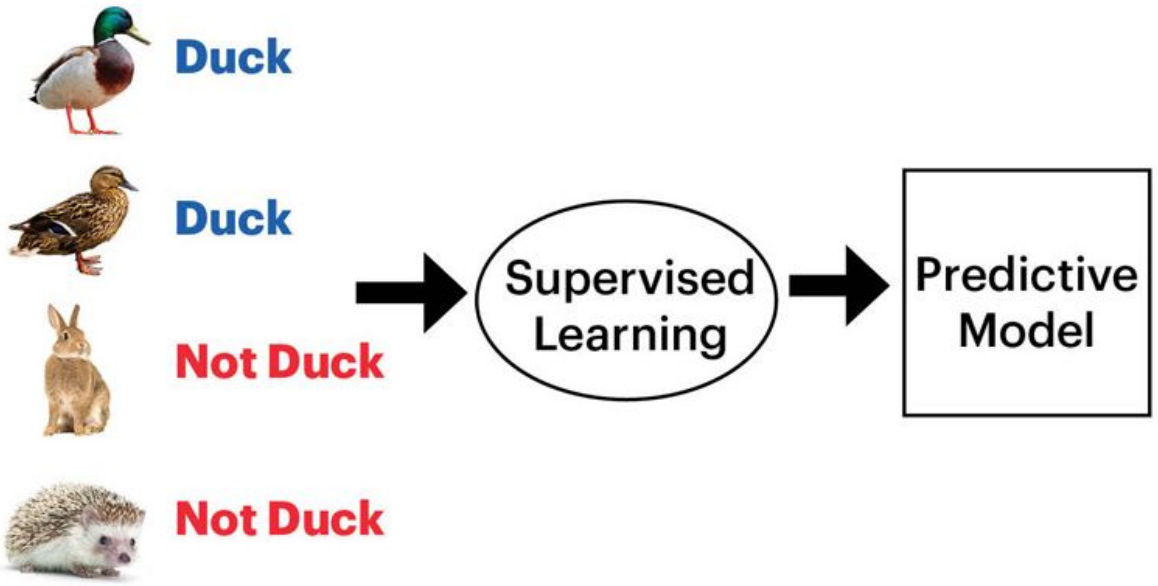
Unsupervised Learning

Data Points have unknown outcome

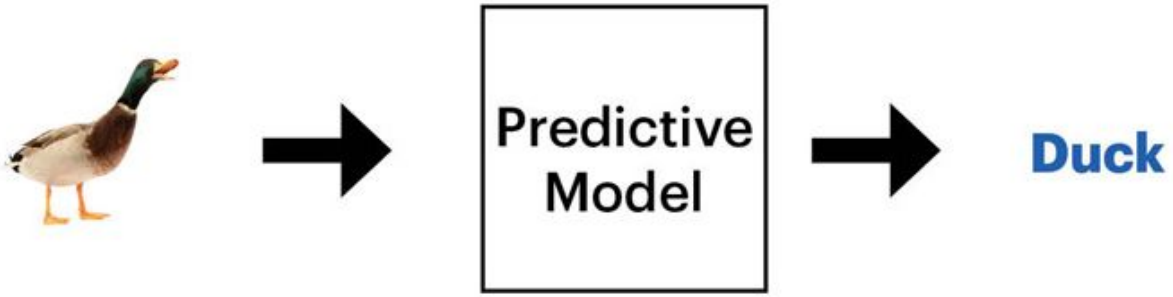
Supervised Learning

Supervised learning learns patterns from labelled data and then makes predictions and try to label new data

TRAINING

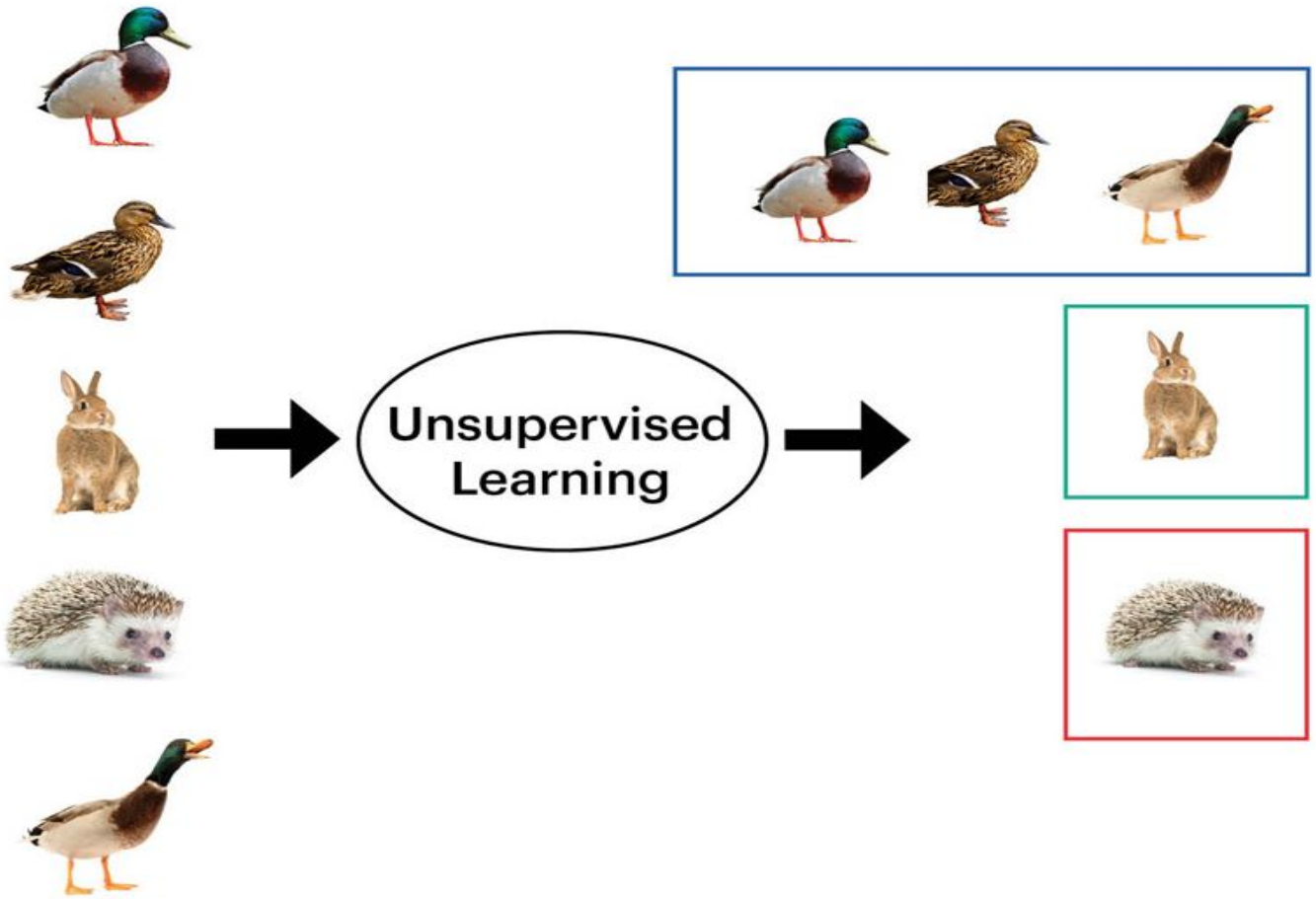


INFERENCE



UnSupervised Learning

- Supervised learning uses unlabelled data, correct classes are not known.
- Interpret and groups the input data only

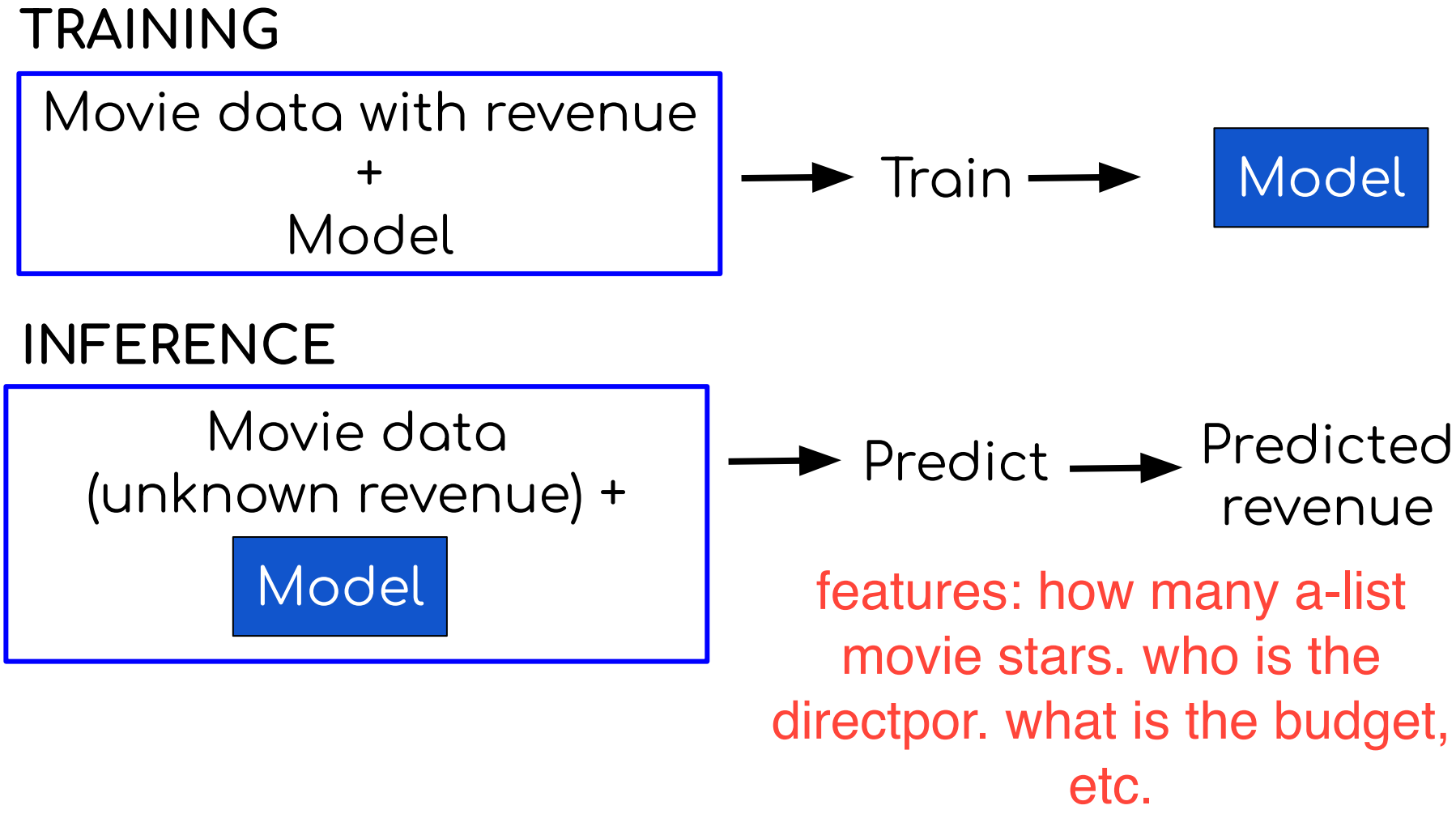


Types of Supervised Learning

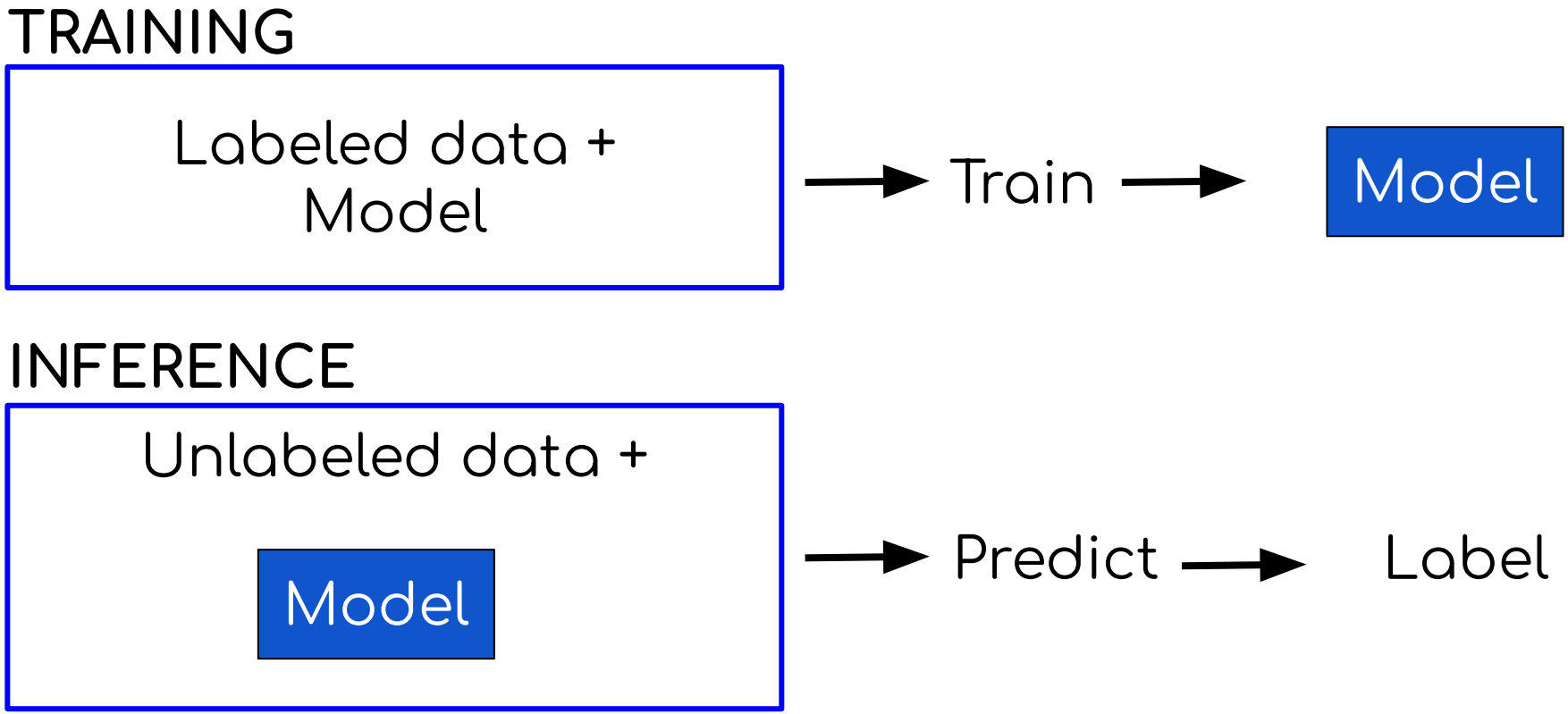
Regression — Outcome is continuous (numerical)

Classification — Outcome is a category

Regression: Numerical Answers



Classification: Categorical Answers



Classification Example

enough data has been collected to say that it is or not spam (i.e. key words like viagra, etc)

TRAINING

Emails labeled as spam/not spam
Model

→ Train →

Model

INFERENCE

Unlabeled email +

Model

→ Predict →

Spam or Not spam

Machine Learning Vocabulary

features, columns, inputs

Feature
s

independent variables

Target

output, dependant variable

Sepal length	Sepal width	Petal length	Petal width	Species
6.7	3.0	5.2	2.3	Virginica
6.4	2.8	5.6	2.1	Virginica
4.6	3.4	1.4	0.3	Setosa
6.9	3.1	4.9	1.5	Versicolor
4.4	2.9	1.4	0.2	Setosa
4.8	3.0	1.4	0.1	Setosa
5.9	3.0	5.1	1.8	Virginica
5.4	3.9	1.3	0.4	Setosa
4.9	3.0	1.4	0.2	Setosa
5.4	3.4	1.7	0.2	Setosa

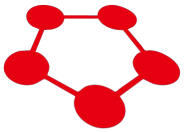
Label

Example

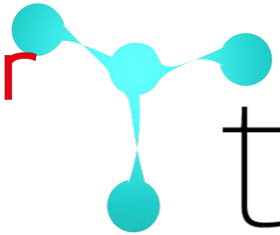
Machine Learning Vocabulary

- Target: Predicted category or value of the data (column to predict)
- Features: properties of the data used for prediction (non-target columns)
- Example: a single data point within the data (one row)
- Label: the target value for a single data point.

Machine Learning Frameworks



Chainer



Caffe

torch

theano

What is Scikits Learn

- Scikit-learn (http://scikit-learn.org) is an open source Python library of popular machine learning algorithms
- It started in 2007 as a Google Summer of Code project
- It is built upon NumPy and SciPy

Scikits Learn Applications

- **Classifications**
  - Identifying to which category an object belongs to.
  - Spam detection, Image recognition.
- **Regression**
  - Predicting a continuous-valued attribute associated with an object.
  - Drug response, Stock prices.
- **Clustering**
  - Automatic grouping of similar objects into sets.
  - Customer segmentation, Grouping experiment outcomes

Scikits Learn Applications

- **Dimensional Reduction**
  - Reducing the number of random variables to consider.
  - Visualization, Increased efficiency
- **Model Selection**
  - Comparing, validating and choosing parameters and models.
  - Improved accuracy via parameter tuning
- **Preprocessing**
  - Feature extraction and normalization.
  - Transforming input data such as text for use with machine learning algorithms.

Install Scikit Learn

You can install scikit learn by typing the following command to terminal or CMD Prompt

pip install sklearn

pip3 install sklearn

Google Colab for Scikit Learn

- Google Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud.
- With Colaboratory you can write and execute Python and Scikit Learn code, save and share your analyses
- You can also access computing resources such as GPU and TPU for free
- You can access Google Colab by typing

  https://colab.research.google.com

Checking your installation

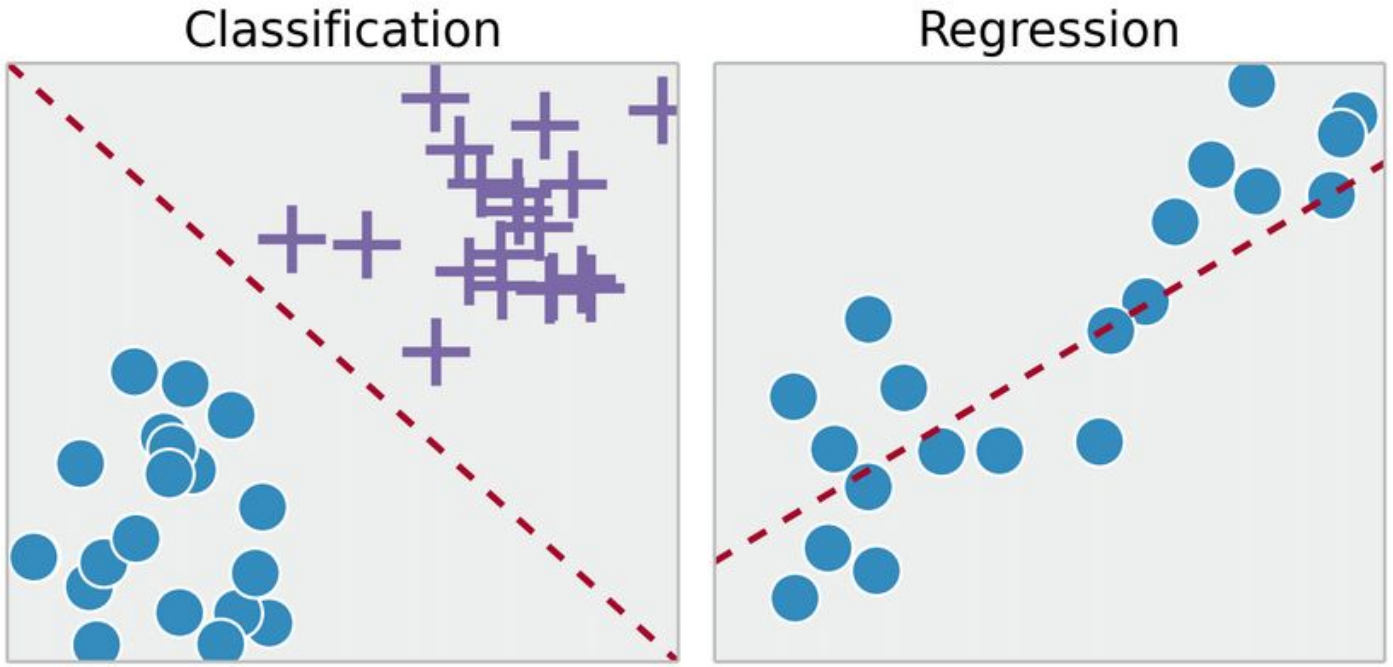
You can check your scikit learn installation by typing the following command to terminal or CMD Prompt

```
import sklearn
```

Topic 2
Classification

What is Classification?

- Classification is to identifying to which category an object belongs to.
- Classification is a supervised learning method. During training, labels/classes are provided.

