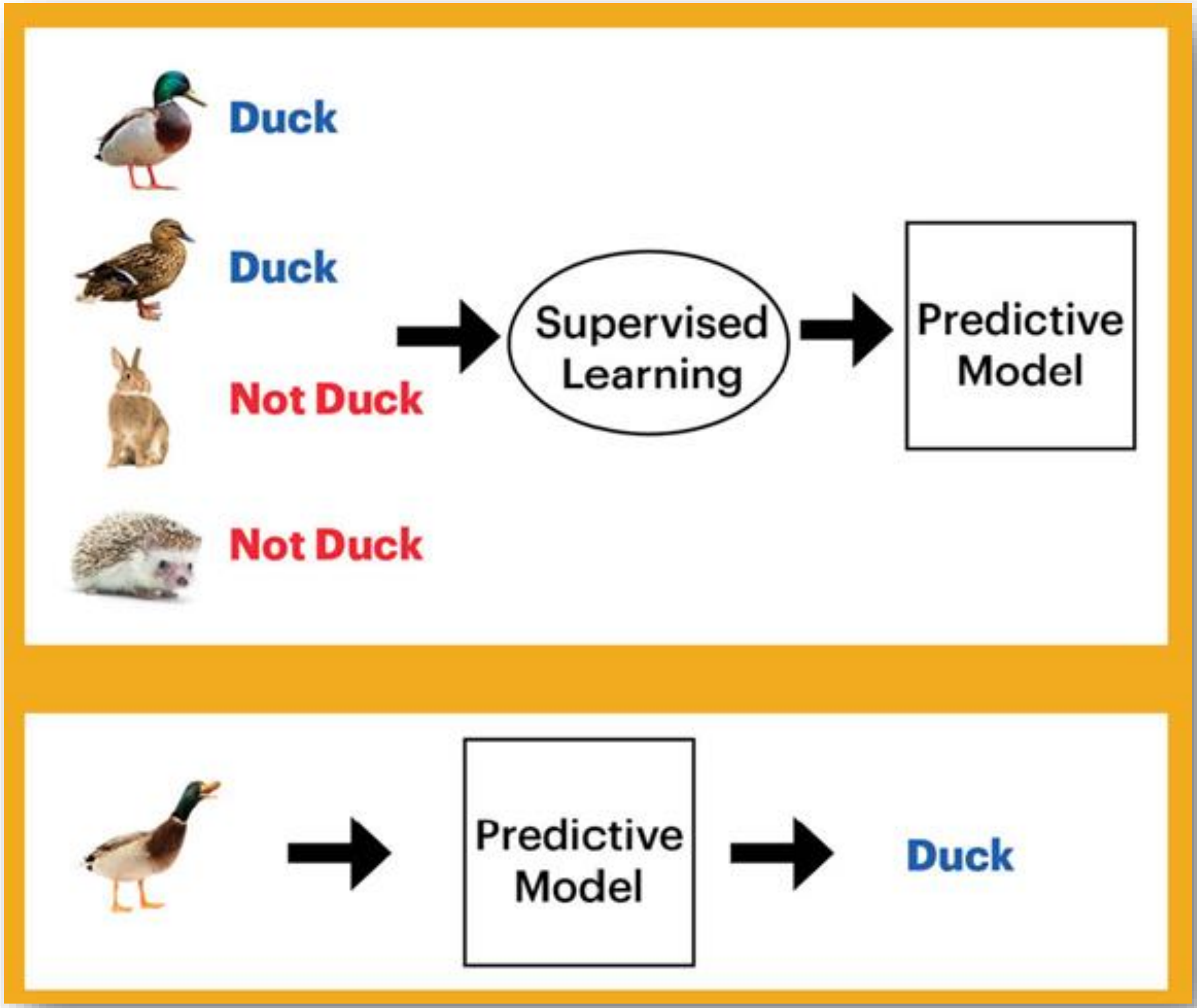
Supervised learning

Topic: introduction to classification & KNN Algorithm

By

Prof. Dr. Sourav Saha

Supervised learning



Supervised learning

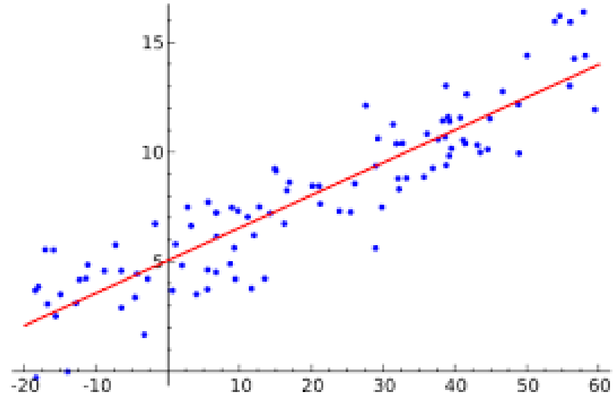
Example: House Prices

House Price in $1000s (y)	Square Feet (x)
245	1400
312	1600
279	1700
308	1875
199	1100
219	1550
405	2350
324	2450
319	1425
255	1700

Estimated Regression Equation:

$$\widehat{\text{house price}} = 98.25 + 0.1098\,(\text{sq.ft.})$$

Predict the price for a house with 2000 square feet



Predictive models (supervised)

- CLASSIFICATION

  Core: Predict the value of a category or class

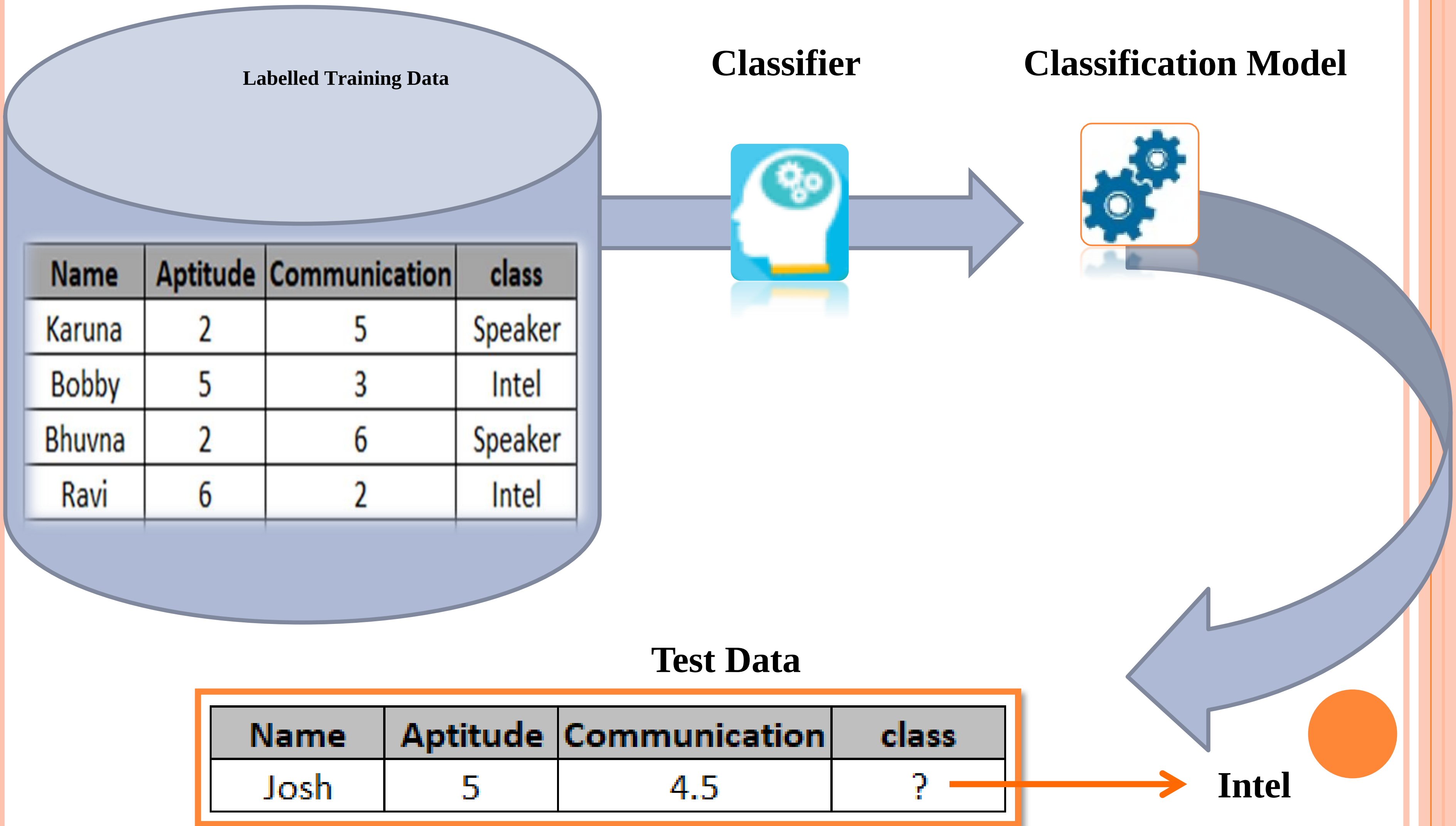
  - ✓ Problems that can be solved : Prediction of win/loss, fraudulent transactions, etc.
  - ✓ Examples : k-Nearest Neighbor (kNN), Decision Tree, Naïve Bayes, SVM, ANN etc.

- REGRESSION

  Core: Predict numerical values of the target

  - ✓ Problems that can be solved : Prediction of revenue growth, rainfall amount, etc.,
  - ✓ Examples: Simple Linear Regression, Multiple Regression, Logistic Regression etc.

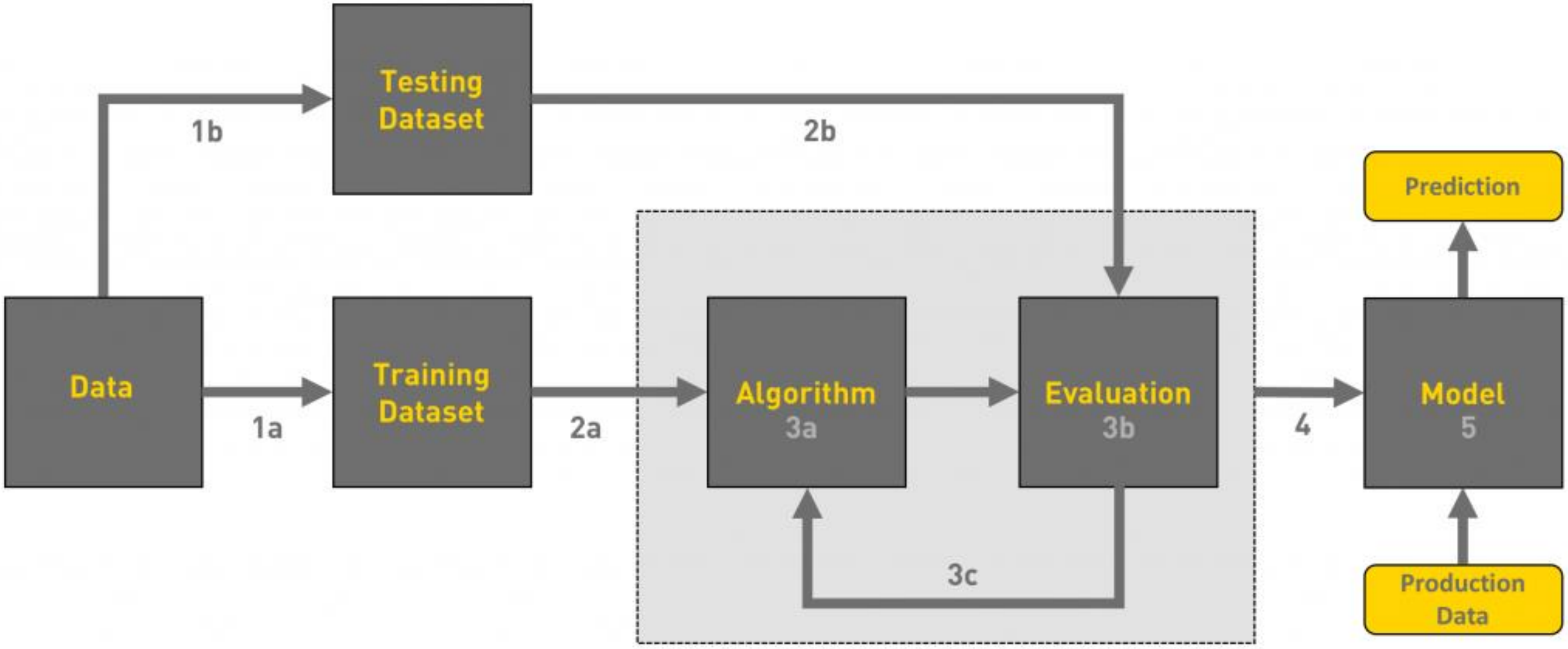
Supervised learning - Classification



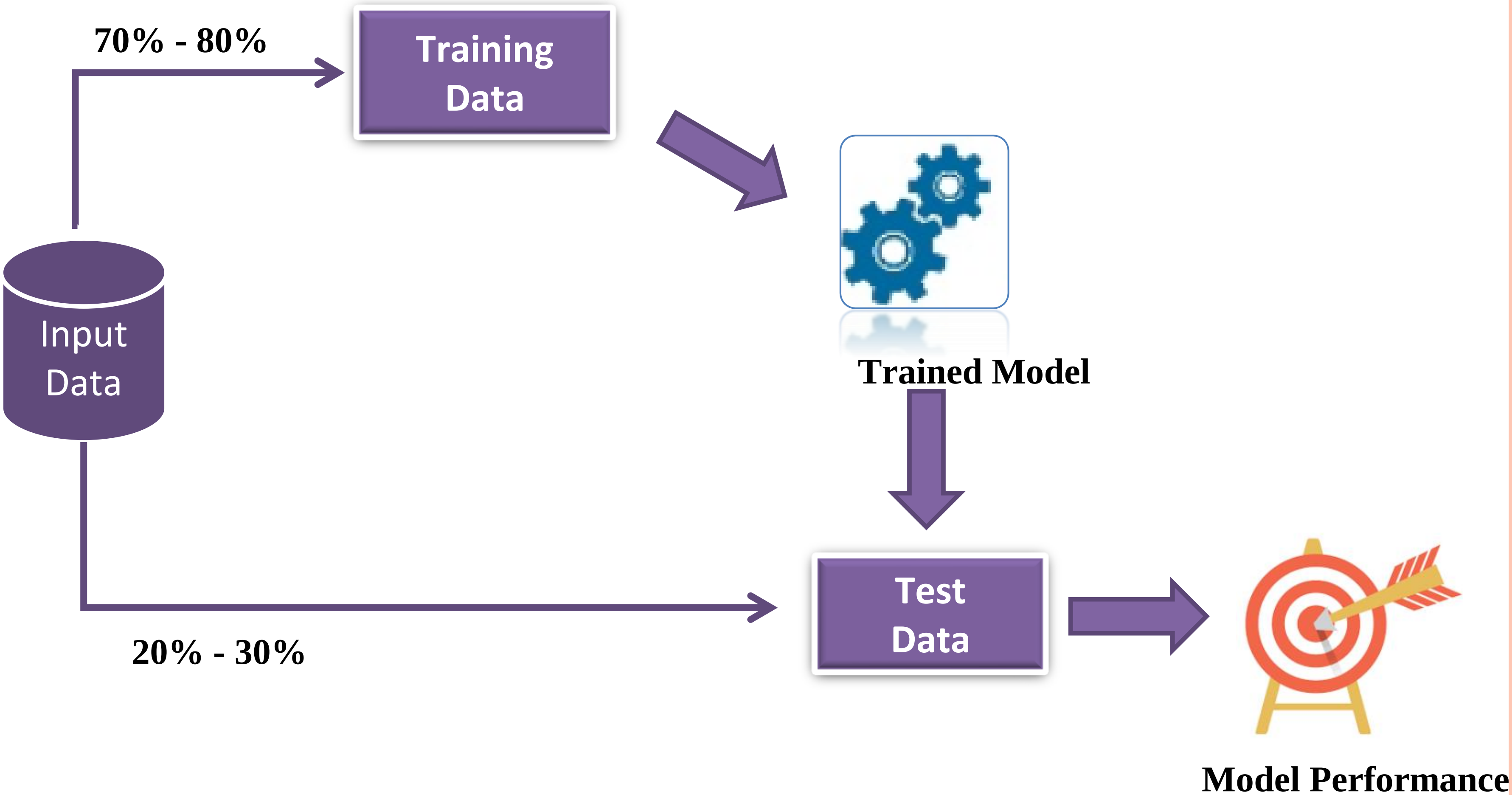
Name	Aptitude	Communication	class
Karuna	2	5	Speaker
Bobby	5	3	Intel
Bhuvna	2	6	Speaker
Ravi	6	2	Intel

Name	Aptitude	Communication	class
Josh	5	4.5	?