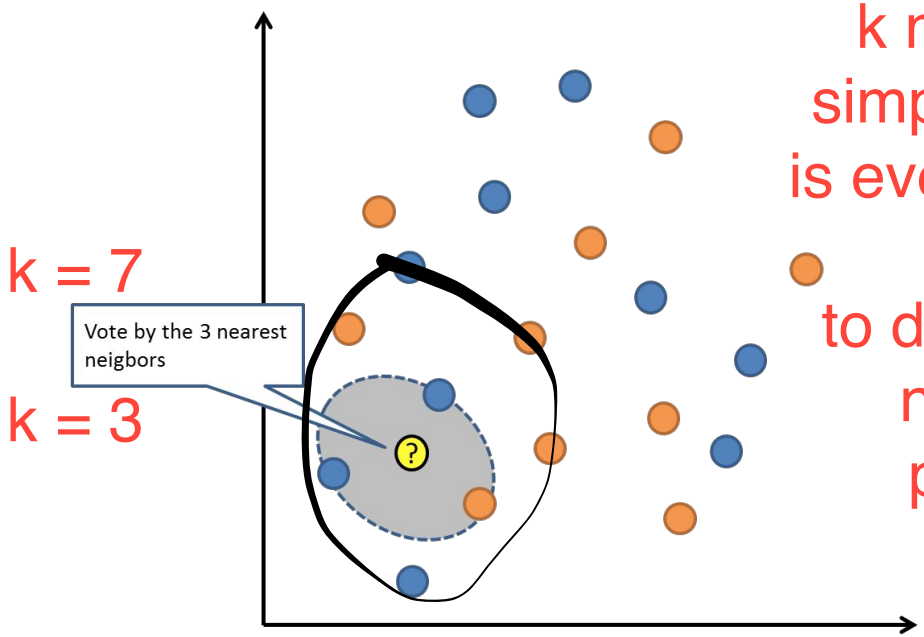


Classification Algorithms

- K Nearest Neighbors
- Logistic Regression (not regression)
- Support Vector Machine
- Gaussian Naive Bayes
- Stochastic Gradient Descent
- Decision Tree
- Ensemble Methods
  - Random Forest
  - Gradient Boost
  - XGBoost

K Nearest Neighbors (KNN)

- A simple majority vote of the nearest neighbors of each point.
- Training data is the model. Fitting is fast just store data. However Prediction can be slow because lots of distances to measure
- Decision boundary is flexible



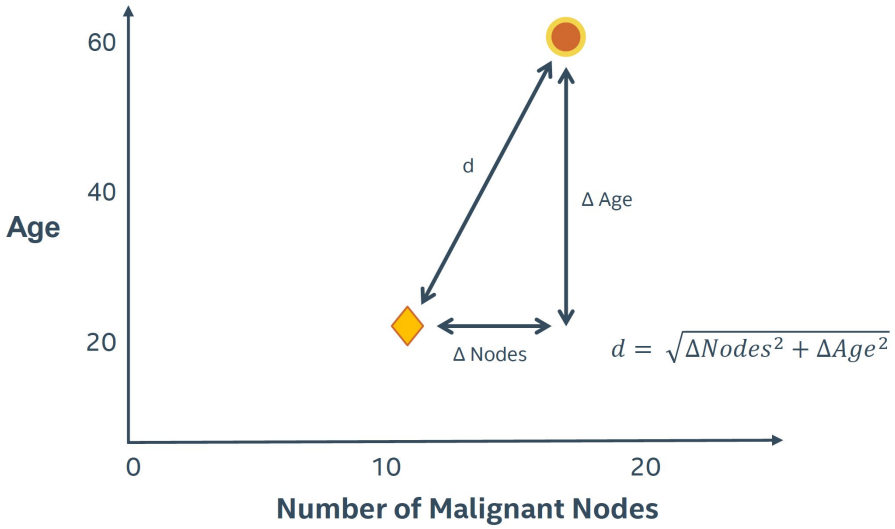
k must be odd (it is a simple majority vote, if it is even, you have a draw)

to determine best k, we need to do hyper-parameter tuning

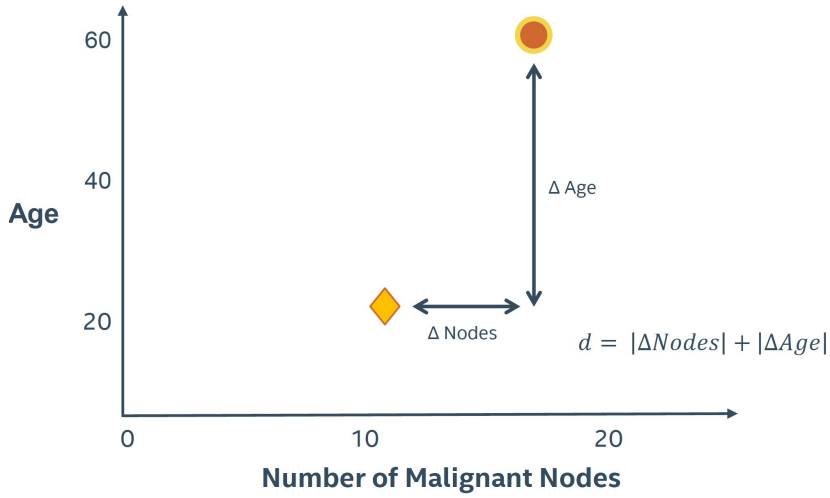
Distance Measure for KNN

- KNN depends on the distance measure
- There are two major distance measures:
  - Euclidean (L2)
  - Manhattan (L1)

Euclidean (L2)



Manhattan (L1)



KNN Classifier

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier()
```

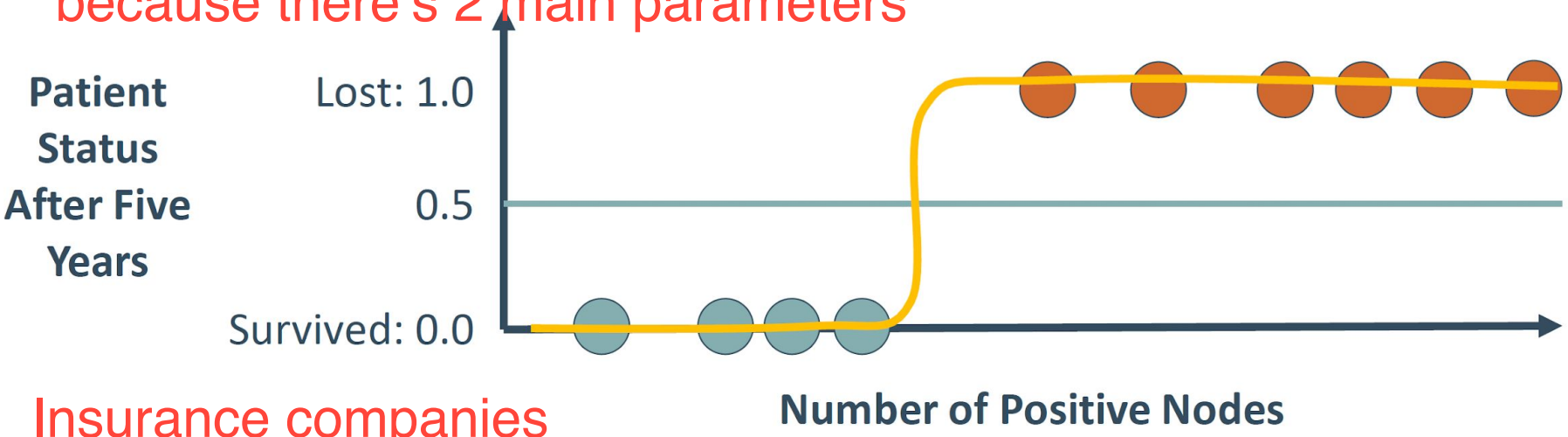
Parameters:

```
KNeighborsClassifier(n_neighbors=5,
weights='uniform', algorithm='auto', leaf_size=30, p=2,
metric='minkowski', metric_params=None,
n_jobs=None, **kwargs)
```

Logistic Regression

- Fit the logistic function to the data
- Fitting can be slow must find best parameters
- Prediction is fast calculate expected value
- Decision boundary is simple, less flexible

Within KPI or not, can use this or not, because there's 2 main parameters



Insurance companies use this to calculate probability of making a claim

$$y_\beta(x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x+\varepsilon)}}$$

Logistic Regression Classifier

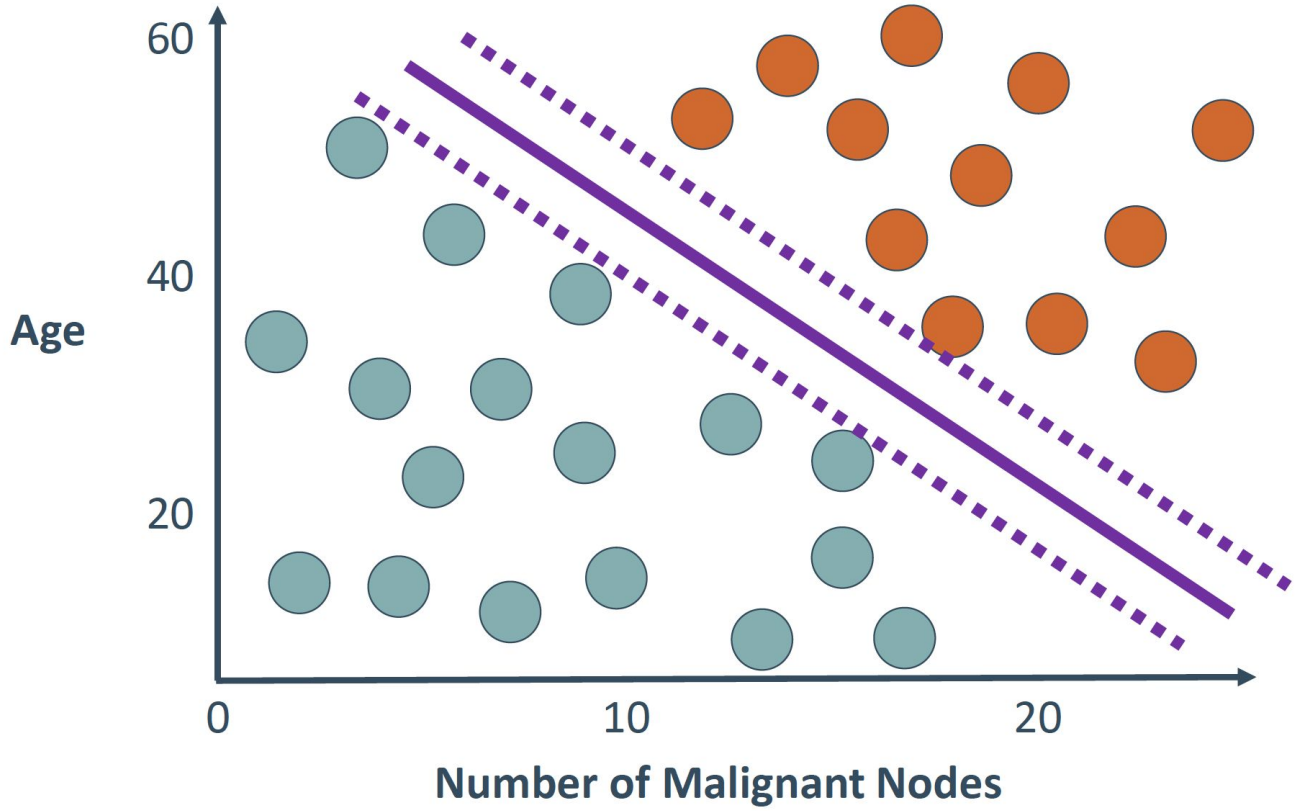
```
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
```

Default values:

```
LogisticRegression(penalty='l2', dual=False, tol=0.0001,
C=1.0, fit_intercept=True, intercept_scaling=1,
class_weight=None, random_state=None, solver='lbfgs',
max_iter=100, multi_class='auto', verbose=0,
warm_start=False, n_jobs=None, l1_ratio=None)[source]
```

Support Vector Machine (SVM)

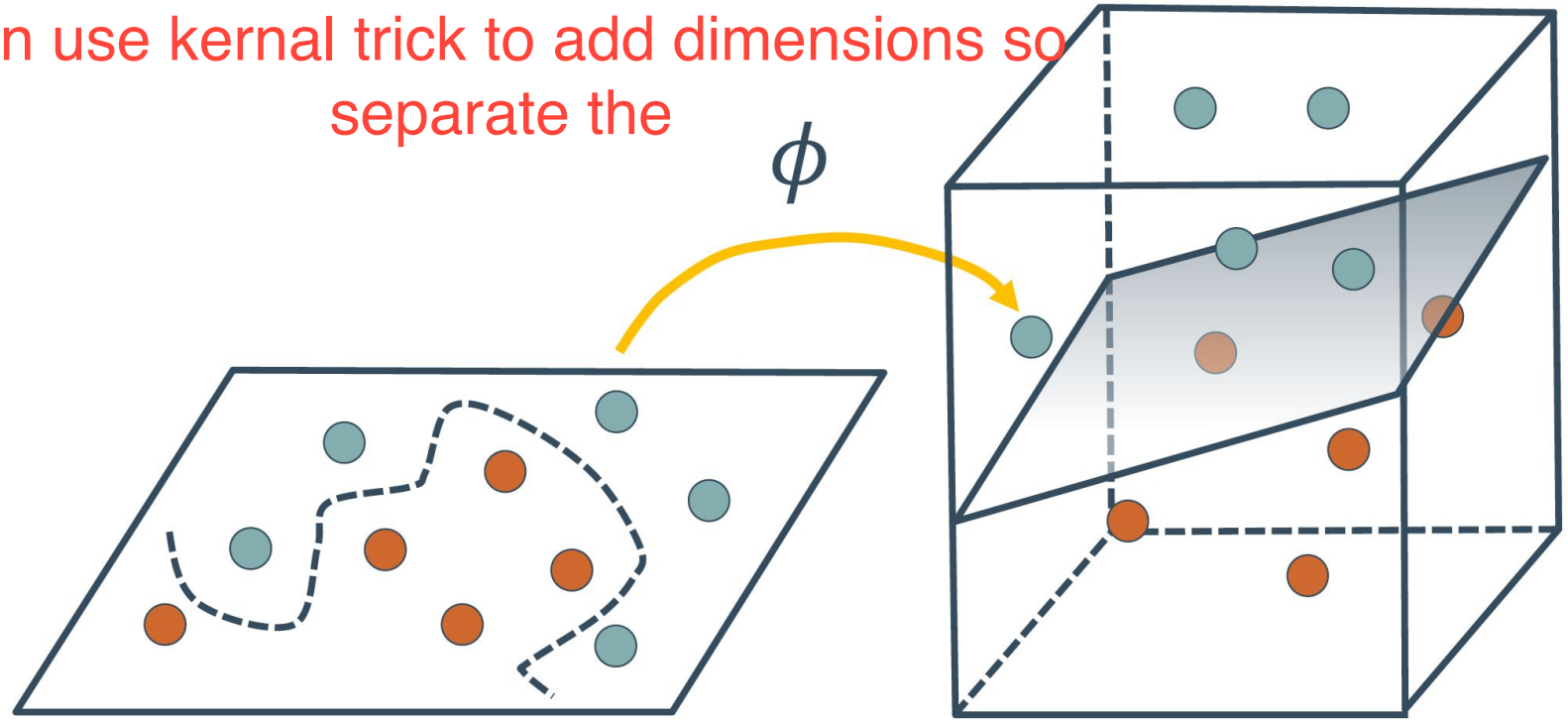
Identify a linear hyperplane (boundary) with maximum distance apart



Kernel Trick

Non-linear data can be made linear with higher dimension using Kernel trick

must always be straight line is using SVM.
can use kernal trick to add dimensions so
separate the



Kernels for SVM

The following 4 kernels are available in Scikit Learn:

- linear No processing at all
- poly
- rbf (default) radial basis function - radial, elliptical
- sigmoid

SVM Classifier

```
from sklearn import svm
clf = svm.SVC()
```

Default values:

```
SVC(C=1.0, kernel='rbf', degree=3, gamma='scale',
coef0=0.0, shrinking=True, probability=False, tol=0.001,
cache_size=200, class_weight=None, verbose=False,
max_iter=-1, decision_function_shape='ovr',
break_ties=False, random_state=None)
```

Gaussian Naive Bayes (GNB)

Gaussian = normal distribution

Naive = independence (refer to stats)

- Naive Bayes is a conditional probability model.
- Given the feature vector x, it predict the probability for class C
- GNB assumes each feature is Gaussian distributed

class - male, female

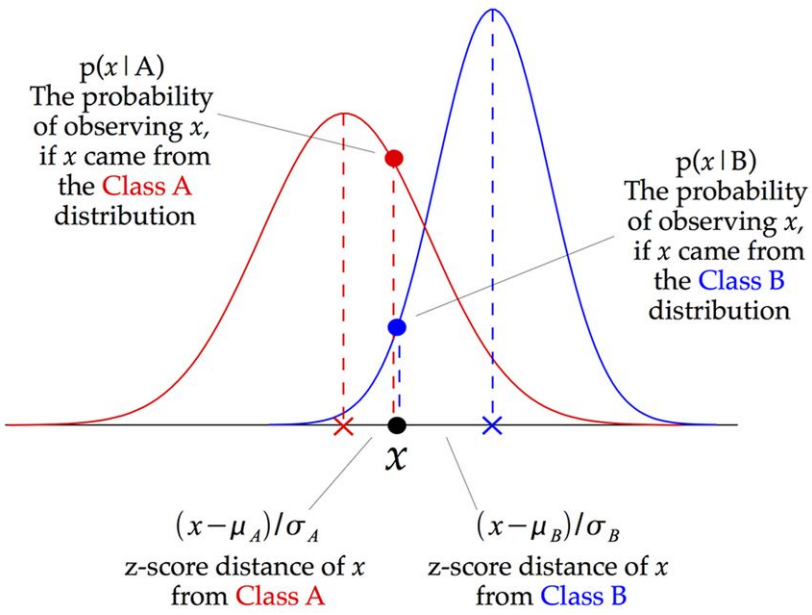
distributions - height, weight, foot size

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})} \qquad \text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

GNB Explanation

Person	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9



Person	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	$3.5033*10^{-2}$	176.25	$1.2292*10^{2}$	11.25	$9.1667*10^{-1}$
female	5.4175	$9.7225*10^{-2}$	132.5	$5.5833*10^{2}$	7.5	1.6667

P(male|x)=P(height|male)*P(weight|male)*P(foot size|male)*P(male)
P(female|x)=P(height|female)*P(weight|female)*P(foot size|female)*P(male)

Example of GNB

Below is a sample to be classified as male or female.

Person	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

posterior of male ~ 6.19×10^{-9}

posterior of female ~ 5.37×10^{-4}

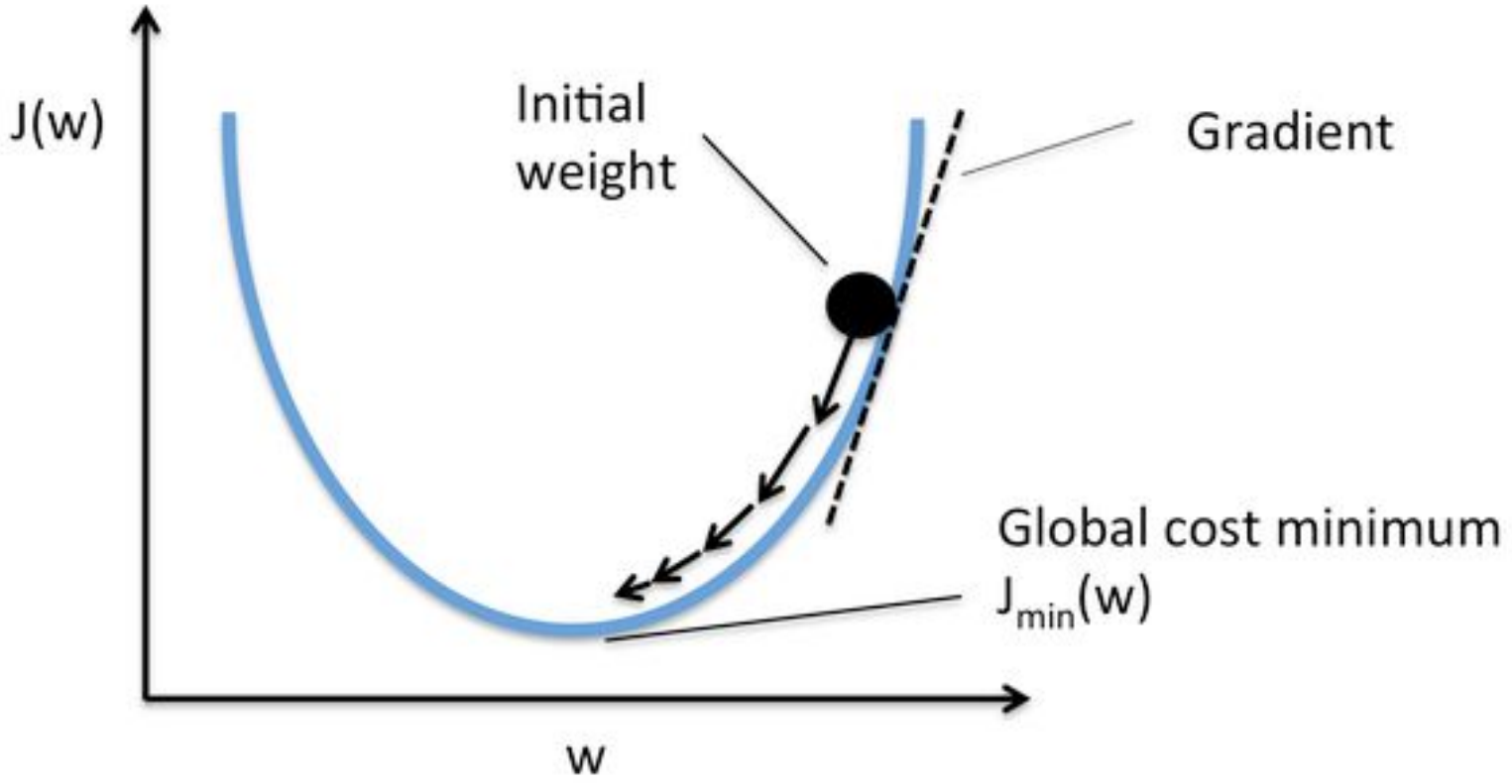
Therefore the person is likely to be female.

GND Classifier

```
from sklearn.naive_bayes import GauassianNB()
clf = GaussianNB()
```

Stochastic Gradient Descent

Stochastic gradient descent (SGD) performs a parameter update for each training using the negative gradient

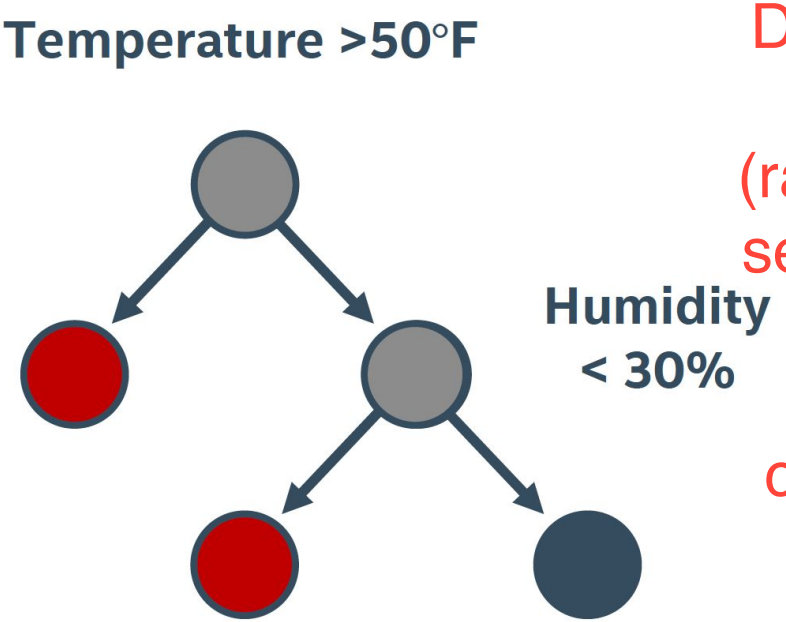


SGD Classifier

```
from sklearn.linear_model import SGDClassifier
clf = SGDClassifier()
```

Decision Tree

- Decision tree is easy to interpret and implement
- Heterogeneous input data allowed, preprocessing
- required
- However, decision trees tend to overfit
- Pruning helps reduce variance to a point. Often not significant for model to generalize well



Decision tree will be good for ordinal (ranked)classes - pri, sec, a level/poly, etc.

Hence need a certain level of pre-process

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
```

If cannot be ranked (ordinal), need to have dummy variables (north south east west)

Iris Flower Dataset

setosa (0)

versicolor (1)

virginica (2)

Iris flower dataset, introduced in 1936 by Sir Ronald Fisher

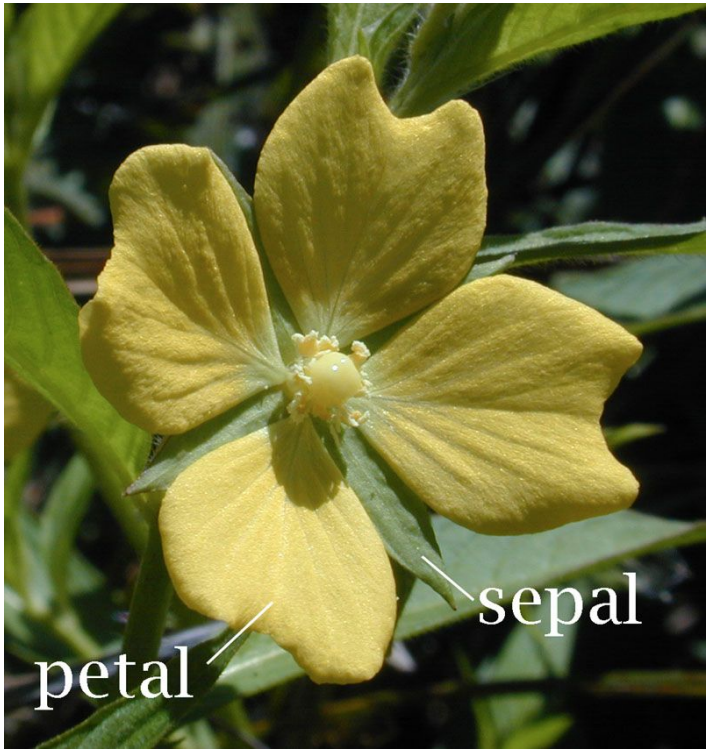
Iris Flower Dataset

Features in the Iris dataset:

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm

Target classes to predict:

- setosa : 0
- versicolor : 1
- virginica : 2



Step 1 Prepare the Data

```
import pandas as pd

dataset_path =
"https://raw.githubusercontent.com/tertiarycourses
/datasets/master/iris.csv"

X = pd.read_csv(dataset_path)

# Drop any missing data
X = X.dropna()
```

Extract Features and Target

```
y = X.pop('Name')


# Encode the Labels
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)
```

Normalize/Scale the Features

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_scaled = pd.DataFrame(scaler.fit_transform(X),
columns=X.columns)
```

Split and Randomize Data

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=0.25,
random_state=33)
```

Step 2 Define the Model

```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(n_neighbors=5)
```

Step 3 Train the Model

```
clf.fit(X_train,y_train)
```

Step 4 Evaluate the Model

clf.score(X_test,y_test)

Step 5 Save the Model

```
joblib.dump(clf, 'mymodel.pkl')
```

Step 6 Load the Model for Inference

```
from sklearn.externals import joblib
clf2 = joblib.load('iris.pkl')

import numpy as np

X_new = np.array([[6.7,3.1,4.7,1.5]])
y = clf2.predict(X_new)
label = {0:'sentosa',1:'versicolor',2:'virginica'}
print('The flower is ',label[y[0]])
```

Ex: Classification

Compare the following classifiers on iris dataset and evaluate which one is the best and worst classifier

- KNN
- Logistic Regression
- SVM
- GND
- SGD
- DT

Time: 10 mins

What is your guess?

Ensemble Methods

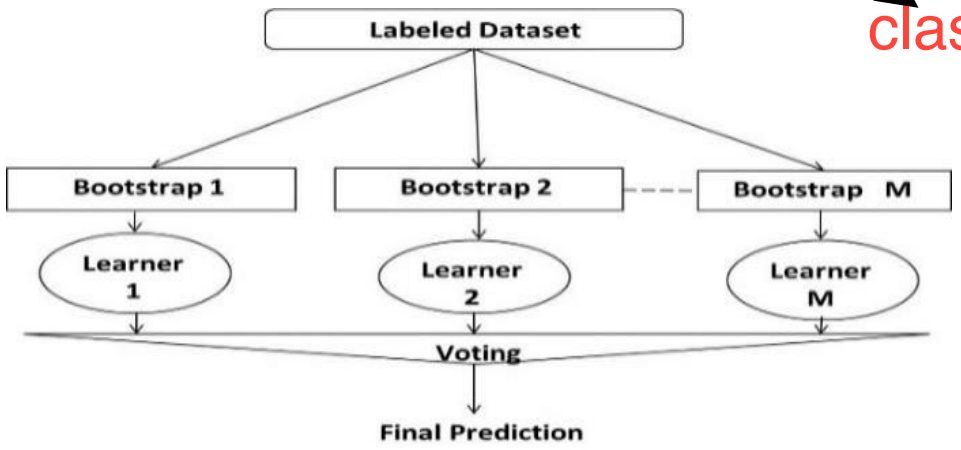
The main principle behind Ensemble methods is that a group of "weak learners" can come together to form a "strong learner".

There are 3 types of Ensemble methods

- **Bagging** (Bootstrap Aggregating) creates separate samples of the training dataset and creates a classifier for each sample. The results of these multiple classifiers are then combined (such as averaged or majority voting).

Regression

classification

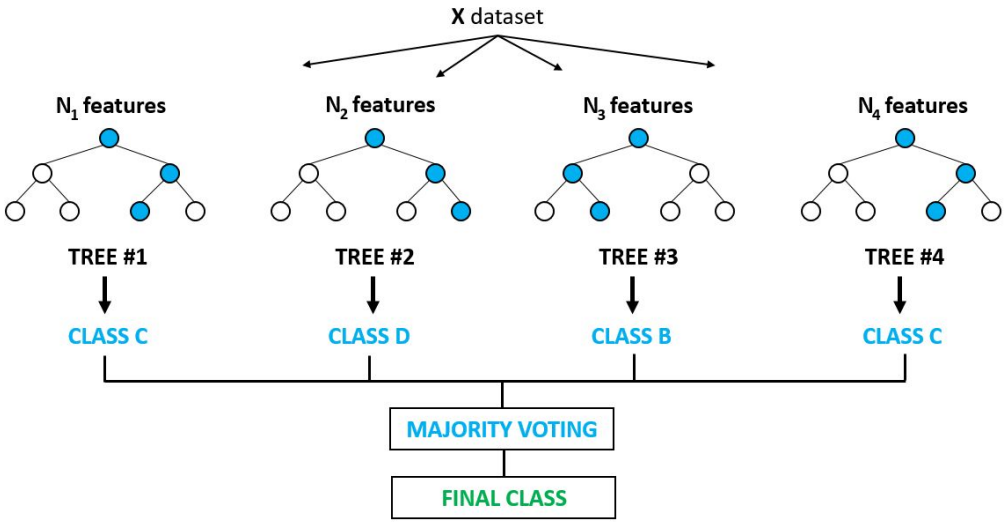


Ensemble Methods

- **Boosting** starts out with a base classifier that is prepared on the training data. A second classifier is then created behind it to focus on the instances in the training data that the first classifier got wrong. The process continues to add classifiers until a limit is reached in the number of models or accuracy
- **Stacking** starts out with a set of base-level classifiers and train a meta-level classifier to combine the outputs of the base-level classifiers

Random Forest a type of bagging

- Random forest is an ensemble learning method used for classification, regression and other tasks.
- It was first proposed by Tin Kam Ho and further developed by Leo Breiman (Breiman, 2001) and Adele Cutler
- Random Forest builds a set of decision trees. Each tree is developed from a bootstrap sample from the training data. When developing individual trees, an arbitrary subset of attributes is drawn (hence the term "Random"), from which the best attribute for the split is selected.
- The final model is based on the majority vote from individually developed trees in the forest

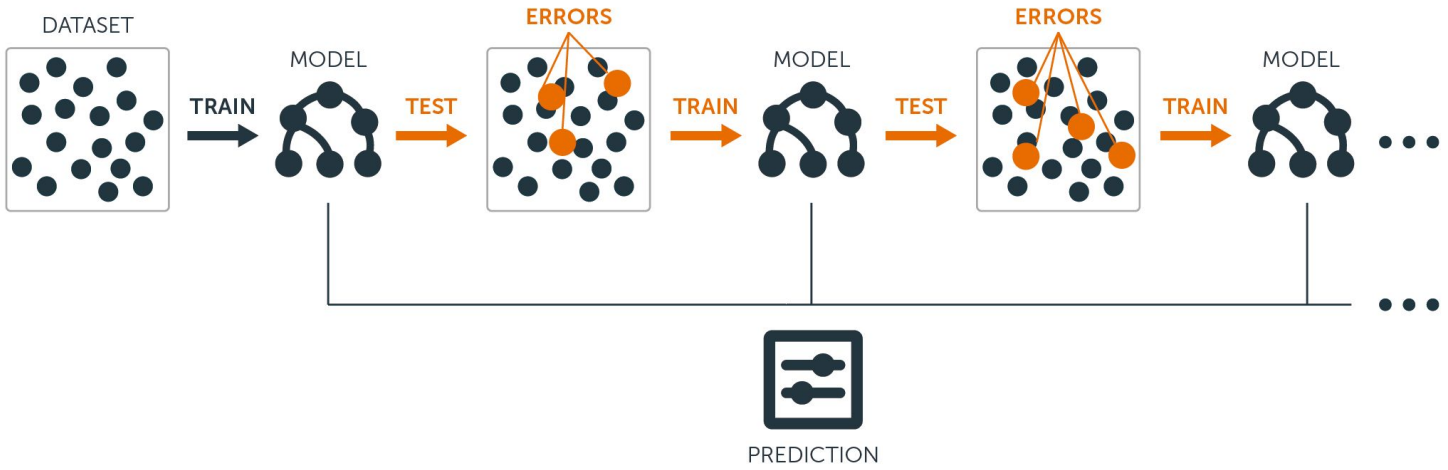


Random Forest Classifier

```
from sklearn.ensemble import
RandomForestClassifier


clf = RandomForestClassifier()
```

Gradient Boosting

- Gradient Boosting is a ensemble boosting method that "boosting" many weak predictive models into a strong one, in the form of ensemble of weak models.
- Boosting utilizes different loss functions At each stage, the margin is determined for each point. Margin is positive for correctly classified points and negative for misclassifications. Value of loss function is calculated from margin

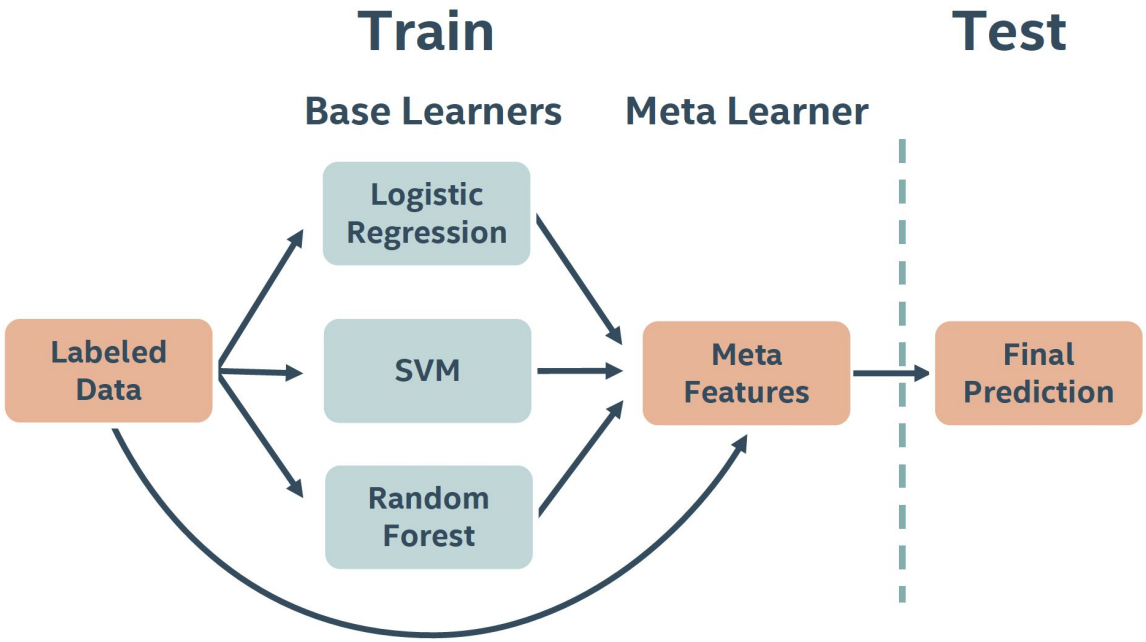


Gradient Boosting Classifier

```
from sklearn.ensemble import
GradientBoostingClassifier


clf = GradientBoostingClassifier()
```

Stacking

- Models of any kind combined to create stacked model
- Output of base learners can be combined via majority vote or weighted
- Additional hold out data needed if meta learner parameters are used
- Be aware of increasing model complexity



Stacking Classifier

```
from sklearn.ensemble import VotingClassifier

clf = VotingClassifier(estimators list, voting='hard')
```

Ex: Ensemble Methods

Load the wine quality data from the following path

dataset_path = "https://raw.githubusercontent.com/tertiarycourses/datasets/master/winequality-red.csv"

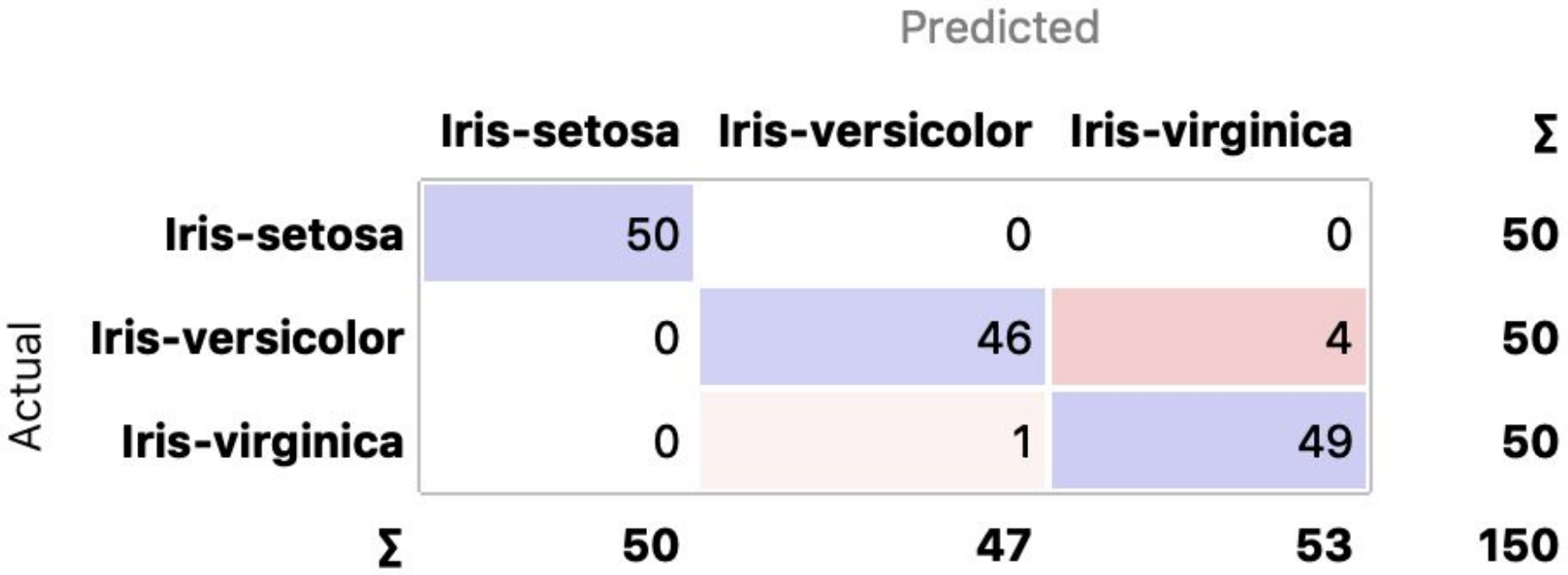
X = pd.read_csv(dataset_path,sep=';')

Determine which methods yield the best score

- Decision Tree
- Random Forest
- Gradient Boost
- Stacking

Confusion Matrix

- A confusion matrix is a table of predicted vs actual classes
- The diagonals are the correct classification
- The off-diagonals are the misclassifications.