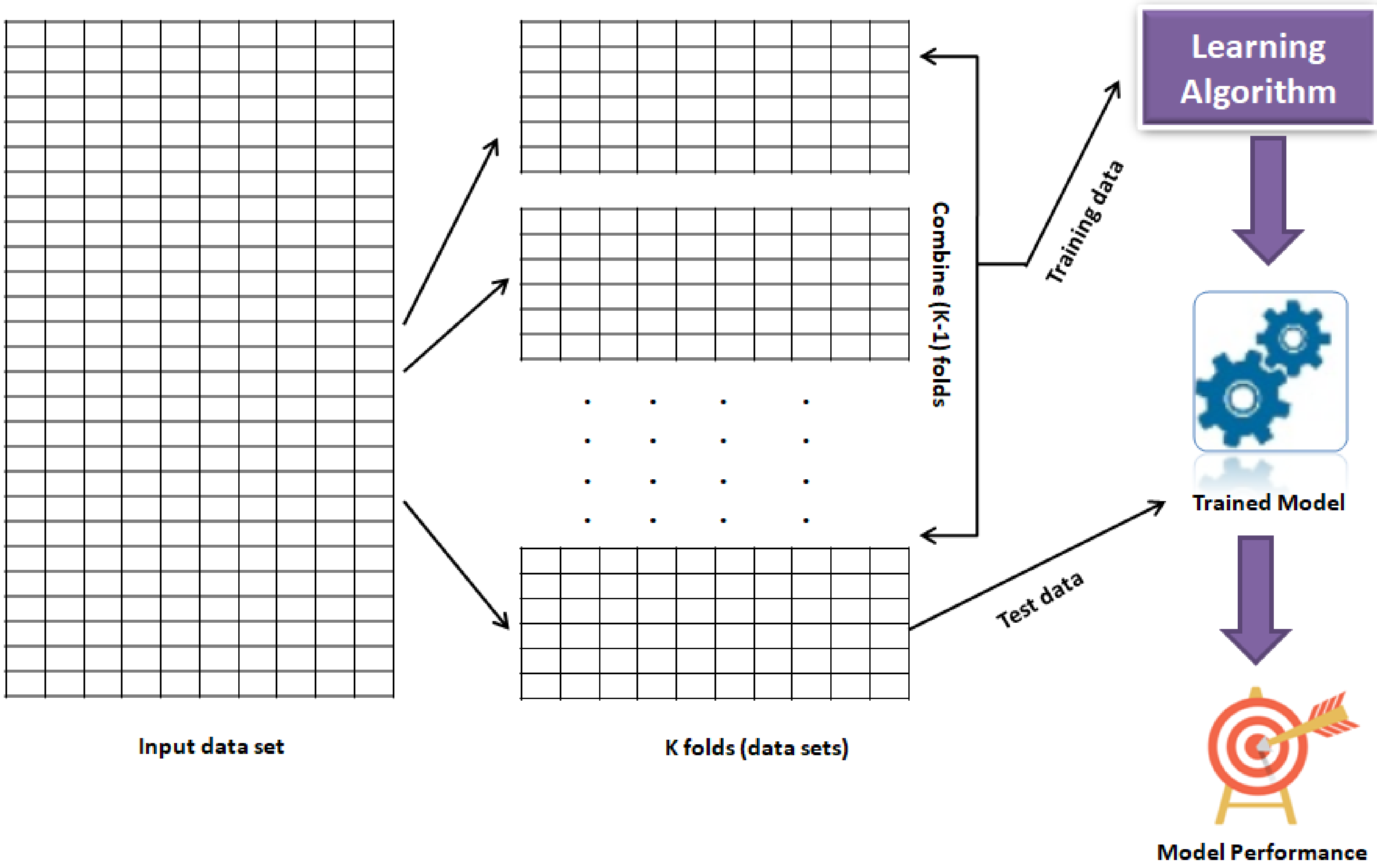
Overall process



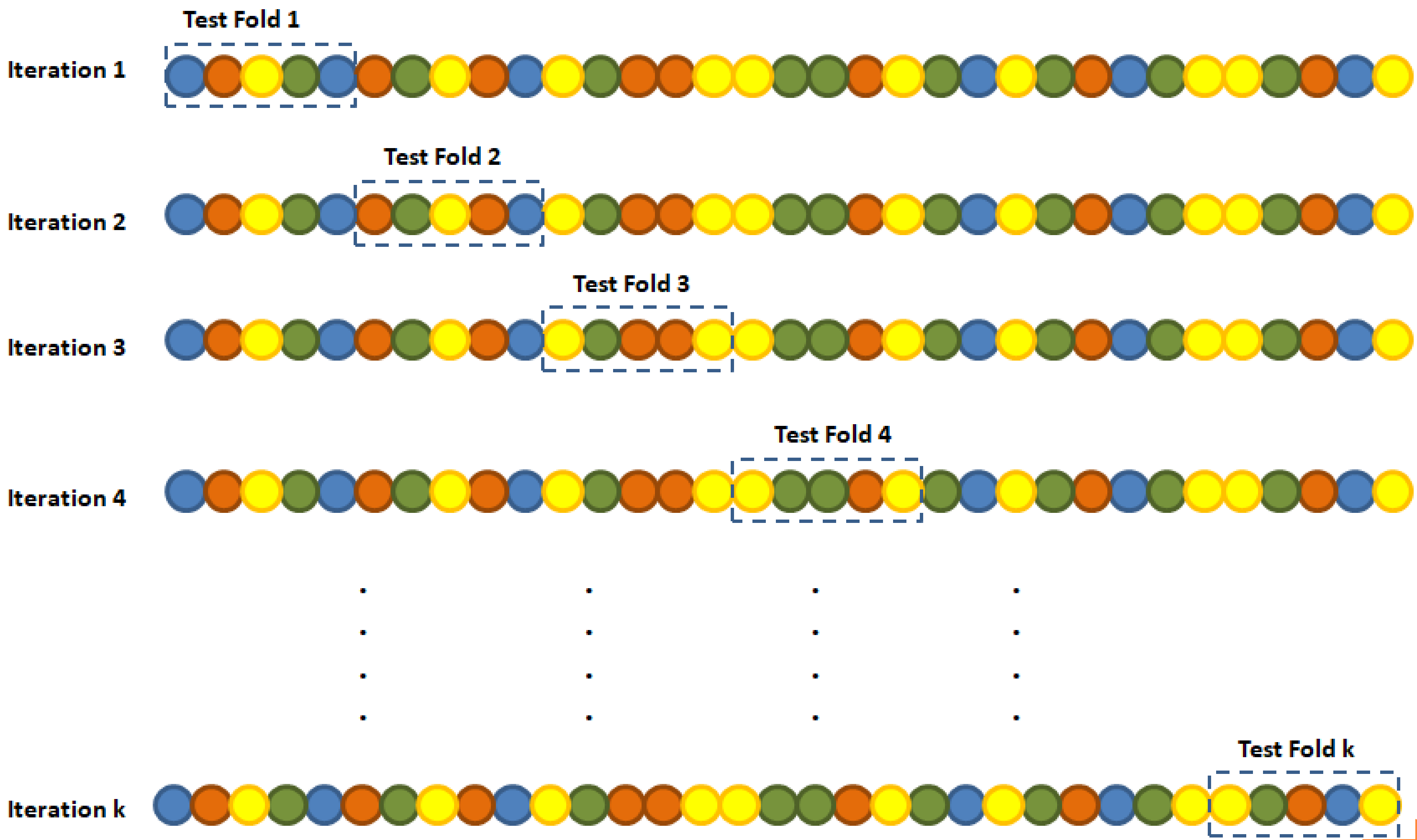
Data splitting – Simple Holdout Method



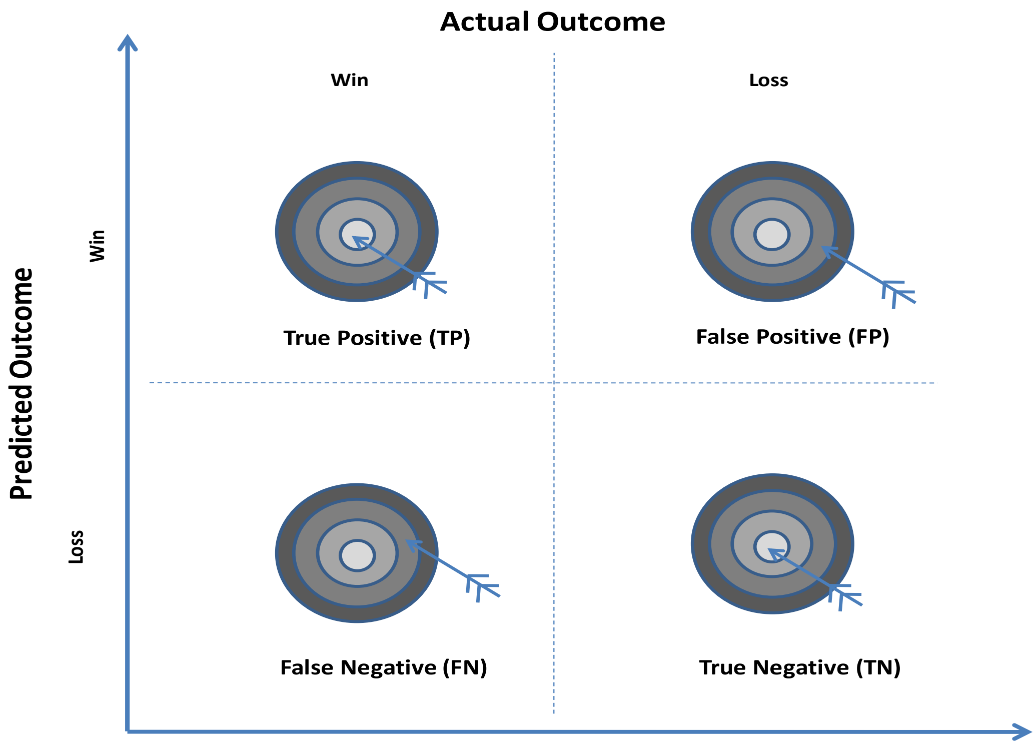
K-FOLD CROSS-VALIDATION– OVERALL APPROACH



K-FOLD CROSS-VALIDATION– DETAILED APPROACH



EVALUATING A MODEL (BINARY CLASSIFICATION)



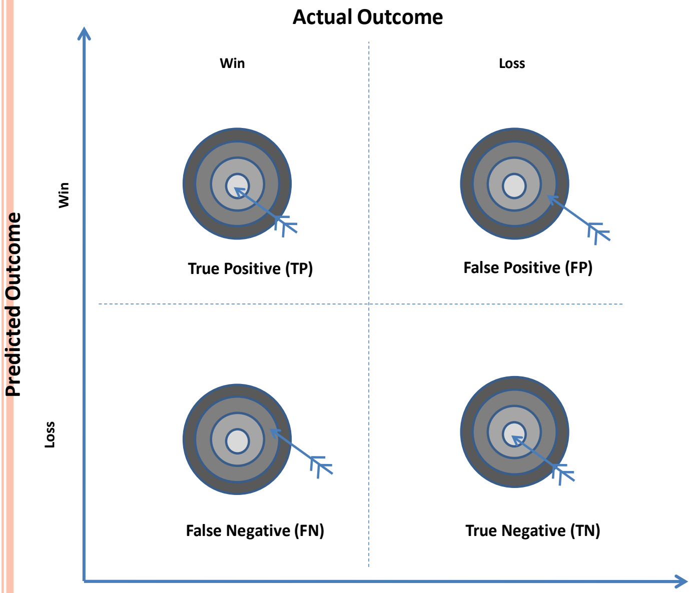
For both TP and TN, predicted outcome matches actual outcome. Hence, they are correct classifications.

Election Outcome Prediction Model:

Two Classes – win, loss

- True Positive (TP) – Predicted win, Actual win : : Truly Classified
- True Negative (TN) – Predicted loss, Actual loss : : Truly Classified
- False Positive (FP) – Predicted win, Actual loss : : Falsely Classified
- False Negative (FN) – Predicted loss, Actual win : : Falsely Classified

EVALUATING A MODEL (CLASSIFICATION)



	Actual Win	Actual Loss
Predicted Win	85	4
Predicted Loss	2	9

$$\text{Model accuracy} = \frac{TP+TN}{TP+FP+FN+TN} = \frac{85+9}{85+4+2+9} = \frac{94}{100} = 94\%$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{85}{85+4} = \frac{85}{89} = 95.5\%$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{85}{85+2} = \frac{85}{87} = 97.7\%$$

$$\text{F-measure} = \frac{2 \text{ X precision X recall}}{\text{precision} + \text{recall}} = \frac{2 \text{ X } 0.955 \text{ X } 0.977}{0.955 + 0.977} = \frac{1.866}{1.932} = 96.6\%$$

$$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{85}{85+2} = \frac{85}{87} = 97.7\%$$

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{9}{9+4} = \frac{9}{13} = 69.2\%$$

Evaluating a model (binary classification)

		Predicted Category	
		C_1 (+) Covid+	C_2 (−) Covid-
Actual Category	C_1 (+) Covid+	True Positive 85	False Negative 2
	C_2 (−) Covid-	False Positive 4	True Negative 9

$$\text{Model accuracy} = \frac{\text{TP+TN}}{\text{TP+FP+FN+TN}} = \frac{85+9}{85+4+2+9} = \frac{94}{100} = 94\%$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} = \frac{85}{85+4} = \frac{85}{89} = 95.5\%$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} = \frac{85}{85+2} = \frac{85}{87} = 97.7\%$$

$$\text{F-measure} = \frac{2 \text{ X precision X recall}}{\text{precision +recall}} = \frac{2 \text{ X } 0.955 \text{ X } 0.977}{0.955+ 0.977} = \frac{1.866}{1.932} = 96.6\%$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+FN}} = \frac{85}{85+2} = \frac{85}{87} = 97.7\%$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN+FP}} = \frac{9}{9+4} = \frac{9}{13} = 69.2\%$$

CONFUSION MATRIX – MULTI-CLASS CLASSIFICATION

Predicted ⇨ Actual ⇩	Classified Dog (38)	Classified Cat (51)	Classified Rabbit (44)
Actual Dog (42)	23 TP(Dog)	12 FN(Dog)	7 FN(Dog)
Actual Cat (53)	FP(Dog) 11 FN(Cat)	29 TP(Cat)	13 FN(Cat)
Actual Rabbit (38)	FP(Dog) 4 FN(Rabbit)	10 FN(Rabbit)	24 TP(Rabbit)

Precision	**Recall**	**F1-Score**
23/(23+11+4) = 23/38 = 0.60	23/(23+12+7) = 23/42 = 0.53	0.56
29/(12+29+10) = 29/51 = 0.56	29/(11+29+13) = 29/53 = 0.54	0.54
24/(7+13+24) = 24/44 = 0.54	24/(4+10+24) = 24/38 = 0.63	0.58
AVG = 0.56	AVG = 0.56	AVG = 0.56

PRECISION = TP/(TP + FP),
RECALL = TP/(TP+FN),
F1 = (2*PRECESION*RECALL)/(PRECESION + RECALL)
ACCURACY = TP/Total = (23+29+24)/(42+53+38) = 0.57 = 57%

Evaluating a model (classification)

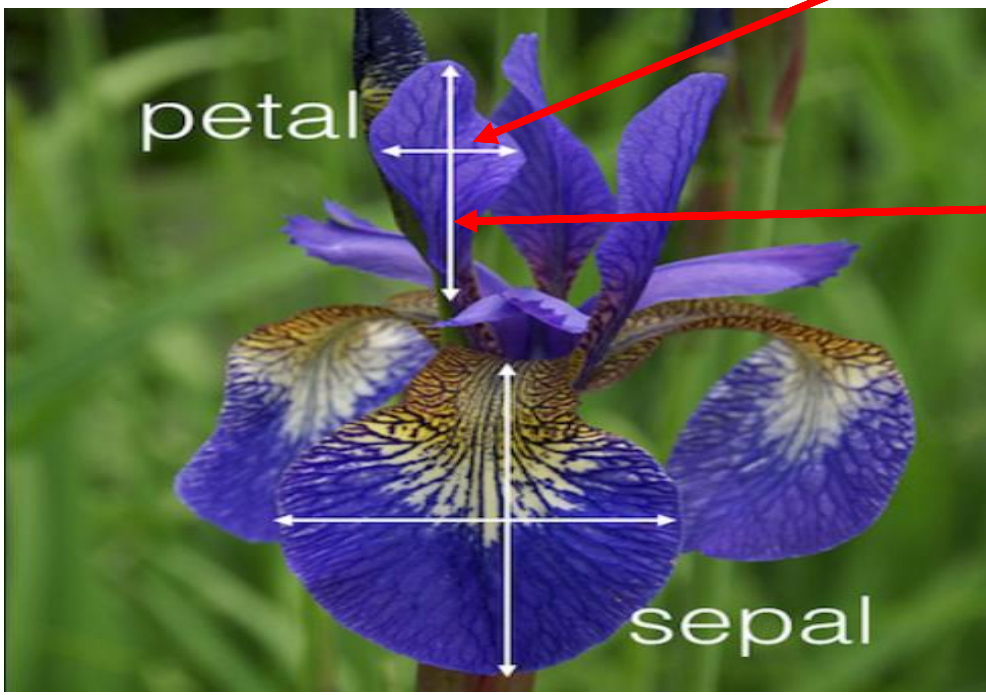
		predicted condition		
	total population	prediction positive	prediction negative	Sensitivity
true condition	condition positive	True Positive (TP)	False Negative (FN) (Type II error)	**Recall =** $\frac{\sum TP}{\sum \text{condition positive}}$
	condition negative	False Positive (FP) (Type I error)	True Negative (TN)	Specificity = $\sum TN / \sum \text{condition negative}$
	Accuracy = $\frac{\sum TP + \sum TN}{\sum \text{total population}}$	**Precision=** $\frac{\sum TP}{\sum \text{prediction positive}}$		**F1 Score =** $\frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$

Classification Example

Iris Versicolor **Iris Setosa** **Iris Virginica**

Classification Example

width



length

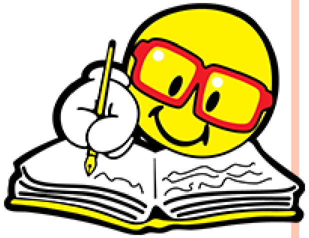
Classification Example

Features

	A	B	C	D	E	F
1	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	1	5.1	3.5	1.4	0.2	Iris-setosa
3	2	4.9	3	1.4	0.2	Iris-setosa
4	3	4.7	3.2	1.3	0.2	Iris-setosa
5	4	4.6	3.1	1.5	0.2	Iris-setosa
6	5	5	3.6	1.4	0.2	Iris-setosa

Class Labels

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor
56	5.7	2.8	4.5	1.3	Iris-versicolor



Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
146	6.7	3	5.2	2.3	Iris-virginica
147	6.3	2.5	5	1.9	Iris-virginica
148	6.5	3	5.2	2	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3	5.1	1.8	Iris-virginica

Classification ExampleSICS

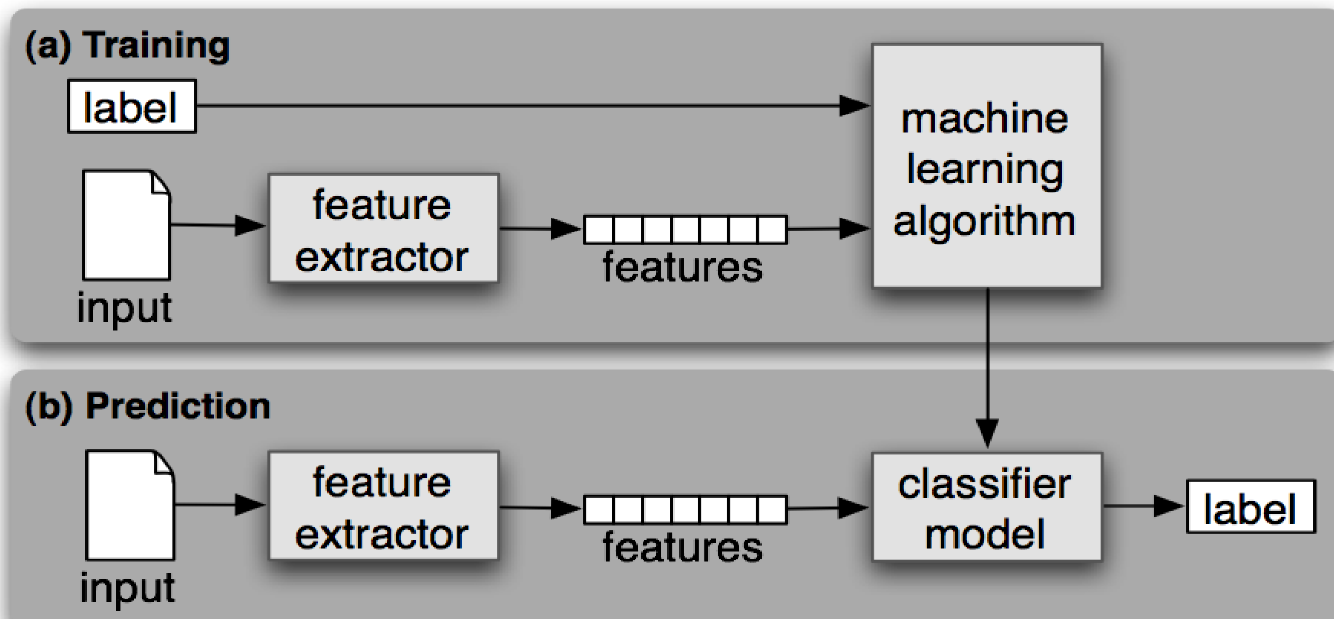
CLASSIFICATION CHALLENGE!!!

SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
6.3	2.8	5.1	1.5	???????????

PREDICTION OF SPECIES!!!

Action flow


Learn from experience
Learn from ~~experience~~ data
Follow instructions


(a) Training
label
input
feature extractor
features
machine learning algorithm
(b) Prediction
input
feature extractor
features
classifier model
label

OVERVIEW OF ACTION FLOW

- DATA PREPROCESSING: FEATURE EXTRACTION, DATA NORMALIZATION
- SPLITTING DATASET INTO TRAINING SET AND TESTING SET
- TRAINING THE MODEL: MACHINE LEARNING ALGORITHM
- TESTING THE MODEL: PERFOMANCE ASSESSMENT
- PREDICTING

Let's consider the input data …

Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	Intel

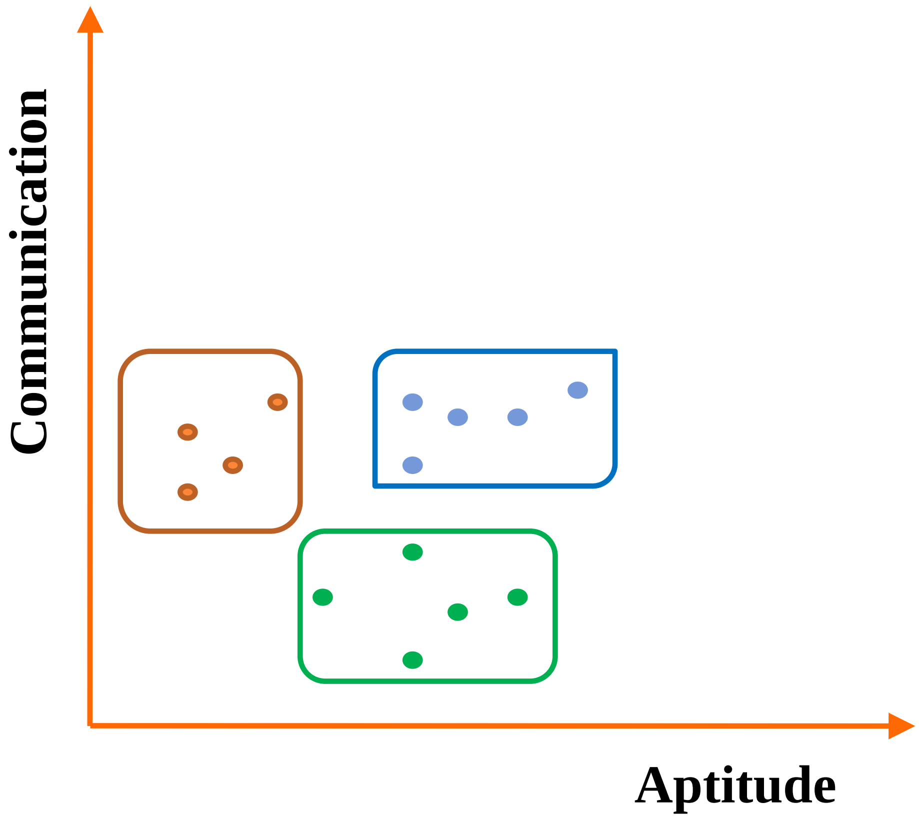
Data holdout

Training data

Test data

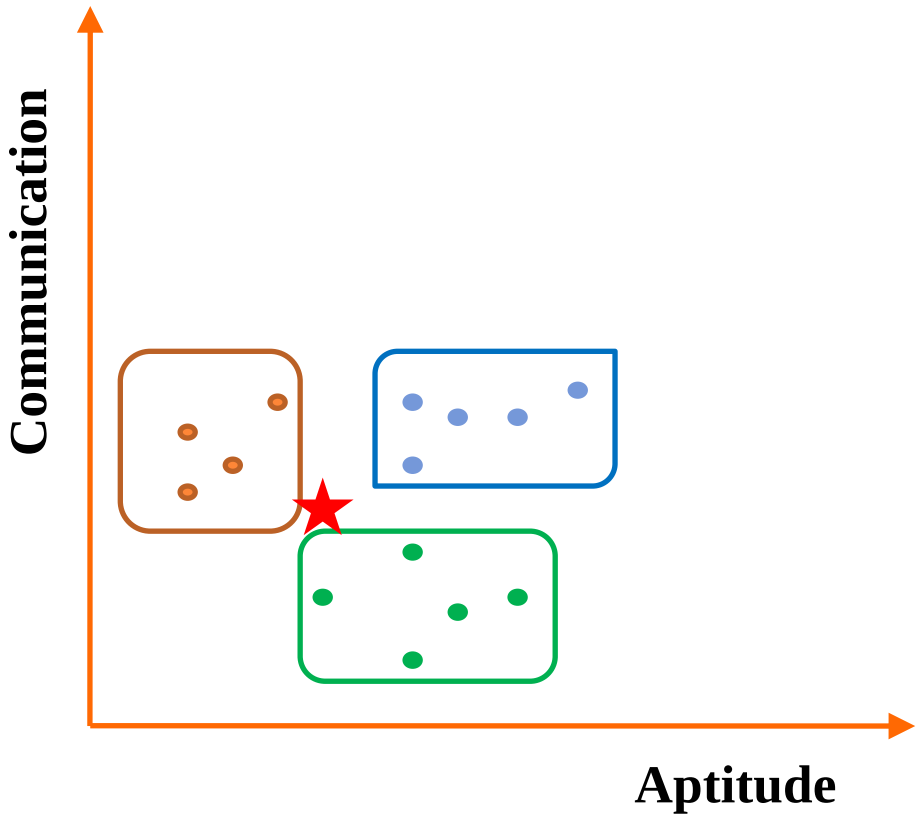
Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	Intel

LET'S SEE HOW THE TRAINING DATA IS GROUPED…



Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader

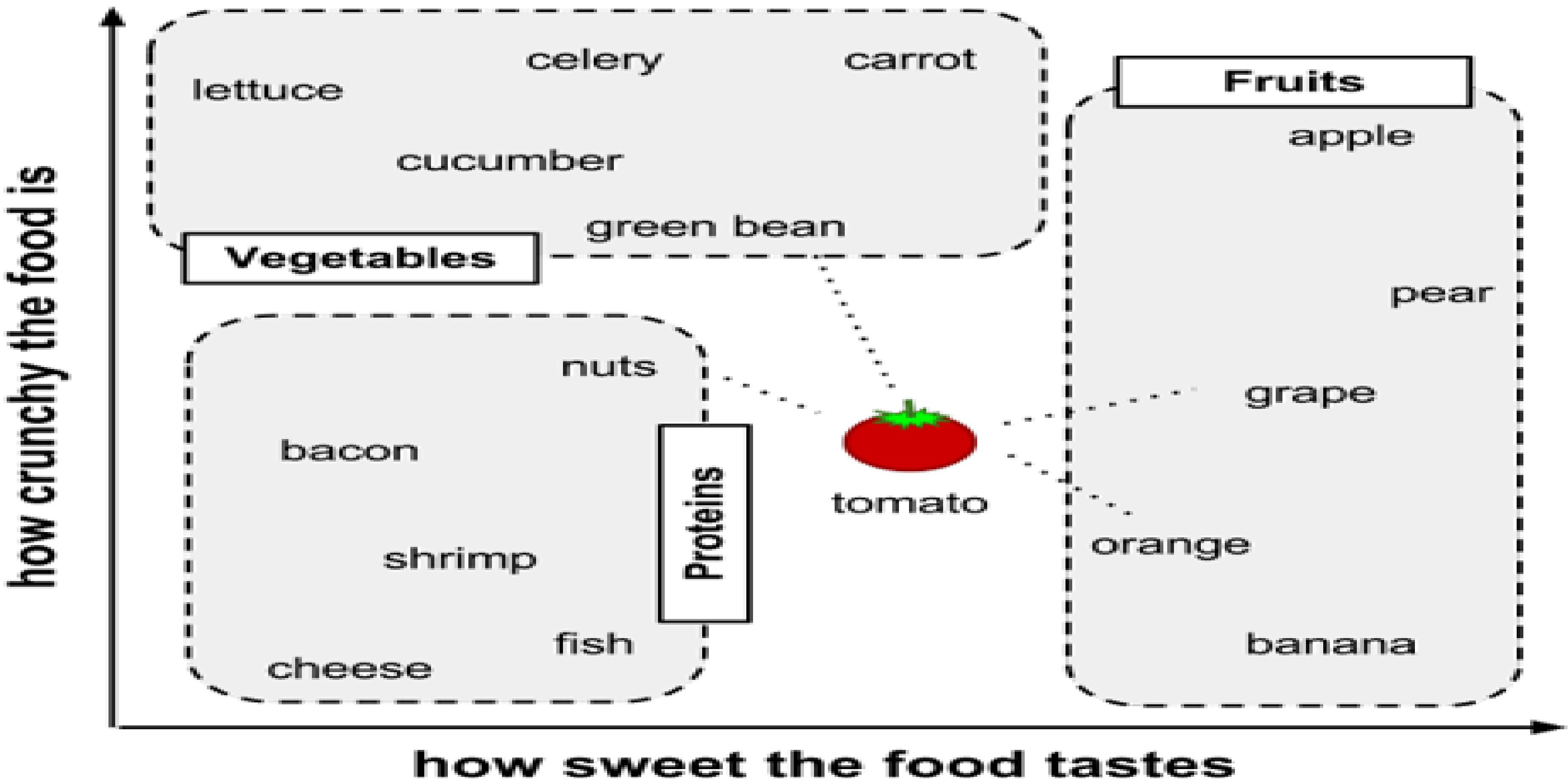
Say we don't know which class the test data belongs to …



Name	Aptitude	Communication	Class
Karuna	2	5	Speaker
Bhuvna	2	6	Speaker
Gaurav	7	6	Leader
Parul	7	2.5	Intel
Dinesh	8	6	Leader
Jani	4	7	Speaker
Bobby	5	3	Intel
Parimal	3	5.5	Speaker
Govind	8	3	Intel
Susant	6	5.5	Leader
Gouri	6	4	Intel
Bharat	6	7	Leader
Ravi	6	2	Intel
Pradeep	9	7	Leader
Josh	5	4.5	???