Confusion Matrix

```
from sklearn.metrics import confusion_matrix

y_pred = clf.predict(X_test)
print(confusion_matrix(y_test,y_pred))
```

```
[[16  0  0]
 [ 0 10  1]
 [ 0  0 18]]
```

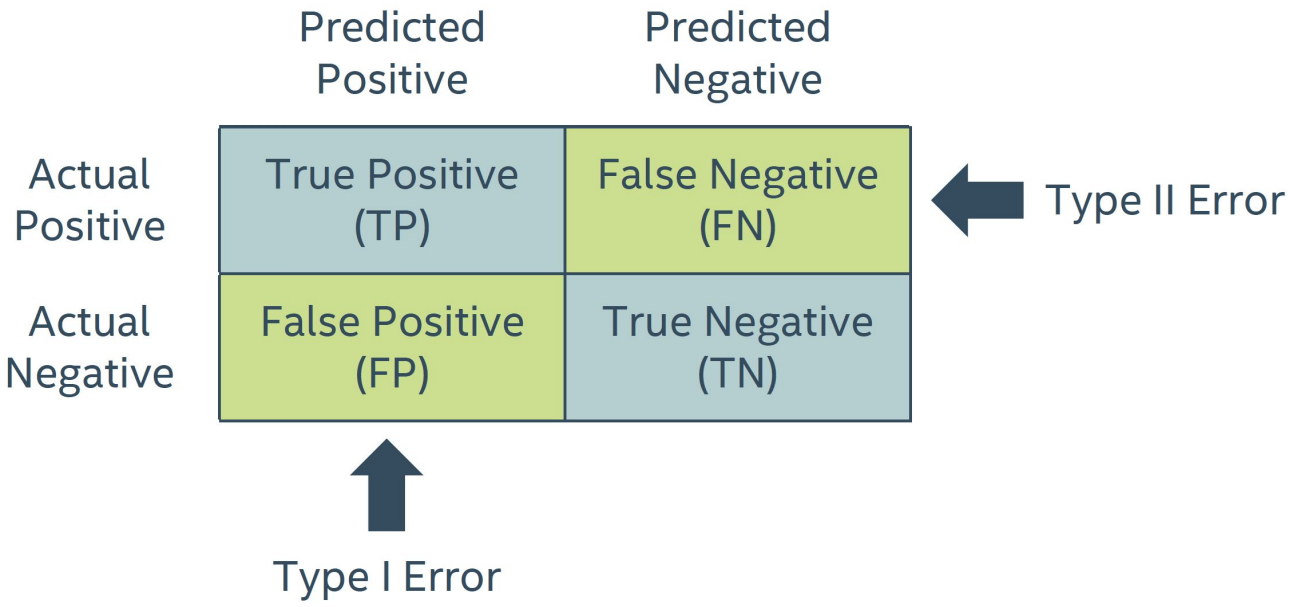
Ex: Confusion Matrix

Compare the confusion matrix for the red wine data for the following classifiers:

- KNN
- SVM

Binary Classification

- In many predictive analysis, we are interested in YES/NO analysis such as spam/not-spam, health/not-healthy etc.
- In this case, we can form a confusion matrix of 2 columns and 2 rows



Example of Type I and Type II Errors

Accuracy

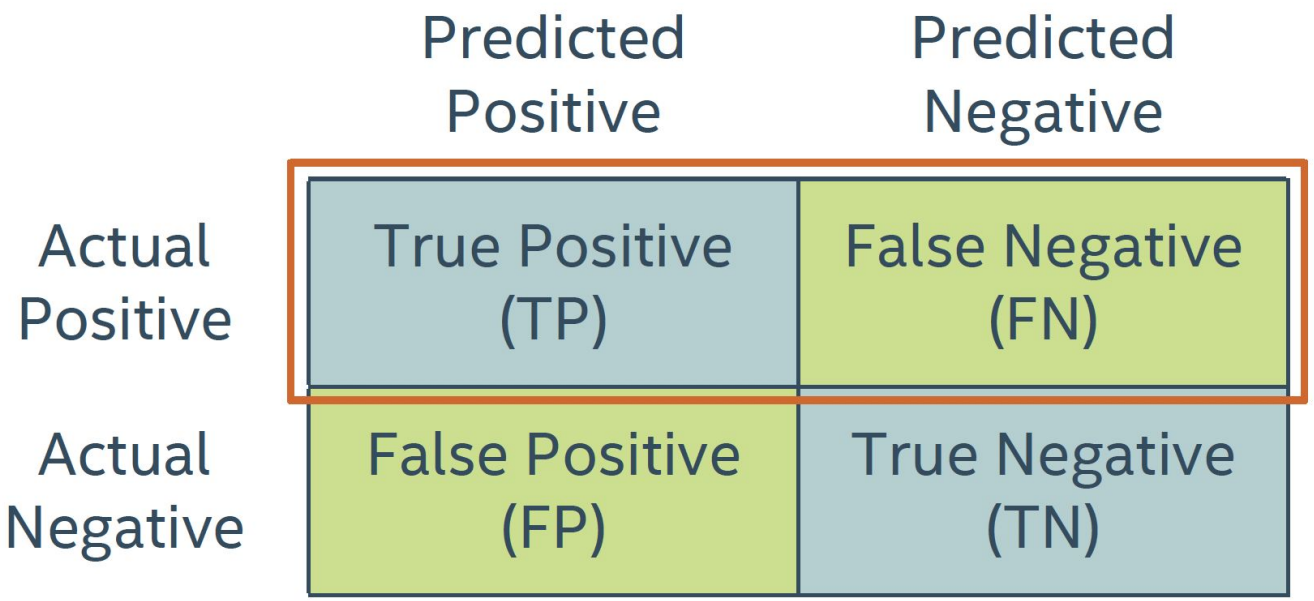
Accuracy is the percentage of correct hits.

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

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

Recall or Sensitivity

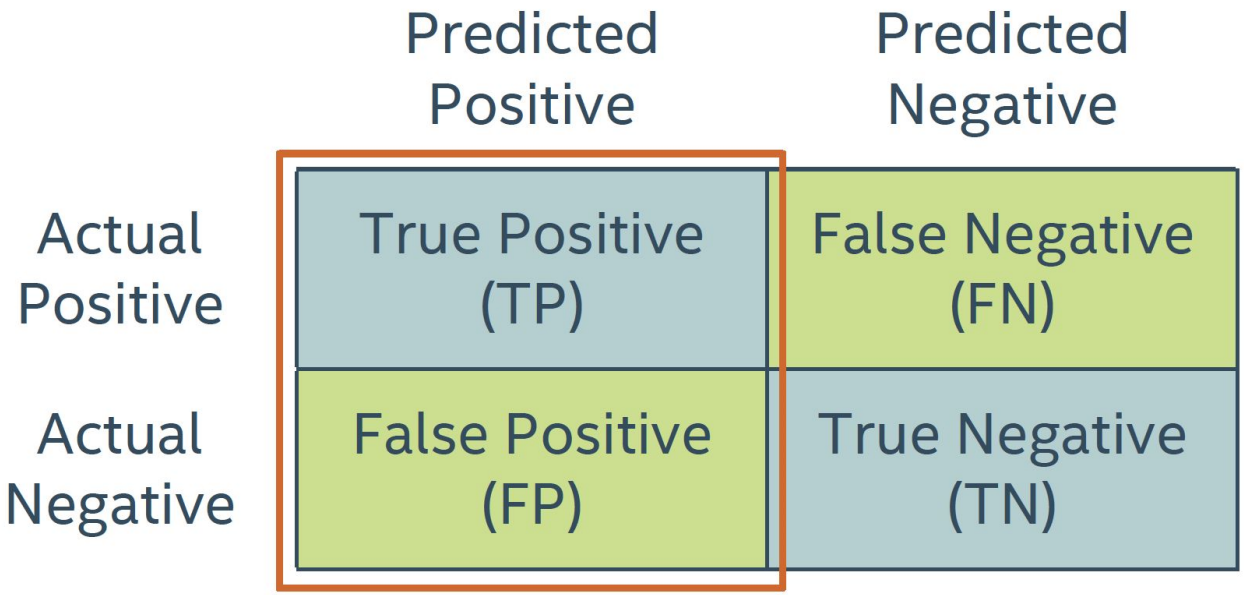
Recall or Sensitivity is to identify all the positive instances



$$\text{Recall or Sensitivity} = \frac{TP}{TP + FN}$$

Precision

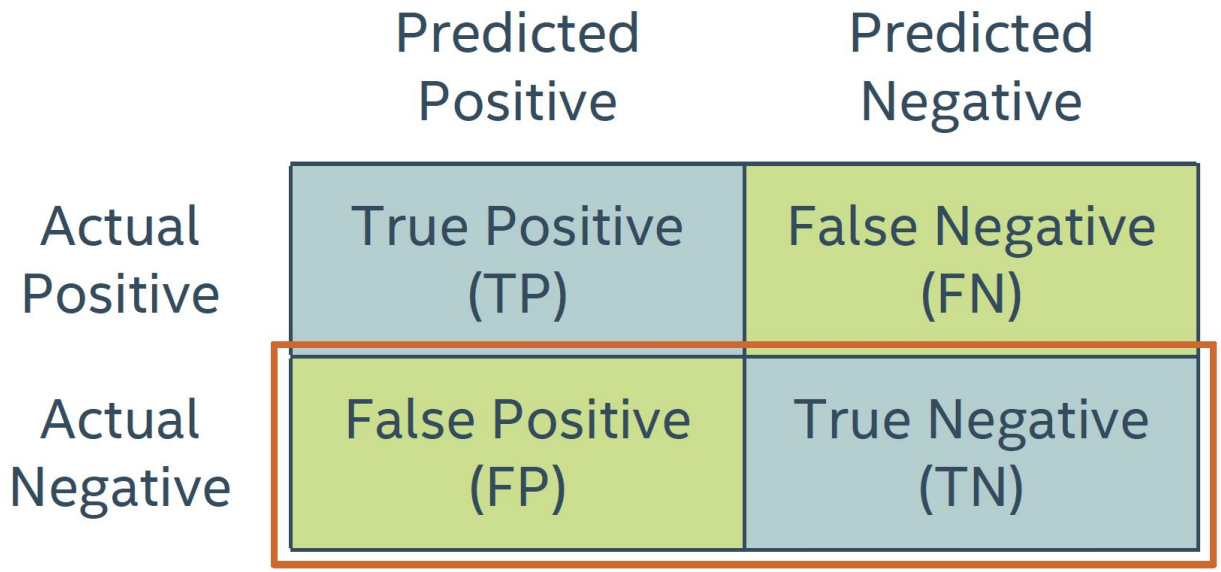
Precision is to identify only the positive instances



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

Specificity

Specificity is to identify only all the negative instances to detect false alarm.



$$\text{Specificity} = \frac{TN}{FP + TN}$$

F1 Score

- For error measurement of a particular model, need to check all the four metrics
- However, it is more convenient to check a single parameter using F1 score

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

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

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

$$\text{Recall or Sensitivity} = \frac{TP}{TP + FN}$$

$$\text{Specificity} = \frac{TN}{FP + TN}$$

$$F1 = 2\,\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Classification Metrics Summary

- Accuracy = (TP+TN)/(TP+FP+FN+TN)
- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)
- F1 Score = 2*(Recall * Precision) / (Recall + Precision)

The higher the scores the better is the classification model

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

Individual Metrics for Binary Classification

```
from sklearn.metrics import
accuracy_score,precision_score,recall_score,f1_score

from sklearn import neighbors
clf = neighbors.KNeighborsClassifier()
clf.fit(X_train,y_train)

y_pred = clf.predict(X_test)
print('Accuracy = ',accuracy_score(y_test,y_pred))
print('Precision = ',precision_score(y_test,y_pred))
print('Recall = ',recall_score(y_test,y_pred))
print('F1 Score = ',f1_score(y_test,y_pred))
```

Classification Report

Classification report will give you the summary of accuracy, precision, recall and f1 score

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	1.00	0.97	0.99	34
1	0.92	1.00	0.96	11
accuracy			0.98	45
macro avg	0.96	0.99	0.97	45
weighted avg	0.98	0.98	0.98	45

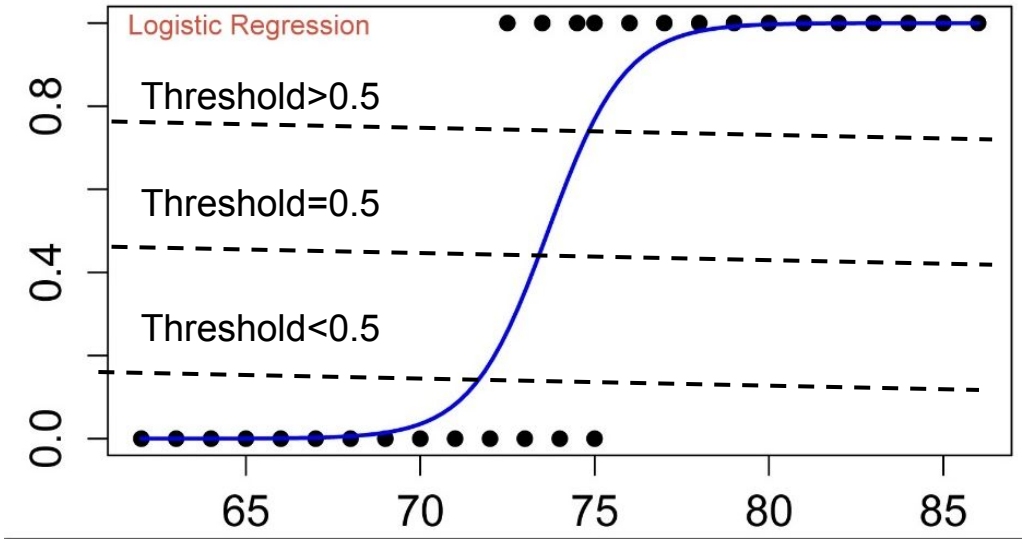
Ex: Metrics

Evaluate the individual accuracy, precision, recall F1 Score, as well as classification report for the red wine data for the following classifiers:

- KNN
- SVM

Adjusting Binary Classification Threshold

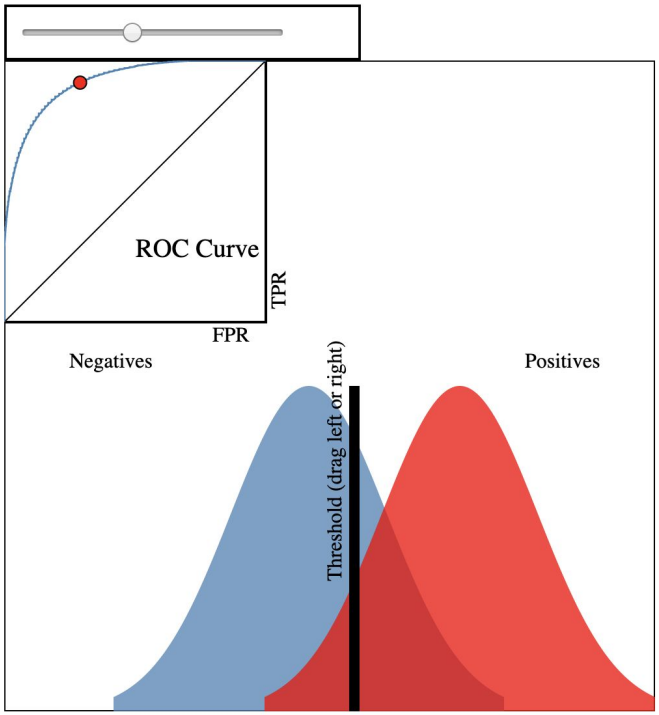
- Adjusting the threshold value of binary classification will give rise to different confusion matrix - different FP and FN values.
- For example, if the threshold value is lower than 0.5, TP and FP rate will increase.



		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

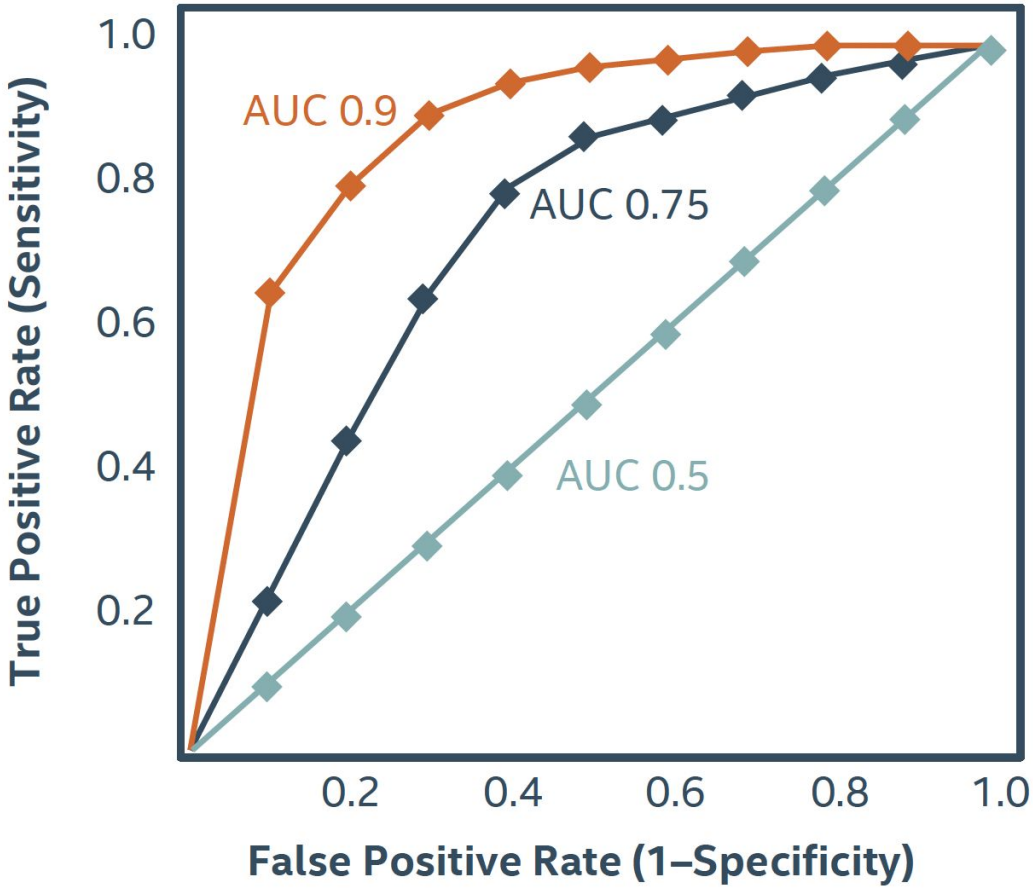
Receiver Operating Characteristics (ROC)

- Receiver Operating Characteristics (ROC) can be used to evaluate all the possible thresholds
- ROC is a plot of TP rate(sensitivity) vs FP rate (1-specificity)
- Demo: http://www.navan.name/roc/



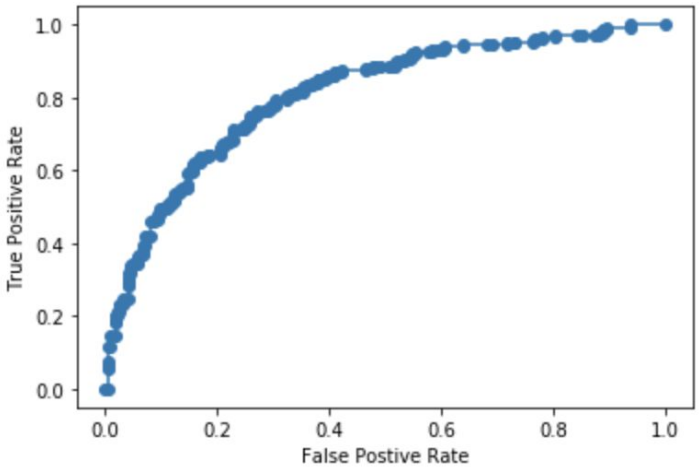
Area Under Curve (AUC)

- AUC is the area under the ROC curve
- AUC ranges in value from 0 to 1.
- The higher AUC value is preferred for a classifier



ROC and AUC

```
from sklearn.metrics import roc_curve,roc_auc_score
from sklearn.linear_model import LogisticRegression
clf = LogisticRegression()
clf.fit(X_train,y_train)

# predict probabilities
y_probs = clf.predict_proba(X_test)
y_probs = y_probs[:, 1]

auc_score = roc_auc_score(y_test, y_probs)
print("AUC = ", auc_score)
fpr, tpr, threshold = roc_curve(y_test,y_probs)
```



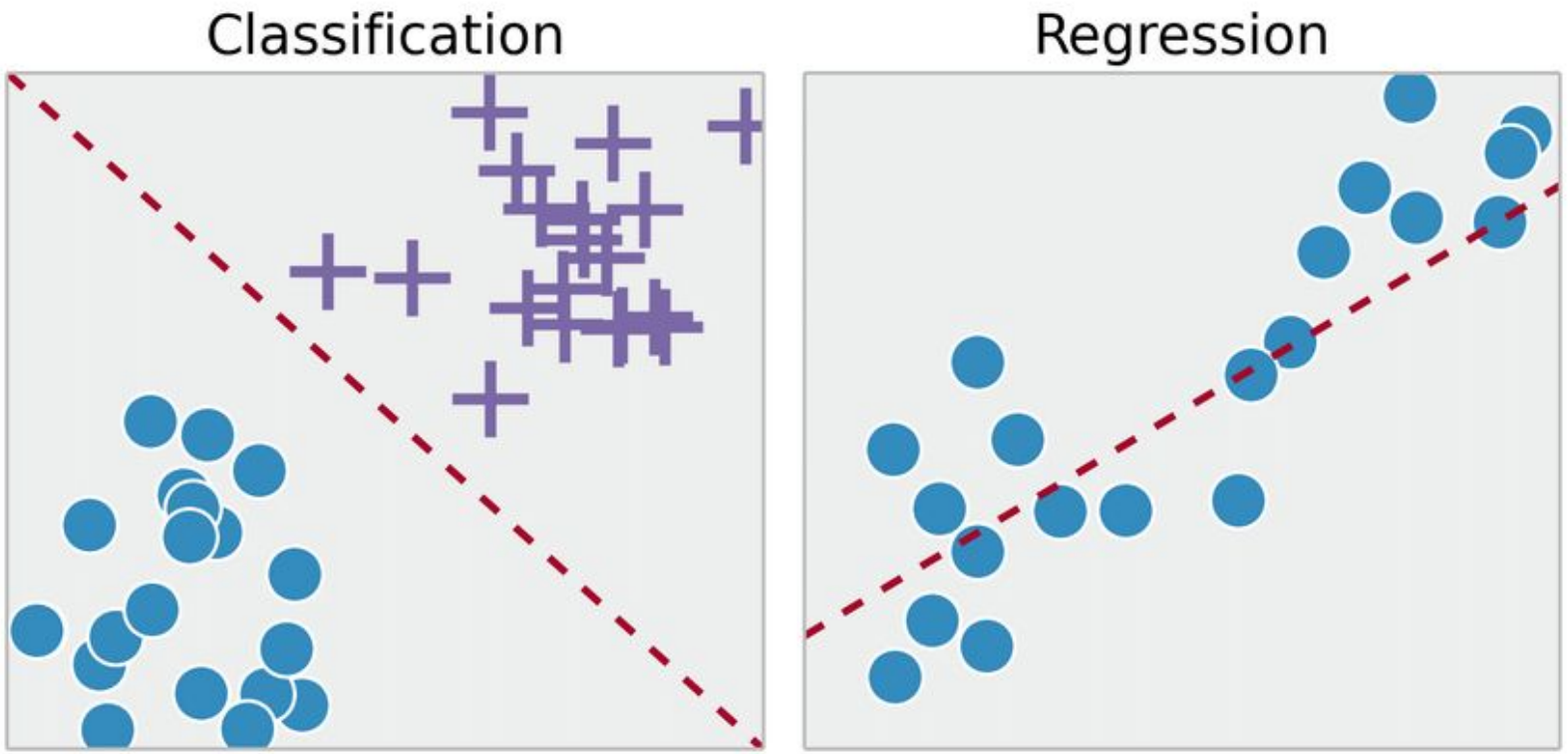
Ex: ROC and AUC

Compute the ROC and AUC for the red wine data using logistics classifer

Topic 3
Regression

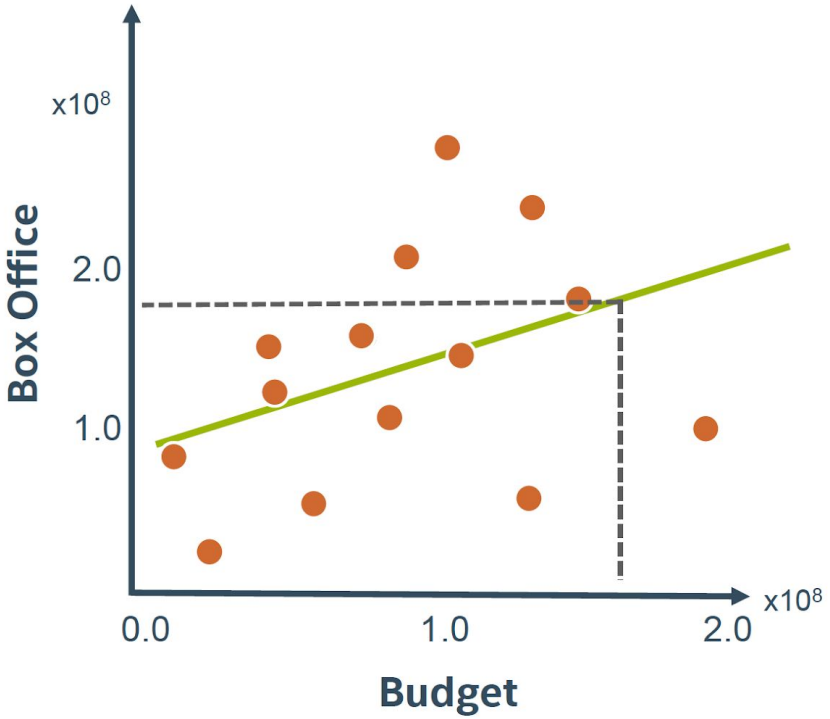
What is Regression?

- Regression it to predicting a continuous-valued attribute associated with an object.
- Regression is a supervised learning method. During training, actual values are provided.



Linear Regression

- Linear regression is the most common regression model. Many predictive models use linear regression models
- You can use a linear regression model to predict the box office from the budget.



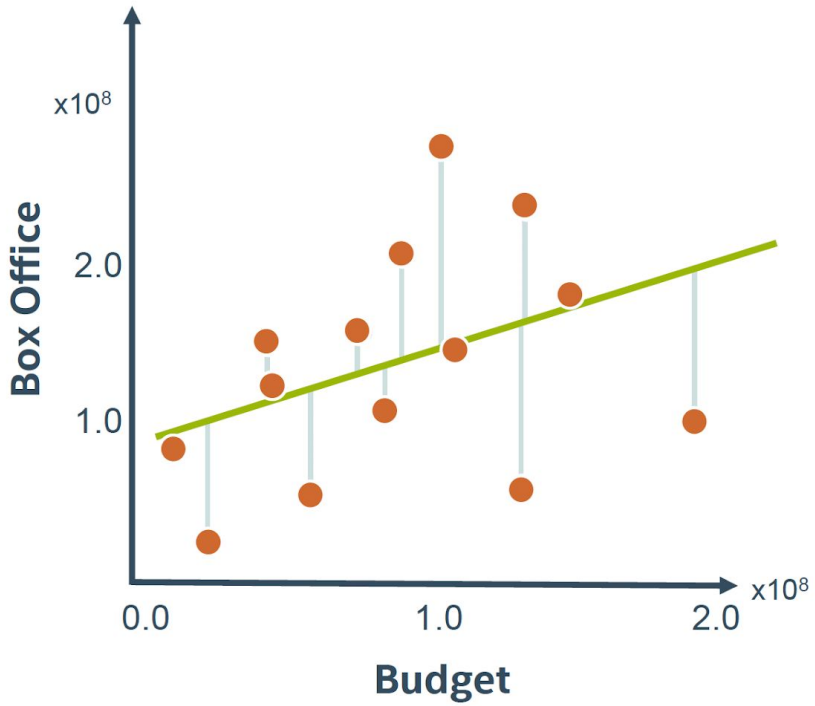
$$y_\beta(x) = \beta_0 + \beta_1 x$$

β_0= 80 million, β_1= 0.6

Predict 175 Million Gross for 160 Million Budget

Residues

- Residue is the difference between the predicted value and actual value



$$y_\beta\left(x_{obs}^{(i)}\right) - y_{obs}^{(i)}$$

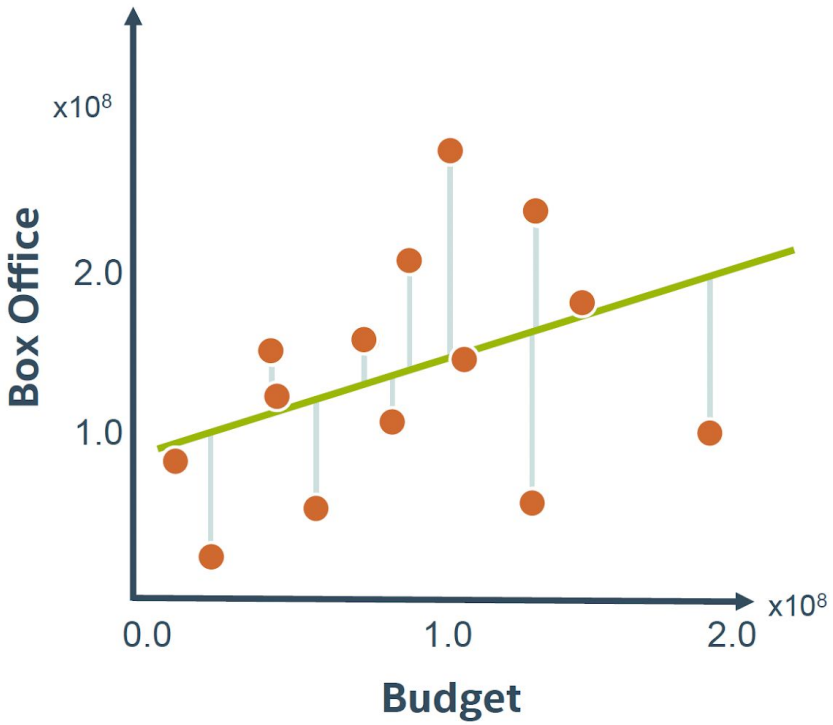
Predicted value

Observed value

$$\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}$$

Mean Square Error

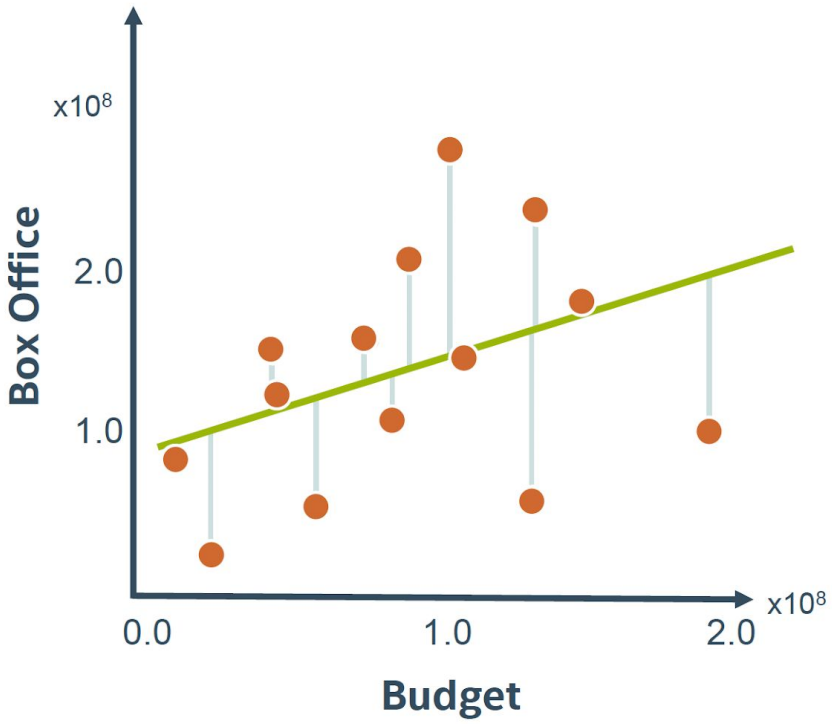
- Mean Square Error (MSE) is the common loss function to measure how good is the linear regression model.



$$\frac{1}{m}\sum_{i=1}^{m}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2$$

Minimum Mean Square Error

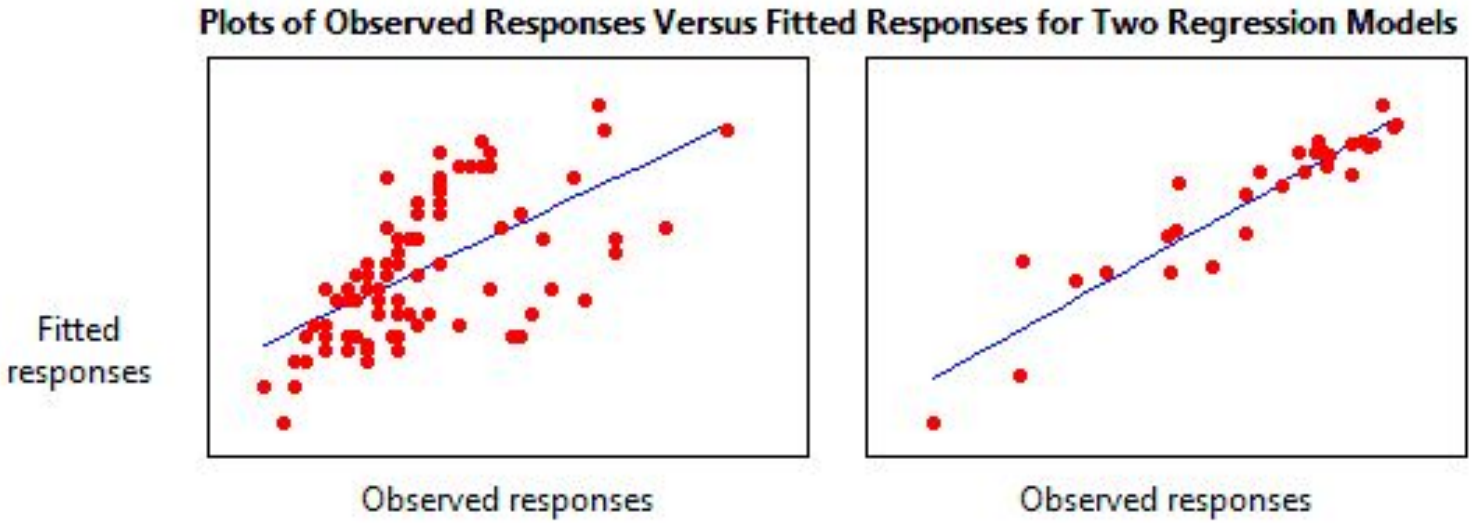
- Machine Learning aims to minimize the MSE to find the best linear regression model.



$$\min_{\beta_0,\beta_1} \frac{1}{m} \sum_{i=1}^{m} \left(\left(\beta_0 + \beta_1 x_{obs}^{(i)} \right) - y_{obs}^{(i)} \right)^2$$

R Square (Goodness Of Fit)

- R-squared is a statistical measure of how close the data are to the fitted regression line.
- R-squared = Explained variation / Total variation
- R-squared is always between 0 and 1
  - 0 indicates that the model explains none of the variability of the response data around its mean.
  - 1 indicates that the model explains all the variability of the response data around its mean.



Demo: Build Predictive Regression Model

This is a demo of how to build a predictive regression model with linear. The steps are as follows:

- **Prepare the data**
  - Import raw data
  - Clean data eg remove missing data
  - Normalize the data
  - Split raw data to train and test dataset
  - Create input and output
- **Build the model**
  - Create a linear regression model
- **Train the model**
- **Evaluate the model**
- **Make prediction**

Boston Housing Price Dataset

There are 13 features for this dataset.

- CRIM — per capita crime rate by town
- ZN — proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS — proportion of non-retail business acres per town
- CHAS — Charles River dummy variable
- NOX — nitric oxides concentration (parts per 10 million)
- RM — average number of rooms per dwelling
- AGE — proportion of owner-occupied units built prior to 1940
- DIS — weighted distances to five Boston employment centres
- RAD — index of accessibility to radial highways
- TAX — full-value property-tax rate per $10,000
- PTRATIO — pupil-teacher ratio by town
- B — 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
- LSTAT — % lower status of the population
- **MEDV** — **Median value of owner-occupied homes in $1000's**

Import the Data

We will get some raw data from github Eg boston hosting price

import pandas as pd

dataset_path = "https://raw.githubusercontent.com/selva86/datasets/master/BostonHousing.csv"

Prepare the Data for Training

- Remove missing data

```
X = X.dropna()
```

- Create input and output

```
y = X.pop('medv')
```

Normalize/Scale the Input Data

```
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

Scale the Input Data

```
from sklearn.model_selection import train_test_split


X_train,X_test,y_train,y_test =
train_test_split(X_scaled,y,test_size=0.3,random_state
=100)
```

Define the Linear Regression Model

```
from sklearn import linear_model

lm = linear_model.LinearRegression()
```

Train the Model

```
lm.fit(X_train,y_train)
```

Model Metrics

Sum of Squared Error (SSE):

$$\sum_{i=1}^{m} \left(y_\beta(x^{(i)}) - y_{obs}^{(i)} \right)^2$$

Total Sum of Squares (TSS):

$$\sum_{i=1}^{m} \left(\overline{y_{obs}} - y_{obs}^{(i)} \right)^2$$

Correlation Coefficient (R2):

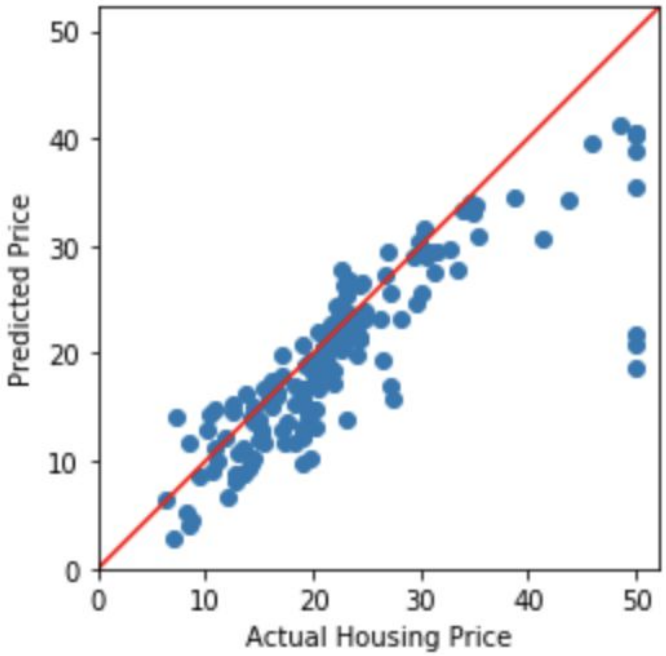
$$1 - \frac{SSE}{TSS}$$

Model Metrics

```
from sklearn.metrics import mean_squared_error, r2_score

yhat = lm.predict(X_train)
mse = mean_squared_error(y_train,yhat)
print('Mean Squared Error, Training: ',mse)
rsq = r2_score(y_train,yhat)
print('R-square, Training: ',rsq)

yhat = lm.predict(X_test)
mse = mean_squared_error(y_test,yhat)
print('Mean Squared Error, Testing: ',mse)
rsq = r2_score(y_test,yhat)
print('R-square, Testing: ',rsq)
```

Evaluate the Model

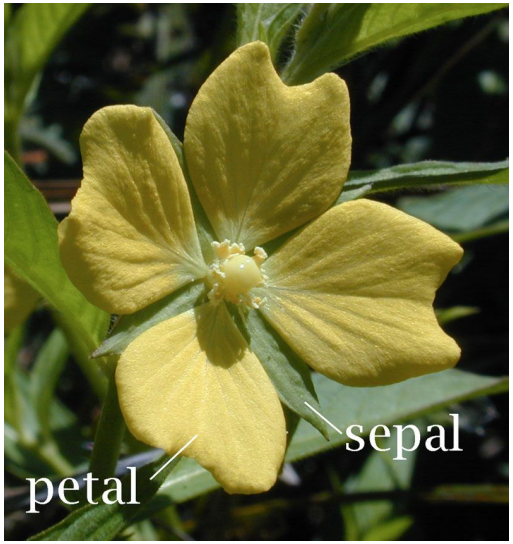
```
%matplotlib inline
import matplotlib.pyplot as plt