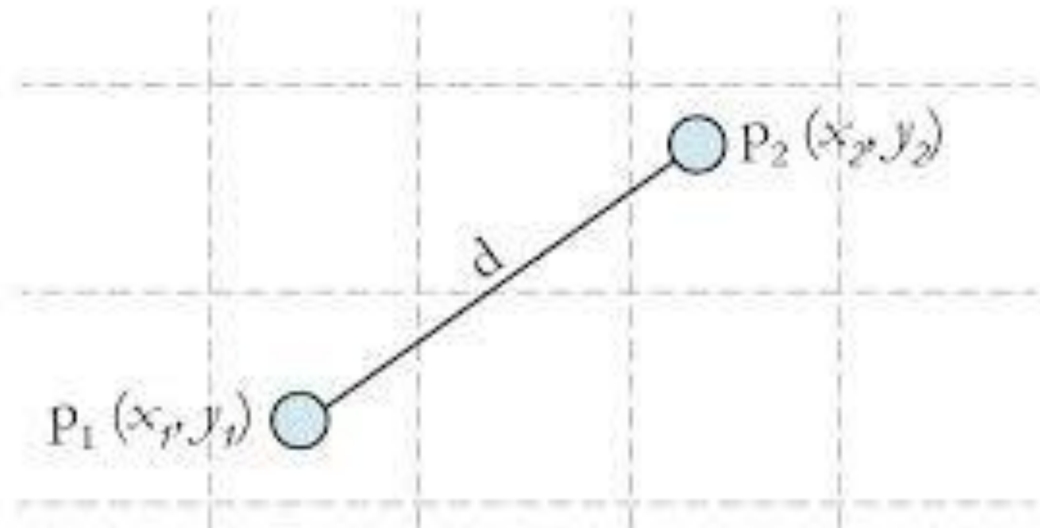
Nearest neighbour example

Item	sweetness	crunchy
grape	High (3)	Low (1)
nuts	Low (1)	High (3)
Green bean	**Medium (2)**	**Low (1)**
tomato	**Medium (2)**	**Low (1)**



Nearest neighbour example

Item	sweetness	crunchy
grape	High (3)	Low (1)
nuts	Low (1)	High (3)
Green bean	**Medium (2)**	**Low (1)**
tomato	**Medium (2)**	**Low (1)**



$$\text{Euclidean distance } (d) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

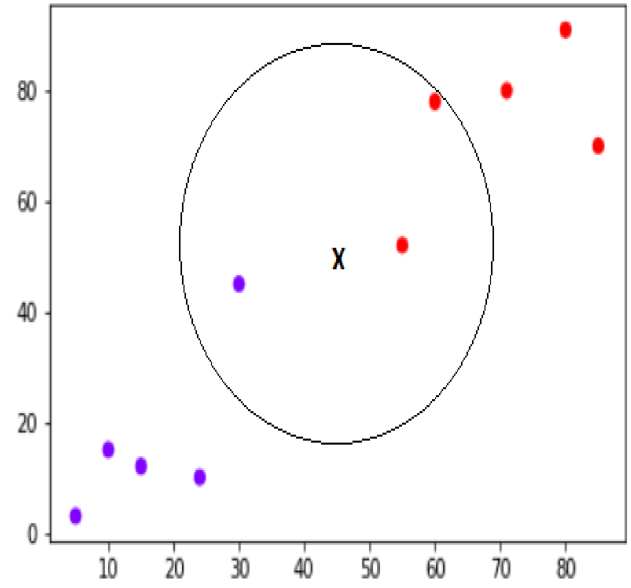
KNN Classifier Basics

Your task is to classify a new data point labelled with 'X' into "Blue" class or "Red" class. The coordinate values of the data point are x=45 and y=50.

The KNN algorithm starts by calculating the distance of point X from all the points.

It then finds the k=3 nearest points with least distance to point X.

The final step of the KNN algorithm is to assign X to the class to which majority of the three nearest points belong. From the figure, we can see that the two of the three nearest points belong to the class "Red" while one belongs to the class "Blue". Therefore X will be classified as "Red".



K NEAREST NEIGHBOR : STEP BY STEP

	Height (in cms)	Weight (in kgs)	T Shirt Size
2	158	58	M
3	158	59	M
4	158	63	M
5	160	59	M
6	160	60	M
7	163	60	M
8	163	61	M
9	160	64	L
10	163	64	L
11	165	61	L
12	165	62	L
13	165	65	L
14	168	62	L
15	168	63	L
16	168	66	L
17	170	63	L
18	170	64	L
19	170	68	L
20			
21	**161**	**61**	

Suppose we have height, weight and T-shirt size of some customers

we need to predict the T-shirt size of a new customer given only height and weight information we have.

Data including height, weight and T-shirt size information is shown here

K NEAREST NEIGHBOR : STEP BY STEP

Step 1 : Calculate Similarity based on distance function

Euclidean :

$$d(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

Manhattan / city - block :

$$d(x,y) = \sum_{i=1}^{m}|x_i - y_i|$$

The idea to use distance measure is to find the distance (similarity) between new sample and training cases and then find the k-closest customers to new customer in terms of height and weight.

New customer named 'Monica' has height 161cm and weight 61kg.

Euclidean distance between first data in the table and the given data for Monica is SQRT((161-158)^2+(61-58)^2)

Similarly, we need to calculate distance of all the training cases with new case and rank them in terms of distance from the new case.

K NEAREST NEIGHBOR : STEP BY STEP

Step 2 : Find K-Nearest Neighbors; Let K = 5

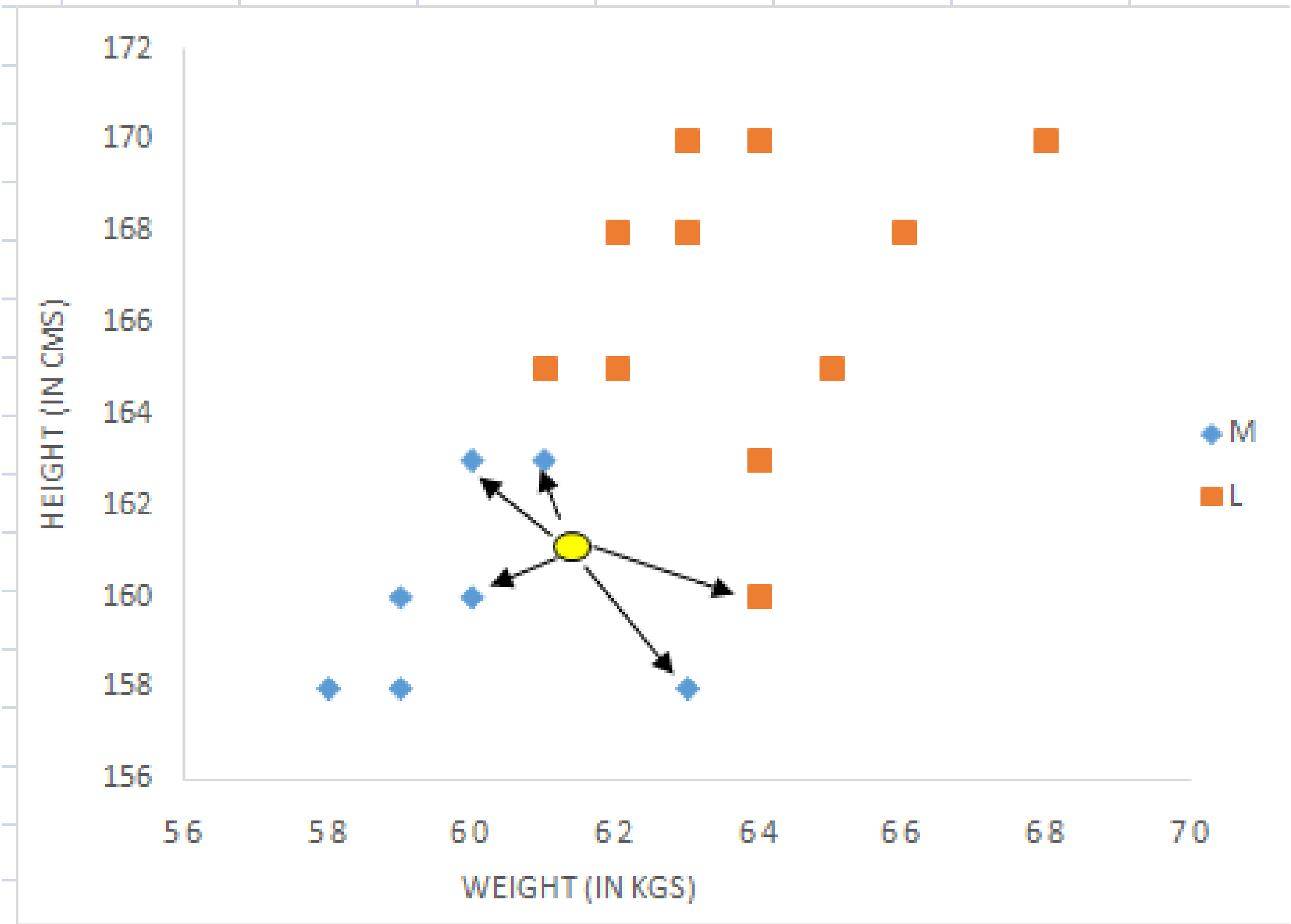
=SQRT((A21-A6)^2+(B21-B6)^2)