yhat = lm.predict(X_test)
plt.scatter(y_test,yhat)
plt.xlabel('Actual Housing Price')
plt.ylabel('Predicted Price')
plt.plot([0, 50], [0, 50],'r')
```



Ex: Predictive Regression Model

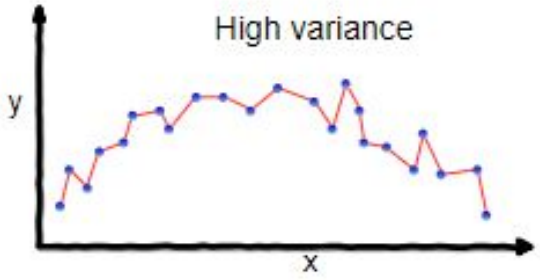
- Import the iris.dataset from github
  https://raw.githubusercontent.com/pandas-dev/pandas/master/pandas/tests/data/iris.csv"
- Create a linear regression model to predict the sepal width from sepal length, petal length and petal width
- Perform all the steps as shown in the demo
  - Prepare the data
  - Build the model
  - Train the model
  - Make prediction
  - Save the model

Time: 30 mins

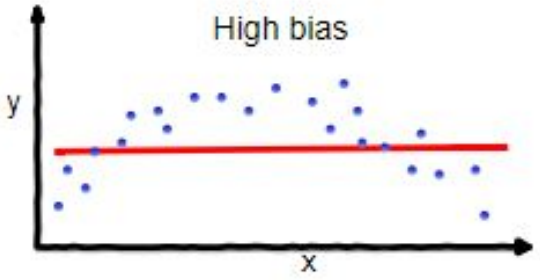


Bias and Variance

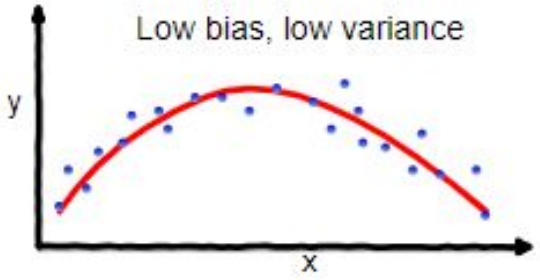
- Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias pays very little attention to the training data and oversimplifies the model. It always leads to high error on training and test data.
- Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance pays a lot of attention to training data and does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.



overfitting



underfitting



Good balance

Bias-Variance Trade Off

- The expected squared error Err(x) at a point x is

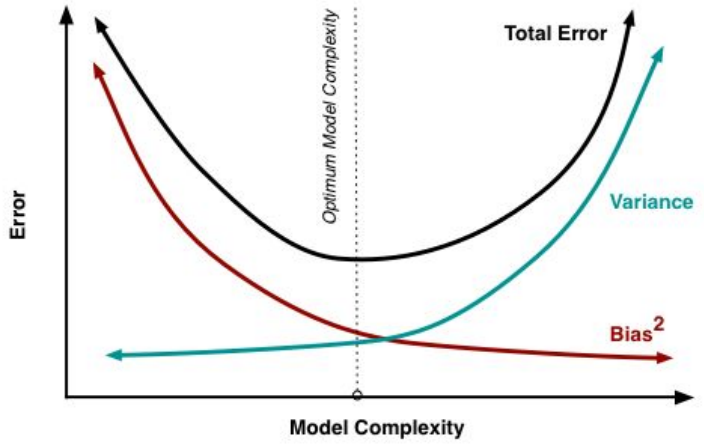
$$Err(x) = E\left[(Y - \hat{f}\,(x))^2\right]$$

- The Err(x) can be further decomposed as

$$Err(x) = \left(E[\hat{f}\,(x)] - f(x)\right)^2 + E\left[\left(\hat{f}\,(x) - E[\hat{f}\,(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

- If our model is too simple and has very few parameters then it may have high bias and low variance. On the other hand if our model has large number of parameters then it's going to have high variance and low bias. So we need to find the right/good balance without overfitting and underfitting the data.



Regularization Methods

- There are 3 regularization techniques
- Regularizations tend to reduce (ridge) or remove (lasso) the dependences of the features/variables.

Ridge Regularization

$$J(\beta_0, \beta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2 + \lambda\sum_{j=1}^{k}{\beta_j}^2$$

Lasso Regularization

$$J(\beta_0, \beta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2 + \lambda\sum_{j=1}^{k}|\beta_j|$$

Elastic Net Regularization

$$J(\beta_0, \beta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(\left(\beta_0 + \beta_1 x_{obs}^{(i)}\right) - y_{obs}^{(i)}\right)^2 + \lambda_1\sum_{j=1}^{k}|\beta_j| + \lambda_2\sum_{j=1}^{k}\beta_j^2$$

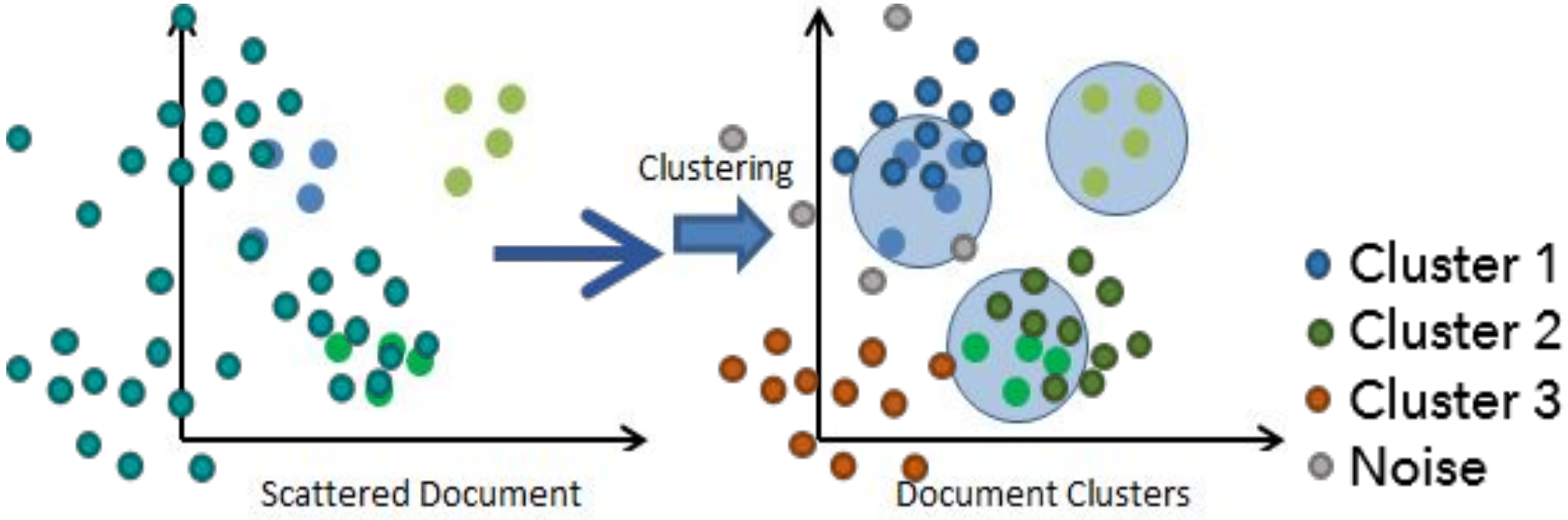
Regularizations in Scikit Learn

```
from sklearn.linear_model import Ridge, Lasso, ElasticNet
rr = Ridge(alpha=0.01)
lr = Lasso(alpha=0.01)
er = ElasticNet(alpha=0.01,l1_ratio=0.5)

rr.fit(X_train,y_train)
lr.fit(X_train,y_train)
er.fit(X_train,y_train)
```

Topic 4
Clustering

What is Clustering?

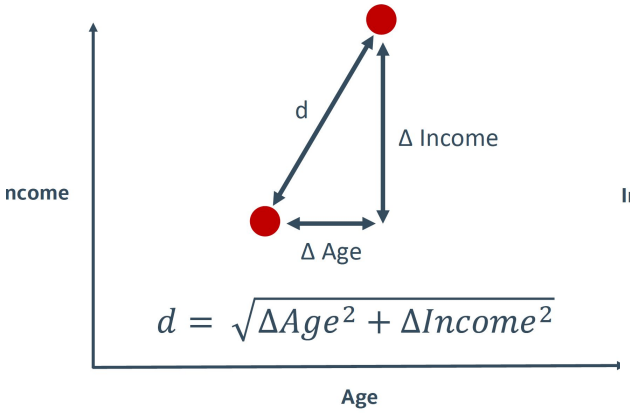
- Clustering is the task of grouping a set of objects in such a way that objects in the same group (cluster) are more similar to each other than to those in other groups (clusters).
- Clustering is a unsupervised learning since no labels (targets) are needed for training.



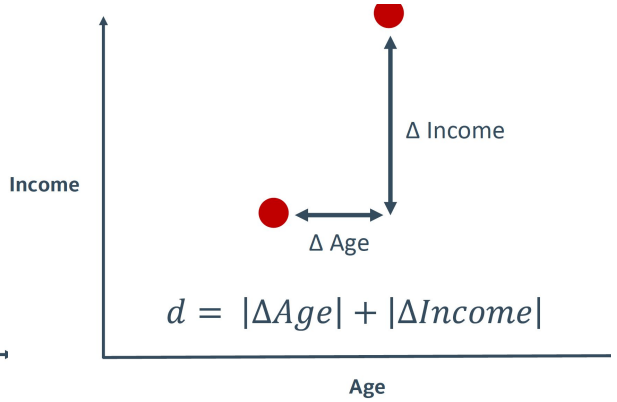
Distance Metric Choice

Choice of distance will significantly impact the clustering algorithms

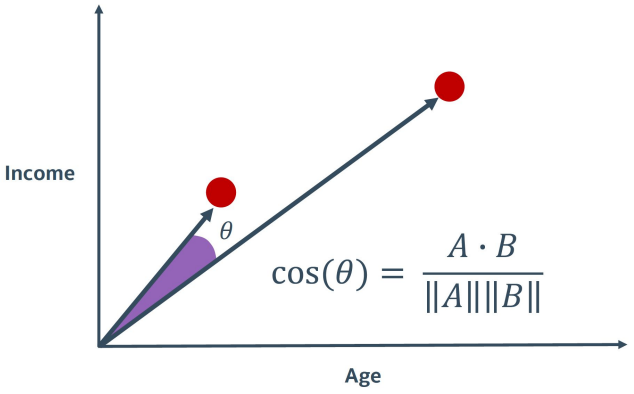
Euclidean (L2)



Manhattan (L1)



Cosine



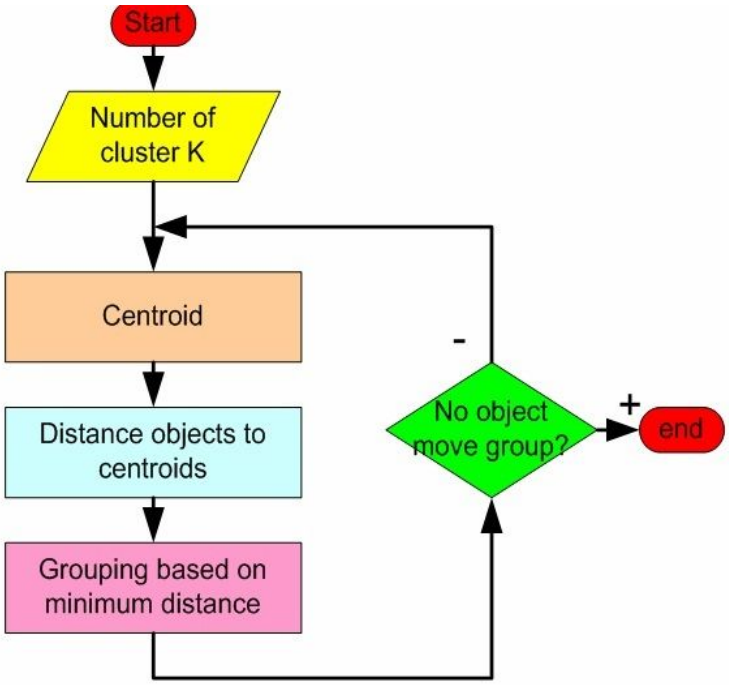
Mainly for text

Key Clustering Algorithms

- K-Means Clustering
- Hierarchical Agglomerative Clustering

K-Means Algorithm

- Kmeans tries to partition the dataset into *K*pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to **only one group**. It tries to make the inter-cluster data points as similar as possible while also keeping the clusters as different (far) as possible.
- The clusters are grouped around centroids, causing them to be globular and have similar sizes.



Parameters are estimated by the model
Hyper-parameters is something you set before running the model

K Mean Algorithm Simulation

Click on the images below to start k-mean clustering simulation (k=4)

Seeds starts on left

Seeds randomly assigned

Limitations of K-Means Clustering

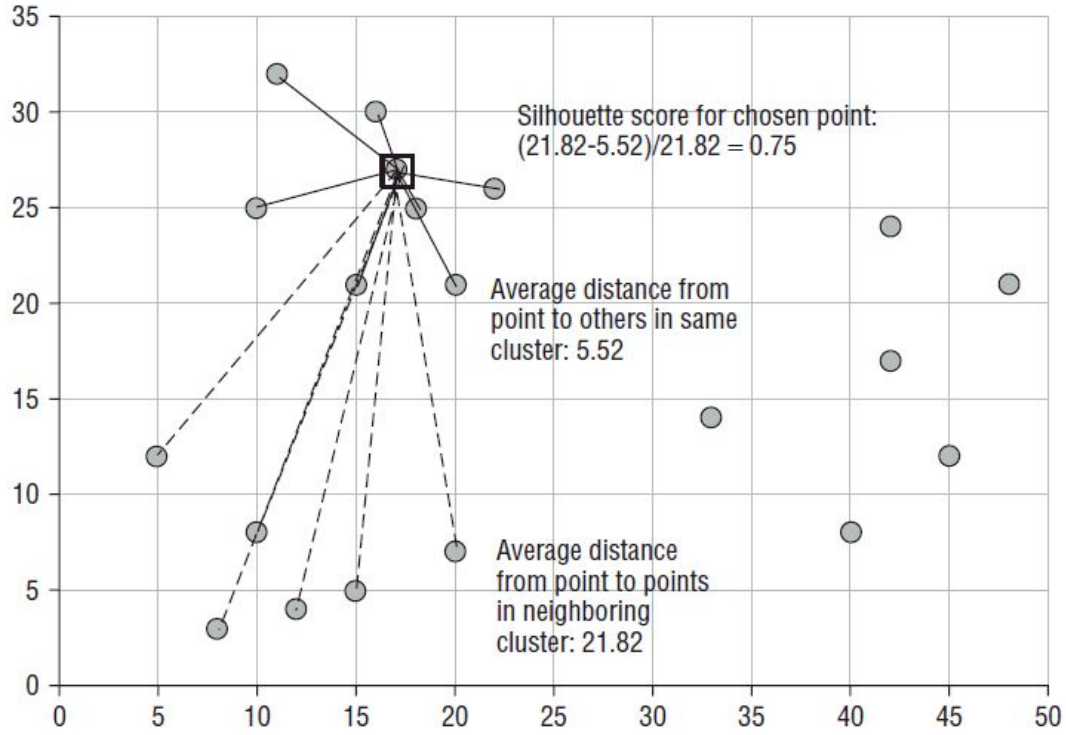
- Generally good for compact spherical clusters. Not suitable for clusters that are spread out
- The clustering is sensitive to initial centroid locations

Silhouette Analysis

- Silhouette analysis refers to a method of interpretation and validation of consistency within clusters of data.
- The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
- It can be used to study the separation distance between the resulting clusters
- If the Silhouette index value is high, the object is well-matched to its own cluster and poorly matched to neighbouring clusters.

Silhouette Score

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b)



How to Choose K

this is hyper-parameter tuning

```
from sklearn.metrics import silhouette_score
for i in range(2,10):
    cluster = KMeans(n_clusters=i,random_state=10)
    cluster.fit(X)
    s = silhouette_score(X, cluster.labels_)
    print('Cluster: ',i, 'Silhouette Score :',s )
```

K-Means Clustering Steps

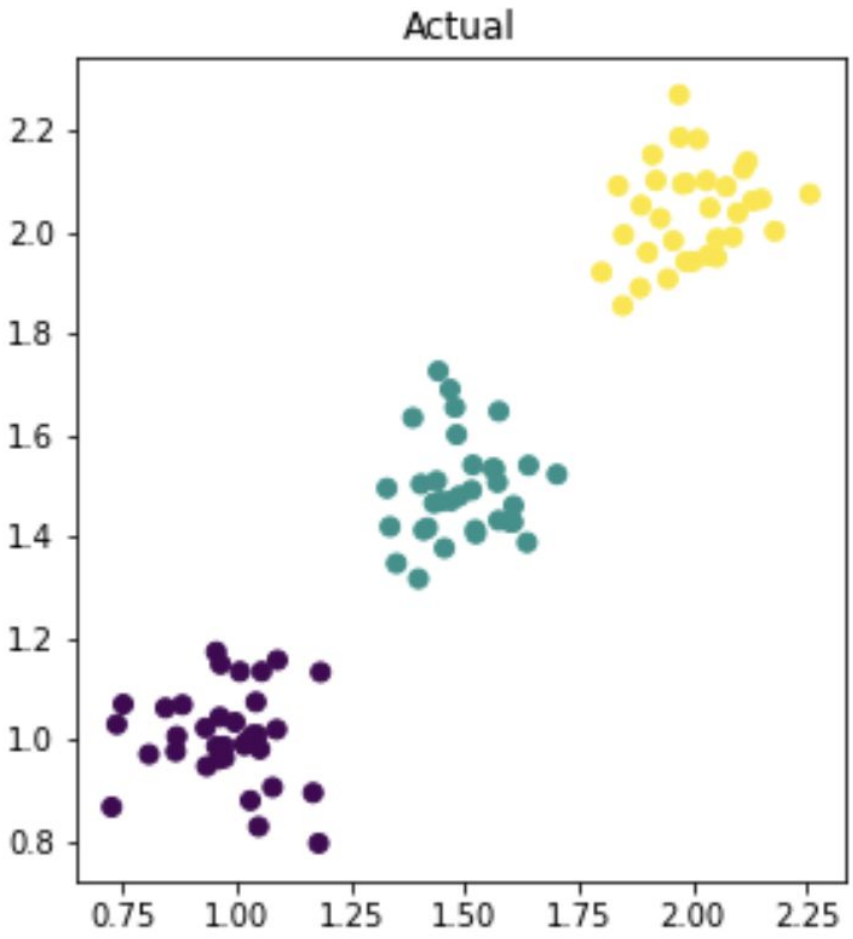
```
# Step 1 Model
from sklearn import cluster
cluster = cluster.KMeans(n_clusters=2)

# Step 2 Training
cluster .fit(X)

# Step 3 Evaluation
plt.scatter(X[:,0],X[:,1],c=cluster.labels_)
plt.show()
```

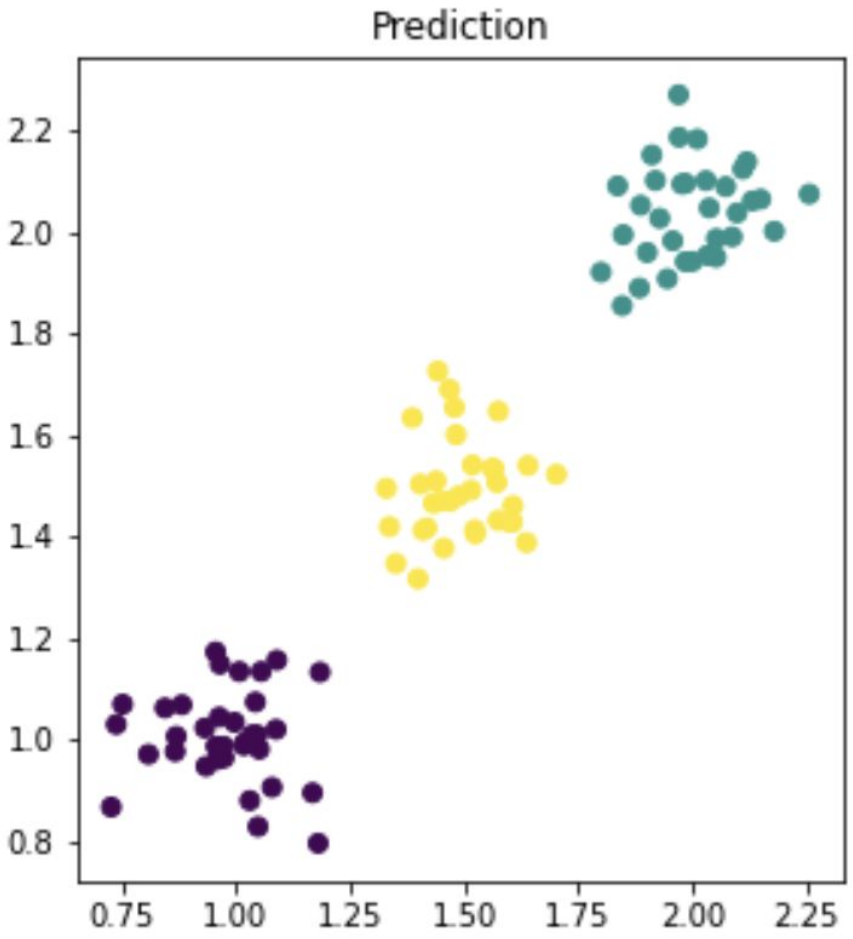
Clustering Output

cluster_centers_: Coordinates of cluster centers
labels_ : Labels of each point

K- Means Clustering Demo

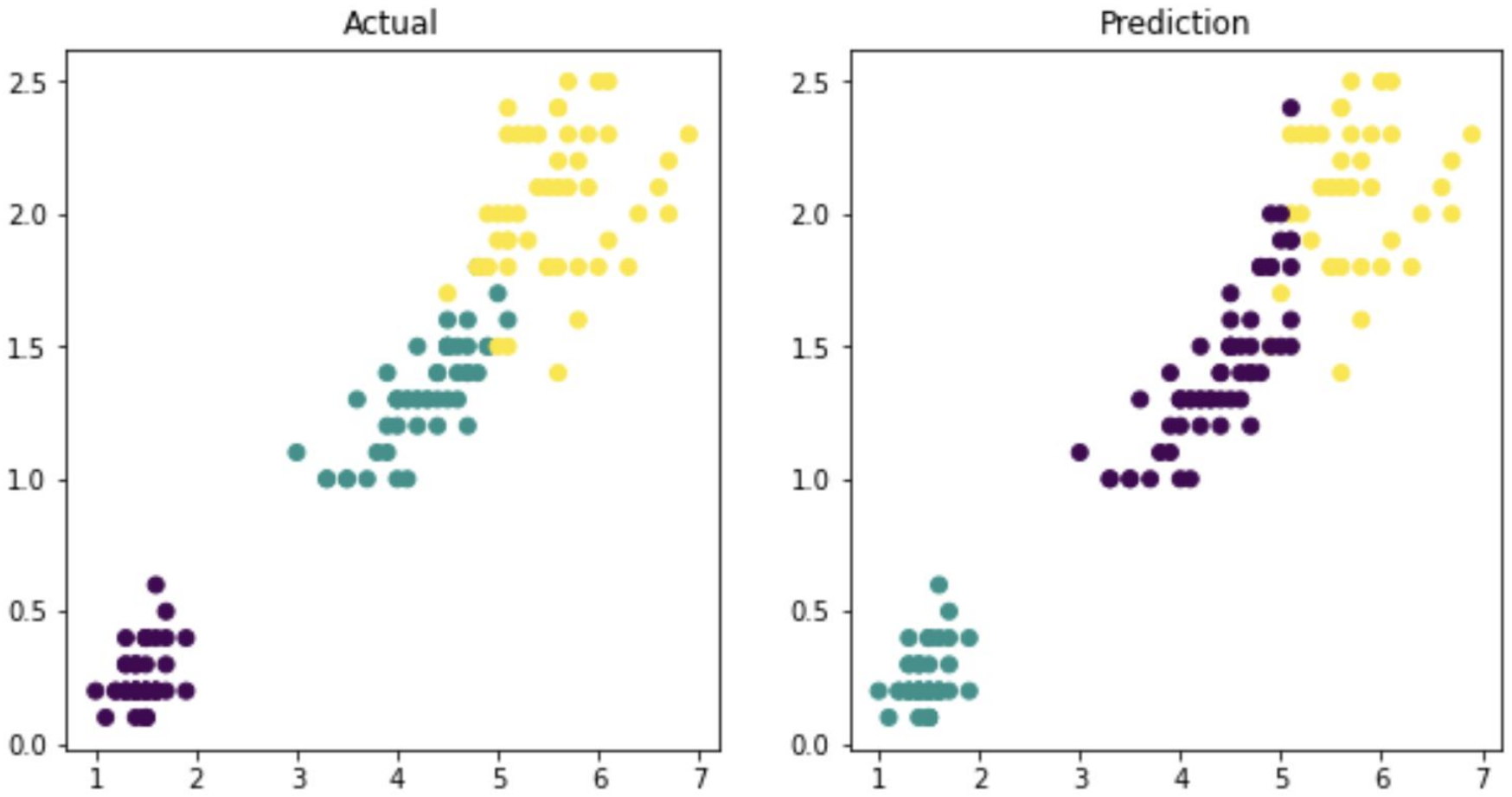
- Blob generator dataset





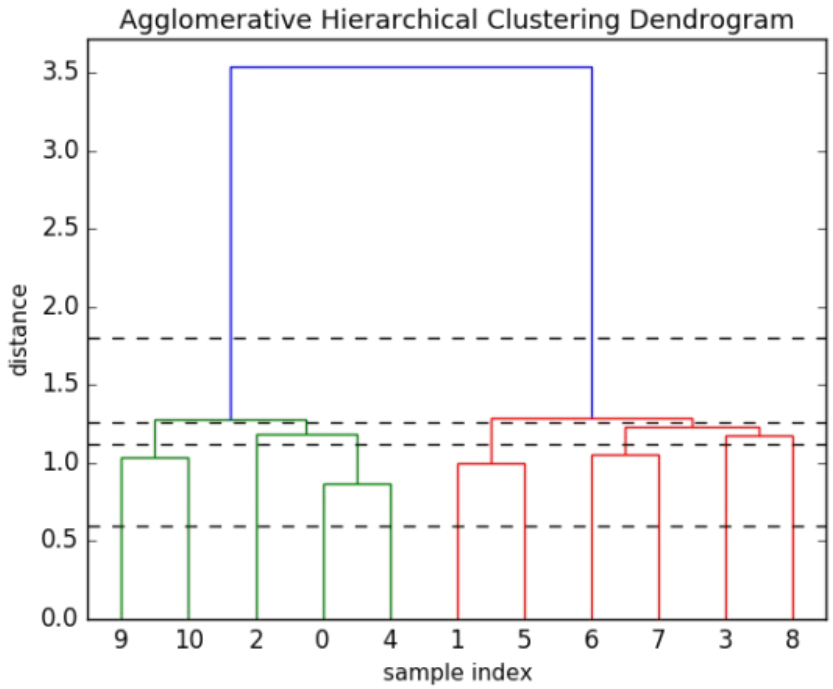
Ex: K-Means Clustering

Apply K-Means Clustering to iris dataset



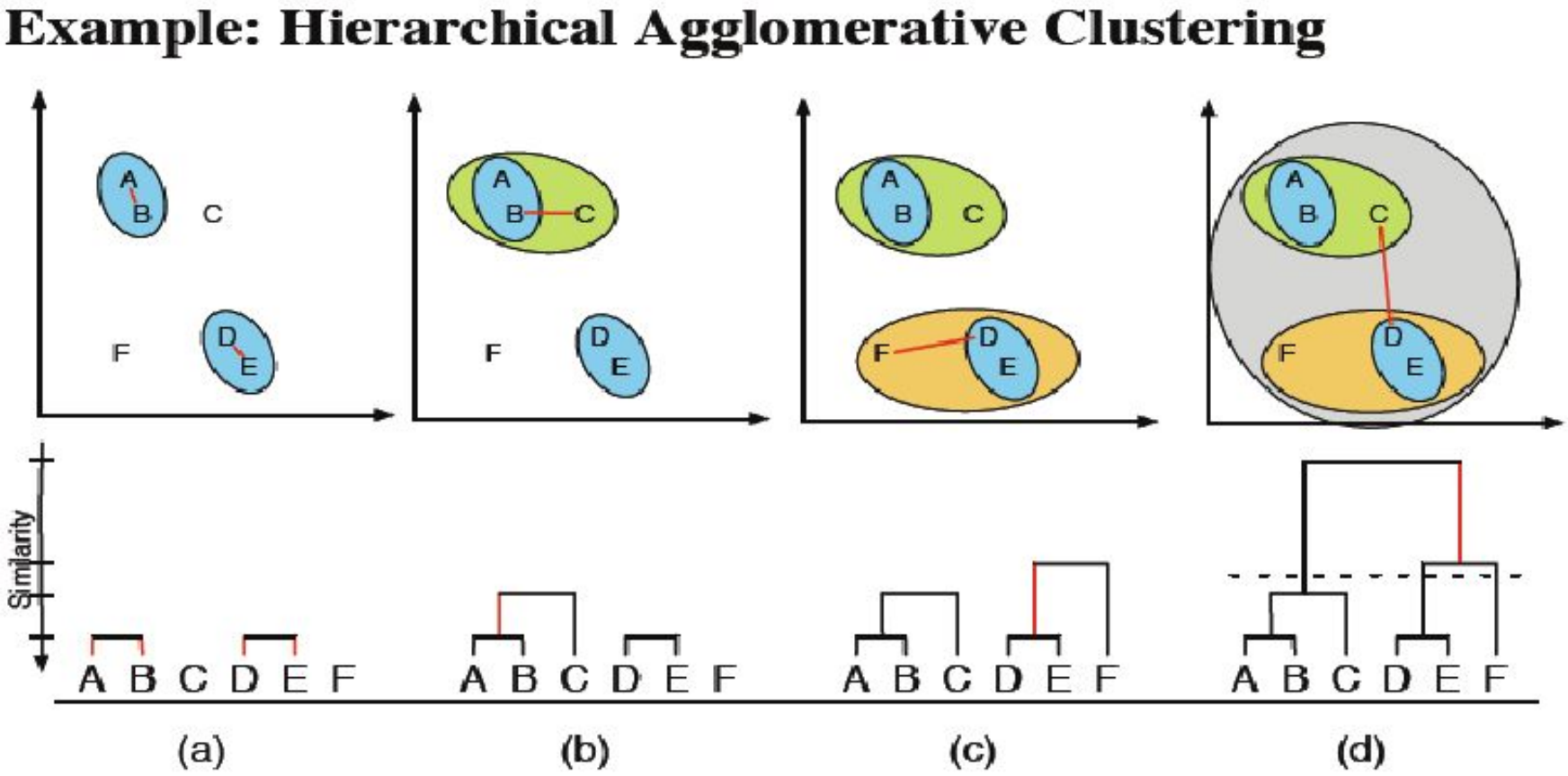
Hierarchical Clustering

- Hierarchical clustering is where you build a cluster tree (a dendrogram) to represent data, where each group (or “node”) links to two or more successor groups.
- The groups are nested and organized as a tree, which ideally ends up as a meaningful classification scheme.



Hierarchical Clustering

Clusters are consecutively merged with the most nearby clusters. The length of the vertical dendogram lines (linkage) reflect the nearness



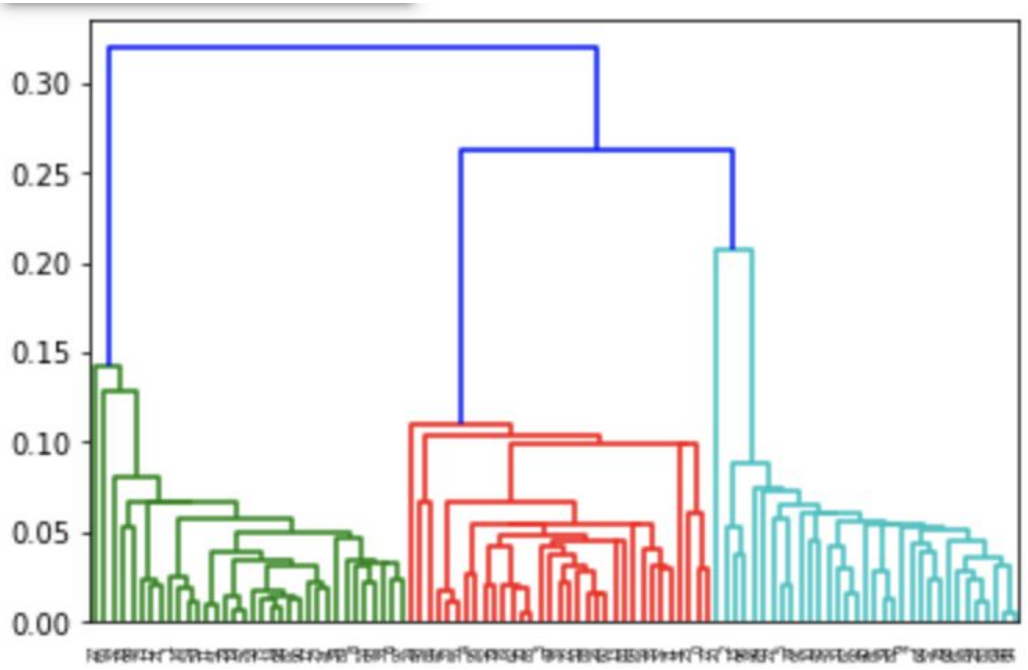
Dendrogram

from scipy.cluster.hierarchy import dendrogram, linkage

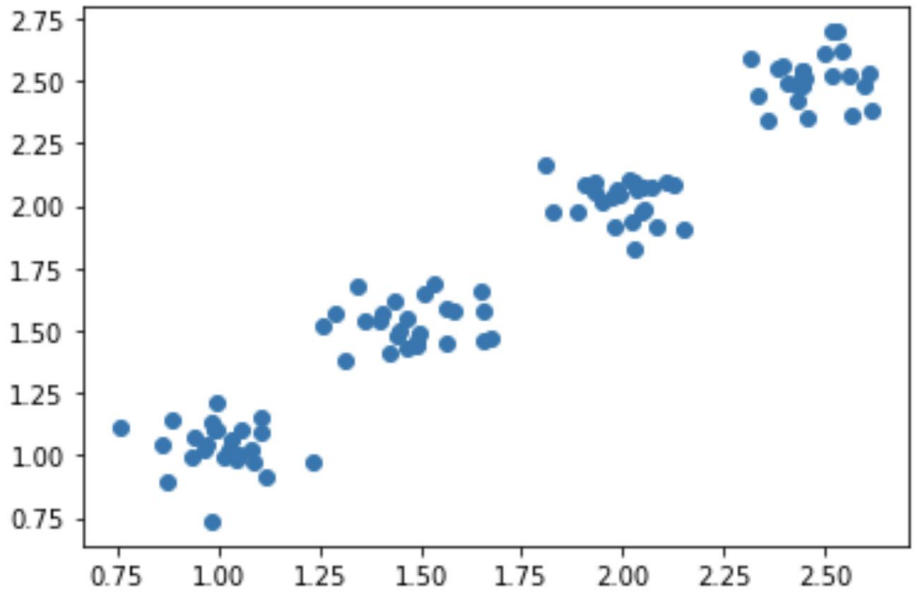
Z = linkage(X)

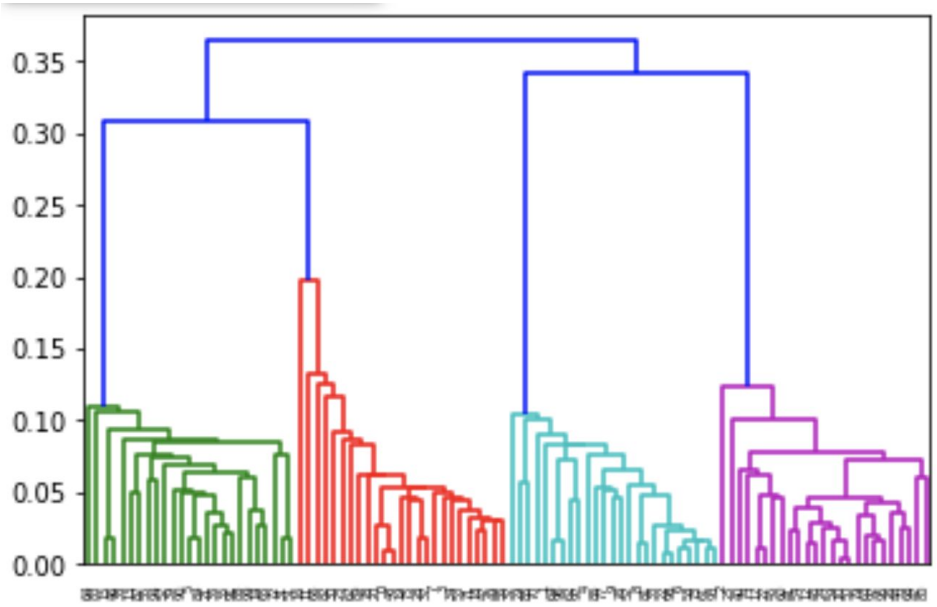
d = dendrogram(Z)

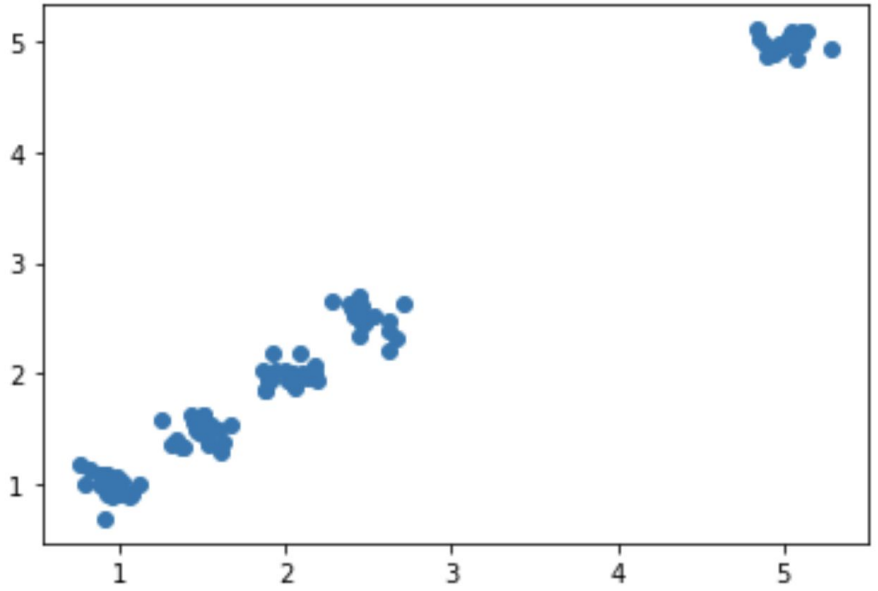
plt.plot(d)

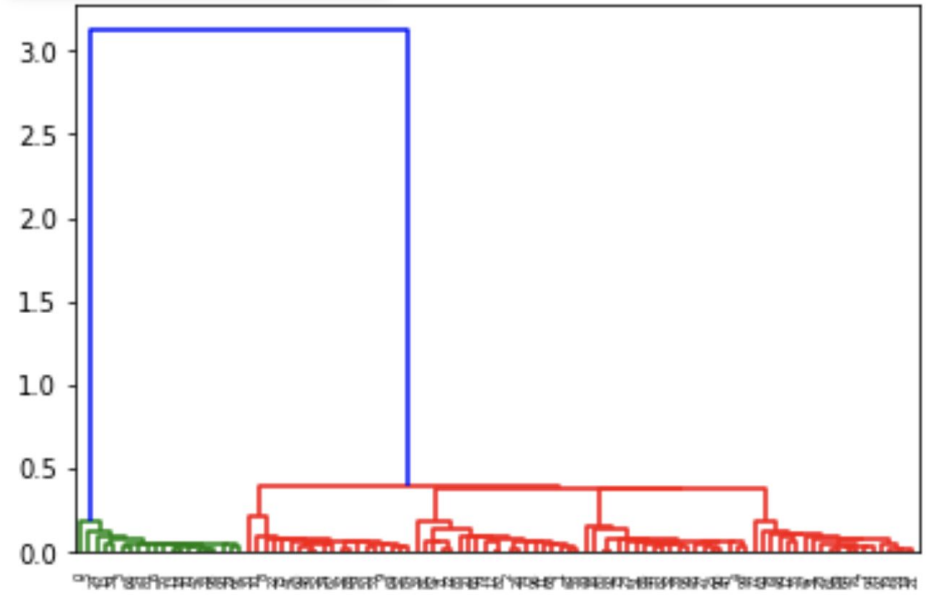


Dendrogram Demo



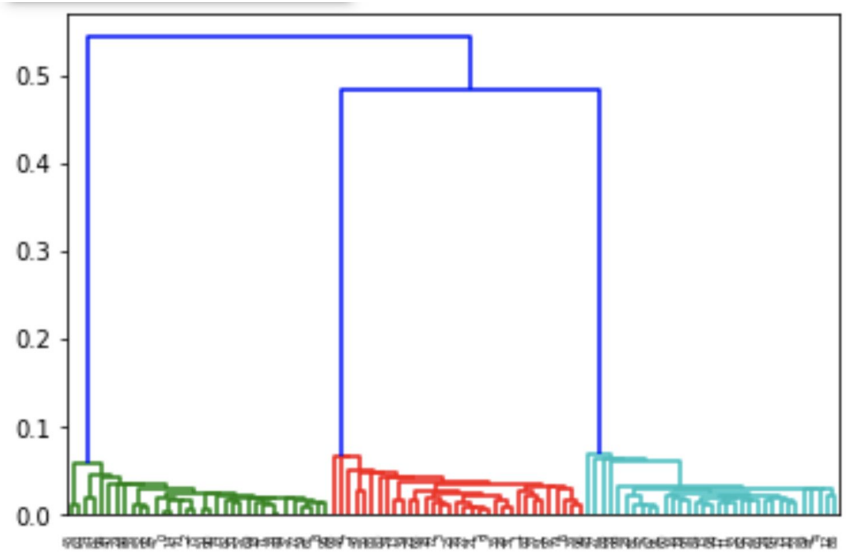


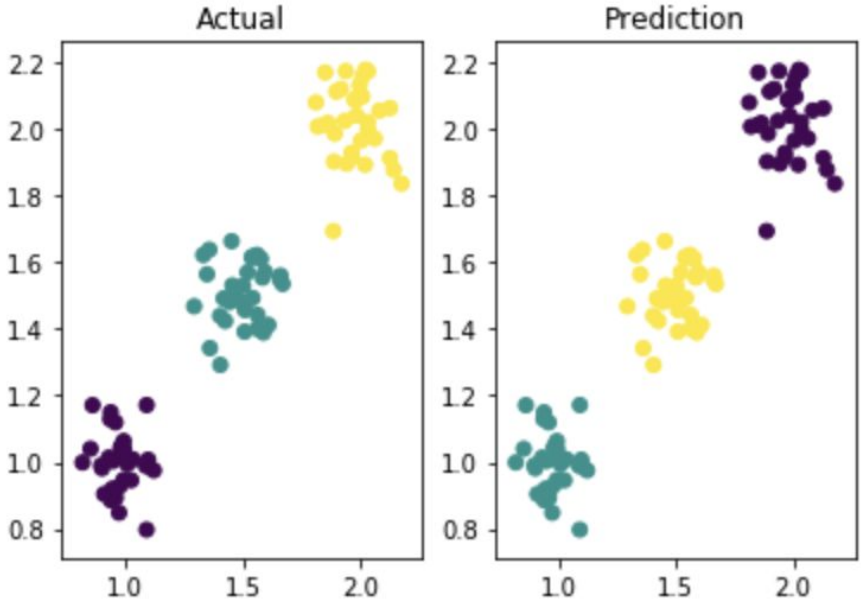




Hierarchical Clustering

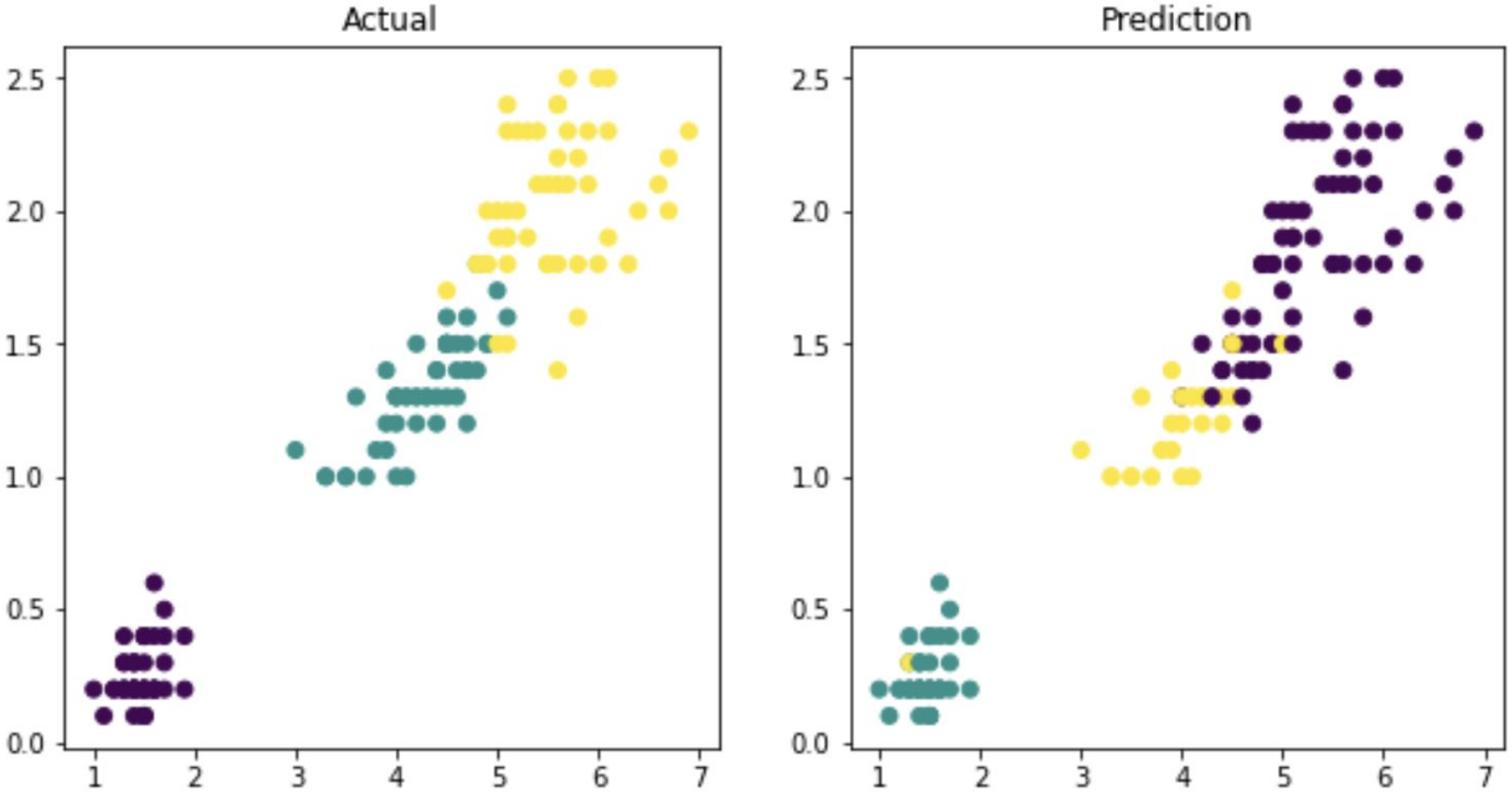
```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters = 3)
cluster.fit(X)
```





Ex: Hierarchical Clustering

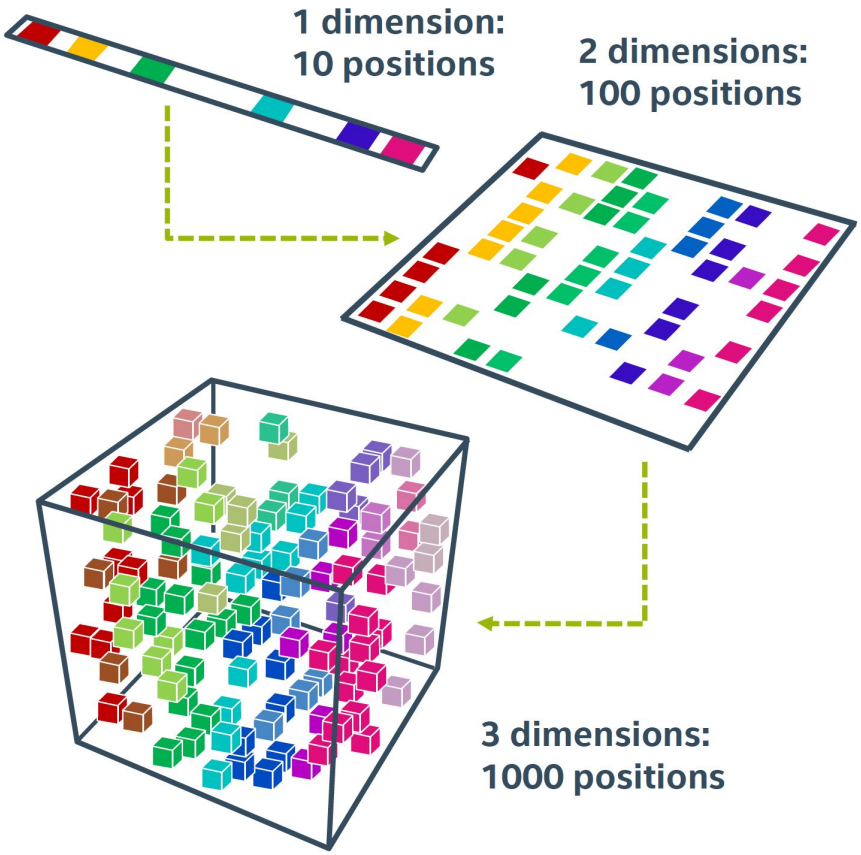
- Create dendrogram for iris dataset
- Apply Hierarchical Clustering to iris dataset



Topic 5
Dimension Reduction

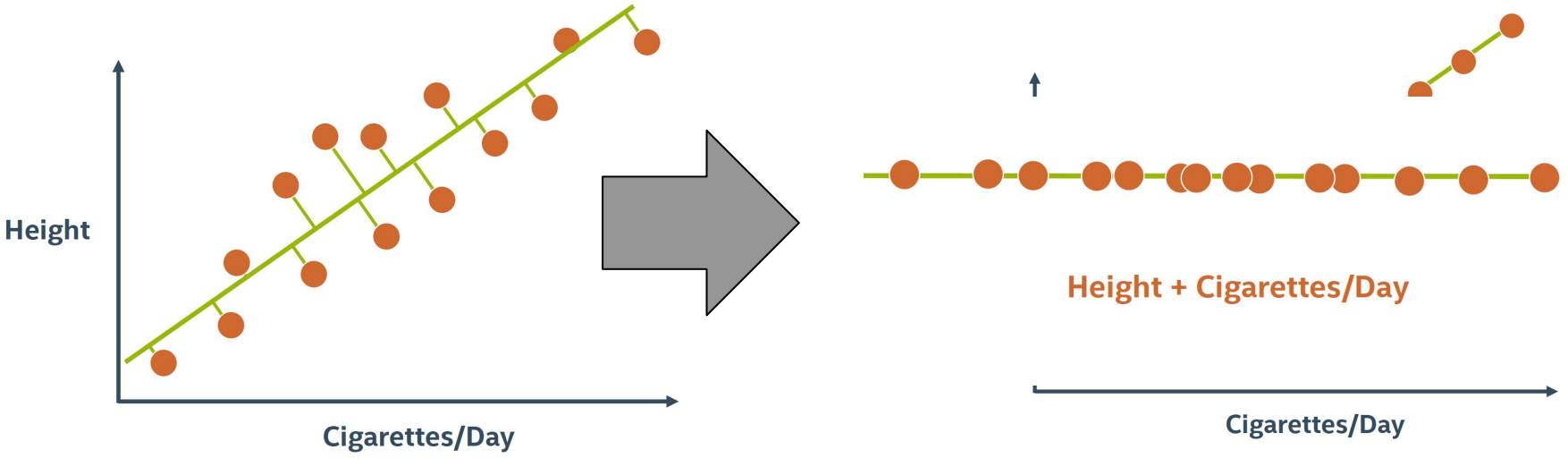
Curse of Dimensionality

- Theoretically, increasing features should improve performance
- In practice, more features leads to worse performance
- Number of training examples required increases exponentially with dimensionality



Dimension Reduction

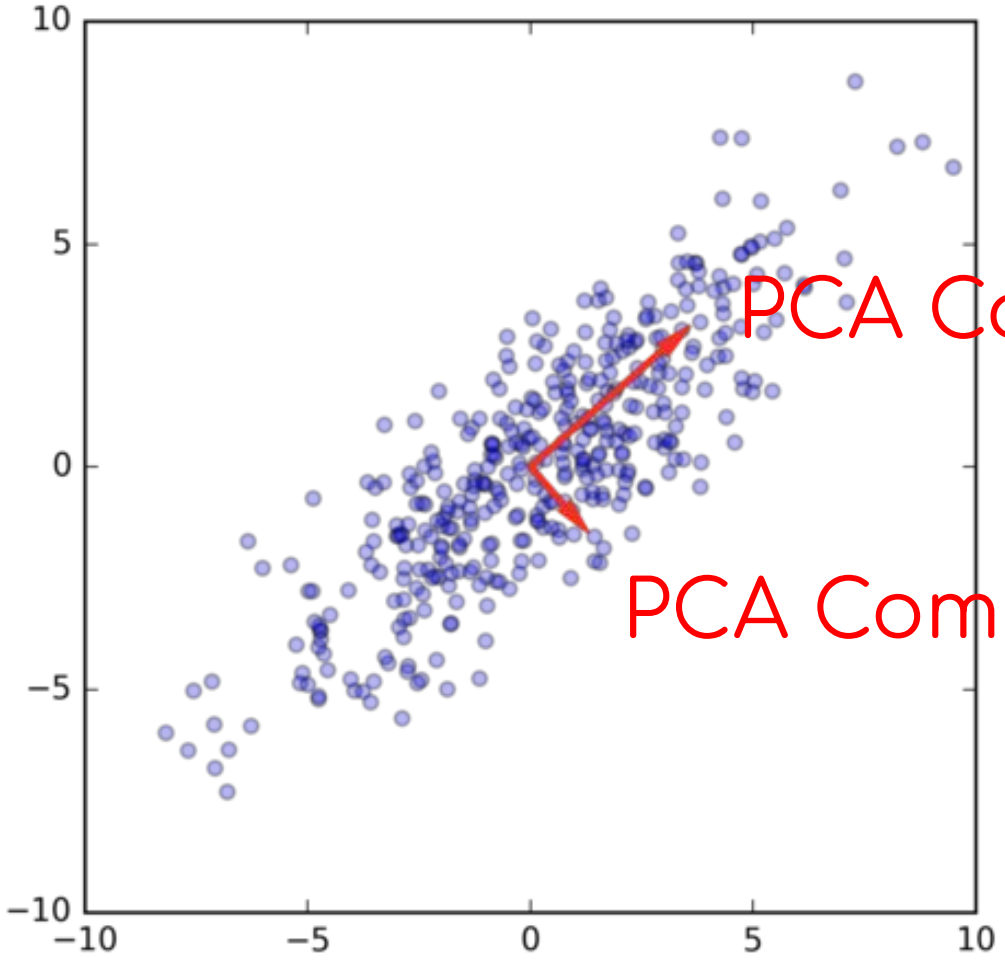
- Data can be represented by fewer dimensions (features)
- Reduce dimensionality by selecting subset (feature elimination)
- Combine with linear and non linear transformations
- Both features increase together. Can we reduce number of features to one?



Principal Component Analysis

- Create single feature that is combination of height and cigarettes. This is Principal Component Analysis (PCA)
- PCA is the most common dimensionality reduction technique.
- PCA was invented in 1901 by Karl Pearson

Principal Component Analysis



The principle components have the largest variations

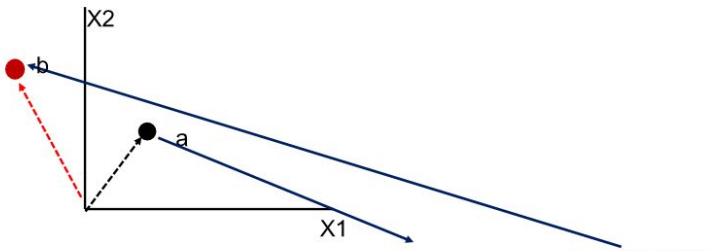
PCA Methodology

- Step 1: Standardize each column
  - If there are values of a different order of magnitude, then scale/standardize them. Convert the categorical variable into dummy numerical variables because PCA works only on numerical data.
- Step 2: Compute Covariance Matrix
  - Start with the analysis of the covariance matrix of the features.
  - Covariance measures how two variables are related to each other, that is, if two variables are moving in the same direction with respect to each other or not

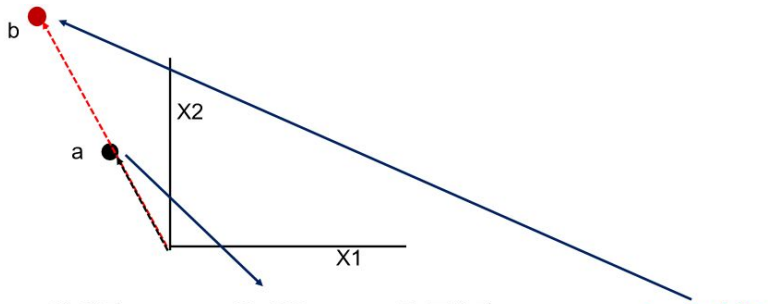
$$Cov(X, Y) = \frac{\Sigma(X_i - \bar{X})\,(Y_i - \bar{Y})}{n}$$

PCA Methodology

- Step 3: Compute Eigenvalues and Eigenvectors
  - The action of a matrix on a general vector can be thought of as a combination of stretch and rotation
  - For a given matrix there exists a special direction along which the effect is only stretch , these special directions are called eigenvector
  - The eigenvectors and eigenvalues of a matrix A are defined as $AX = \lambda X$ (A is just stretching)



$$\begin{bmatrix} 1.36 & -2.35 \\ -2.35 & 4.18 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -0.99 \\ 1.83 \end{bmatrix}$$



$$\begin{bmatrix} 1.36 & -2.35 \\ -2.35 & 4.18 \end{bmatrix} \begin{bmatrix} -0.49 \\ 0.87 \end{bmatrix} = \begin{bmatrix} -2.715 \\ 4.795 \end{bmatrix} = 5.51 \begin{bmatrix} -0.49 \\ 0.87 \end{bmatrix}$$

PCA Methodology

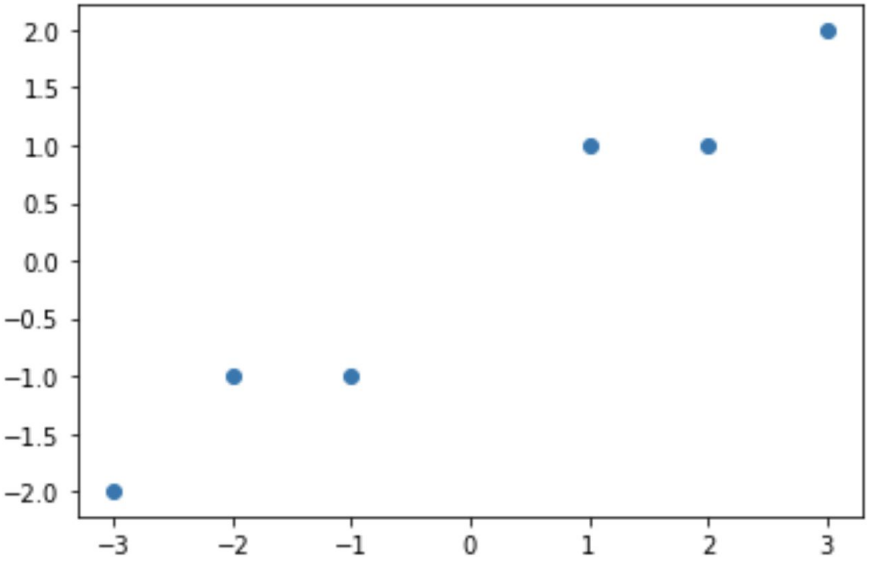
- Step 4: Derive principal component features
  - By taking the dot product of eigenvector and standardized columns, derive the principal component feature

PCA Attributes

- **explained_variance_:** The amount of variance explained by each of the selected components.
- **explained_variance_ratio_:** Percentage of variance explained by each of the selected components.
- **components_:** Principal axes in feature space, representing the directions of maximum variance in the data. The components are sorted by explained_variance_.
- **n_components_:** The estimated number of components

PCA Simple Demo