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	158	58	M	4.2	
3	158	59	M	3.6	
4	158	63	M	3.6	
5	160	59	**M**	2.2	**3**
6	160	60	**M**	1.4	**1**
7	163	60	**M**	2.2	**3**
8	163	61	**M**	2.0	**2**
9	160	64	**L**	3.2	**5**
10	163	64	L	3.6	
11	165	61	L	4.0	
12	165	62	L	4.1	
13	165	65	L	5.7	
14	168	62	L	7.1	
15	168	63	L	7.3	
16	168	66	L	8.6	
17	170	63	L	9.2	
18	170	64	L	9.5	
19	170	68	L	11.4	
20					
21	**161**	**61**			

The algorithm searches for the 5 customers closest to Monica, i.e. most similar to Monica in terms of attributes, and see what categories those 5 customers were in. If 4 of them had 'Medium T shirt sizes' and 1 had 'Large T shirt size' then your best guess for Monica is 'Medium T shirt. See the calculation shown in the snapshot

KNN-Graphical illustration

Medium T-shirt size' is shown in blue color and 'Large T-shirt size' is shown in orange color. New customer information is exhibited in yellow circle. Four blue highlighted data points and one orange highlighted data point are close to yellow circle. So the prediction for the new case is blue highlighted data point which is Medium T-shirt size.



KNN-Classification example

Name	Aptitude	Communication	Class	Distance	k=1	k=2	k=3
Karuna	2	5	Speaker	3.041			
Bhuvna	2	6	Speaker	3.354			
Parimal	3	5.5	Speaker	2.236			
Jani	4	7	Speaker	2.693			
Bobby	5	3	Intel	1.500			1.500
Ravi	6	2	Intel	2.693			
Gouri	6	4	Intel	1.118	1.118	1.118	1.118
Parul	7	2.5	Intel	2.828			
Govind	8	3	Intel	3.354			
Susant	6	5.5	Leader	1.414		1.414	1.414
Bharat	6	7	Leader	2.693			
Gaurav	7	6	Leader	2.500			
Dinesh	8	6	Leader	3.354			
Pradeep	9	7	Leader	4.717			
Josh	5	4.5	???				

Can KNN be used for regression?

Yes, K-nearest neighbor can be used for regression. In other words, K-nearest neighbor algorithm can be applied when dependent or target variable is continuous. In this case, the predicted value is the average of the values of its k nearest neighbors.

Name	Aptitude	Communication	Score	Distance	k=1	k=2	k=3
Karuna	2	5	11	3.041			
Bhuvna	2	6	13	3.354			
Parimal	3	5.5	16	2.236			
Jani	4	7	29	2.693			
Bobby	5	3	14	1.500			1.500
Ravi	6	2	11	2.693			
Gouri	6	4	23	1.118	1.118	1.118	1.118
Parul	7	2.5	13	2.828			
Govind	8	3	23	3.354			
Susant	6	5.5	29	1.414		1.414	1.414
Bharat	6	7	43	2.693			
Gaurav	7	6	41	2.500			
Dinesh	8	6	47	3.354			
Pradeep	9	7	62	4.717			
Josh	5	4.5	???		23	26	22

KNN-Algorithm

Step 1. Load the data
Step 2. Initialize K to your chosen number of neighbors
Step 3. For each example in the data

Step 3.1 Calculate the distance between the query example and the current example from the data.

Step 3.2 Add the distance and the index of the example to an ordered collection

Step 4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
Step 5. Pick the first K entries from the sorted collection
Step 6. Get the labels of the selected K entries
Step 7. If regression, return the mean of the K labels
Step 8. If classification, return the mode of the K labels

STANDARDIZATION

When independent variables in training data are measured in different units, it is important to standardize variables before calculating distance. For example, if one variable is based on height in cms, and the other is based on weight in kgs then height will influence more on the distance calculation. In order to make them comparable we need to standardize

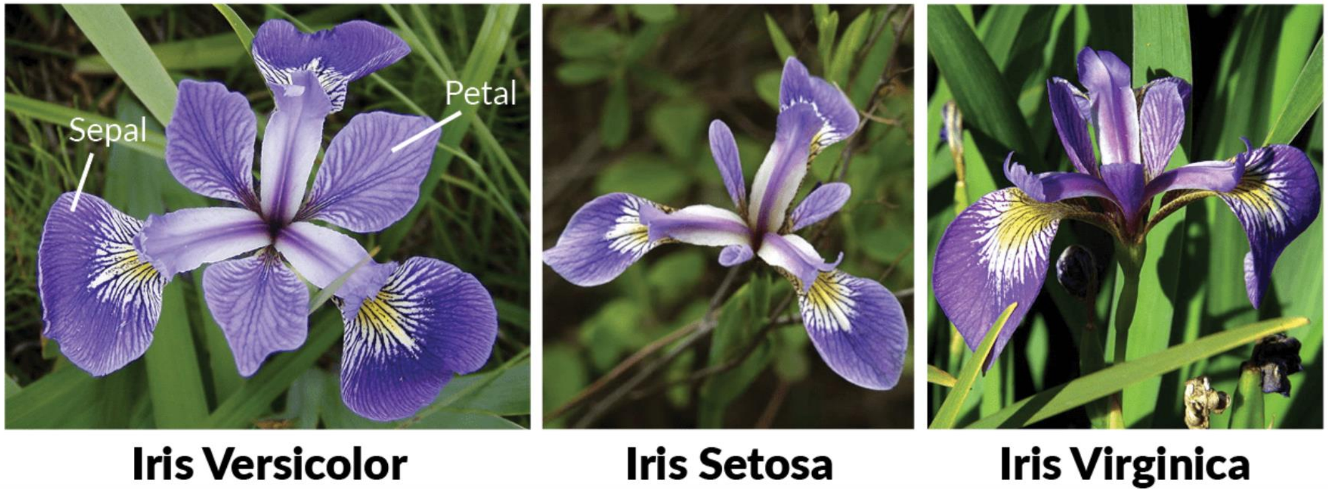
$$Xs = \frac{X - mean}{s.d.}$$

$$Xs = \frac{X - mean}{max - min}$$

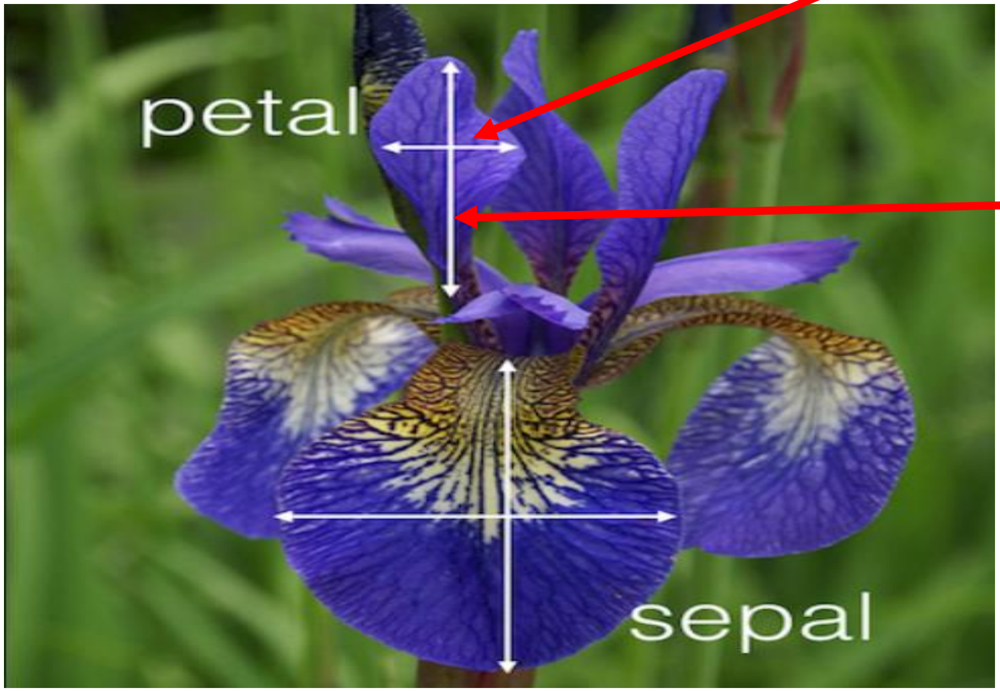
$$Xs = \frac{X - min}{max - min}$$

	A	B	C	D	E
1	Height (in cms)	Weight (in kgs)	T Shirt Size	Distance	
2	-1.39	-1.64	M	1.3	
3	-1.39	-1.27	M	1.0	
4	-1.39	0.25	M	1.0	
5	-0.92	-1.27	**M**	0.8	**4**
6	-0.92	-0.89	**M**	0.4	**1**
7	-0.23	-0.89	**M**	0.6	**3**
8	-0.23	-0.51	**M**	0.5	**2**
9	-0.92	0.63	L	1.2	
10	-0.23	0.63	L	1.2	
11	0.23	-0.51	**L**	0.9	**5**
12	0.23	-0.13	L	1.0	
13	0.23	1.01	L	1.8	
14	0.92	-0.13	L	1.7	
15	0.92	0.25	L	1.8	
16	0.92	1.39	L	2.5	
17	1.39	0.25	L	2.2	
18	1.39	0.63	L	2.4	
19	1.39	2.15	L	3.4	
20					
21	**-0.7**	**-0.5**			