```
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])

from sklearn import decomposition
pca = decomposition.PCA(n_components=2)
pca.fit(X)
```

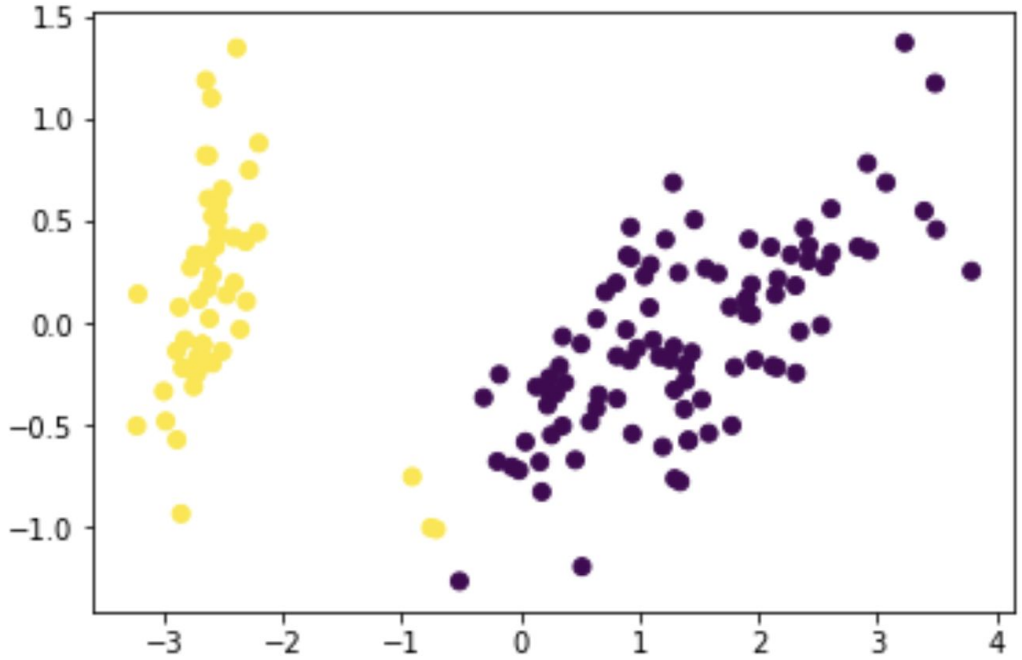


Principal Component Analysis

X_t= pca.fit_transform(X)
pca.explained_variance_
pca.explained_variance_ratio_

Ex: PCA on Iris Dataset

- Perform PCA on the iris dataset and analyze the variances of the components, and infer number of components needed to represent the iris dataset
- Compare K Means clustering to iris dataset before and after PCA



Summary
Q&A

Practice Sesion

- Open up the Practice Section from the Google Classroom
- It is a scenario based practice
- Code the program

Summary Q&A

Final Assessment

Written Assessment (Q&A)

This is the Written Assessment (Q&A)

Duration: 50 mins

1. The assessor will pass the questions in hardcopy to you. There are 10 questions. You need to answer all the questions.
2. This is an open book exam that must be completed individually.
3. You can use Python IDE or Google Colab to work out the answer

Written Assessment (Case Study)

This is a Written Assessment (Case Study)
Duration: 70 mins
1. The assessor will pass the case study in hardcopy to you.
2. You need to code a Python program in a new Google Colab project.
3. This is an open book exam that must be completed individually.

<u>Submission Procedure:</u>
1. At the Google Colab, Goto File -> Download as .ipynb
2. Rename your file as this format
CRS-Q-0038362-ICT_JohnTan_XXXX567A
3. Email the Jupyter file to the assessor

Feedback

https://goo.gl/R2eumq

TRAQOM Survey

- You will receive a TRAQOM link after the class.
- Please submit TRAQOM feedback and survey after the class.

Thank You!

Marcel

spectral.analytics
@protonmail.com