Classification Example: revisited



Classification Example: revisited



Classification Example: revisited

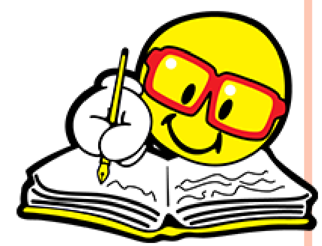
Features

Class Labels

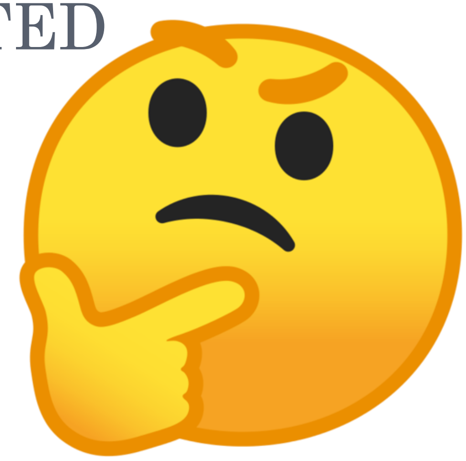
	A	B	C	D	E	F
1	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
2	1	5.1	3.5	1.4	0.2	Iris-setosa
3	2	4.9	3	1.4	0.2	Iris-setosa
4	3	4.7	3.2	1.3	0.2	Iris-setosa
5	4	4.6	3.1	1.5	0.2	Iris-setosa
6	5	5	3.6	1.4	0.2	Iris-setosa

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
52	6.4	3.2	4.5	1.5	Iris-versicolor
53	6.9	3.1	4.9	1.5	Iris-versicolor
54	5.5	2.3	4	1.3	Iris-versicolor
55	6.5	2.8	4.6	1.5	Iris-versicolor
56	5.7	2.8	4.5	1.3	Iris-versicolor

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
146	6.7	3	5.2	2.3	Iris-virginica
147	6.3	2.5	5	1.9	Iris-virginica
148	6.5	3	5.2	2	Iris-virginica
149	6.2	3.4	5.4	2.3	Iris-virginica
150	5.9	3	5.1	1.8	Iris-virginica



Classification Example: revisited



CLASSIFICATION CHALLENGE!!!

SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
6.3	2.8	5.1	1.5	???????????

PREDICTION OF SPECIES!!!

DATA SPLITTING INTO TRAINING & TESTING SET: HOLDOUT METHOD

```python
# Importing Libraries
import pandas as pd

# Importing the Dataset
dataset = pd.read_csv("./data/iris.csv")
print(dataset.head())

# Preprocessing
X = dataset.iloc[:, 1:5]
y = dataset.iloc[:, 5]

# Train Test Split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)
```

Preprocessing: Standardization implementation

```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

Implementation: MODEL TRAINING

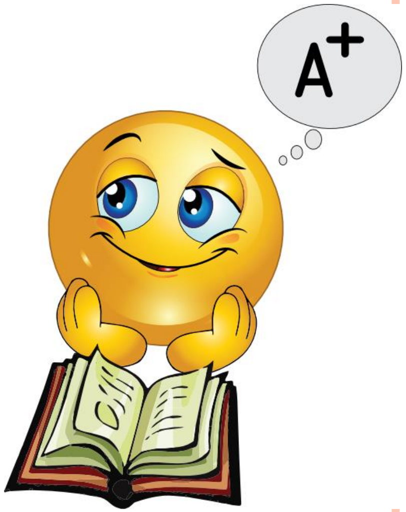
```
# Training
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)
```

PERFORMANCE ASSESSMENT

```
# Predictions
y_pred = classifier.predict(X_test)

# Evaluating the Algorithm
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

CLASSIFIER'S PERFORMANCE ASSESSMENT



	Iris-setosa	Iris-versicolor	Iris-virginica
Iris-setosa	[[9	0	0]
Iris-versicolor	[0	5	2]
Iris-virginica	[0	1	13]]

Confusion Matrix

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	9
Iris-versicolor	0.83	0.71	0.77	7
Iris-virginica	0.87	0.93	0.90	14
avg / total	0.90	0.90	0.90	30

PRECISION = TP/(TP + FP),
RECALL = TP/(TP+FN),
F1 = (2*PRECESION*RECALL)/(PRECESION+RECALL)

SAVING MODEL OFFLINE

```python
import pickle
# Model persistence
output_model_file = 'Knnmodel.pkl'

# Save the model
with open(output_model_file, 'wb') as f:
    pickle.dump(classifier, f)
```

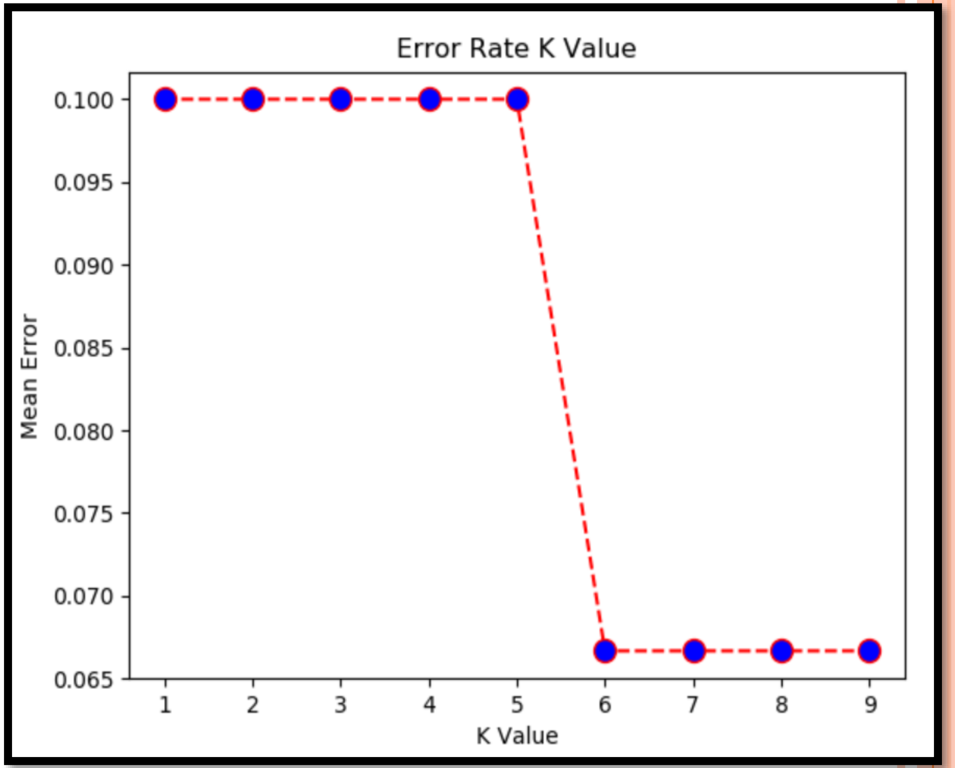
VALUE OF K???

```python
import numpy as np

# Comparing Error Rate with the K Value
error = []

# Calculating error for K values between 1 and 40
for i in range(1, 10):
    knn = KNeighborsClassifier(n_neighbors = i)
    knn.fit(X_train, y_train)
    pred_i = knn.predict(X_test)
#     print(pred_i != y_test)
#     print(np.mean(pred_i != y_test))
    error.append(np.mean(pred_i != y_test))

import matplotlib.pyplot as plt
plt.plot(range(1, 10), error,
         color='red', linestyle='dashed',
         marker='o', markerfacecolor='blue',
         markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
plt.show()
```



Low k-value is sensitive to outliers and a higher K-value is more resilient to outliers as it considers more voters to decide prediction.

CLASSIFICATION PREDICTION

```python
import pickle
# Model persistence
output_model_file = 'Knnmodel.pkl'

# Load the model
with open(output_model_file, 'rb') as f:
    knn = pickle.load(f)

# New Feature Set
import numpy as np
X = np.array([[ 4.8, 3.0, 1.4, 0.3], [7,    3.2,    4.7,    1.4]])
print(X)

# Feature pre-processing by standardization
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X = scaler.transform(X)

# Making Predictions
predictions = knn.predict(X)
print(predictions)
```

OUTPUT

```
[[ 4.8  3.   1.4  0.3]
 [ 7.   3.2  4.7  1.4]]
['Iris-setosa' 'Iris-virginica']
```

Pros and Cons of KNN

Lazy learning algorithm − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.

Non-parametric learning algorithm − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Pros

- **It is very simple algorithm to understand and interpret.**
- **It is very useful for nonlinear data because there is no assumption about data in this algorithm.**
- **It is a versatile algorithm as we can use it for classification as well as regression.**
- **It has relatively high accuracy but there are much better supervised learning models than KNN. It is fairly easy to add new data to algorithm.**

Cons

- **It is computationally a bit expensive algorithm because it compares with all the training data.**
- **High memory storage required as compared to other supervised learning algorithms.**
- **Hard to work with categorical features.**
- **It is very sensitive to the scale of data as well as irrelevant features.**

Thank You