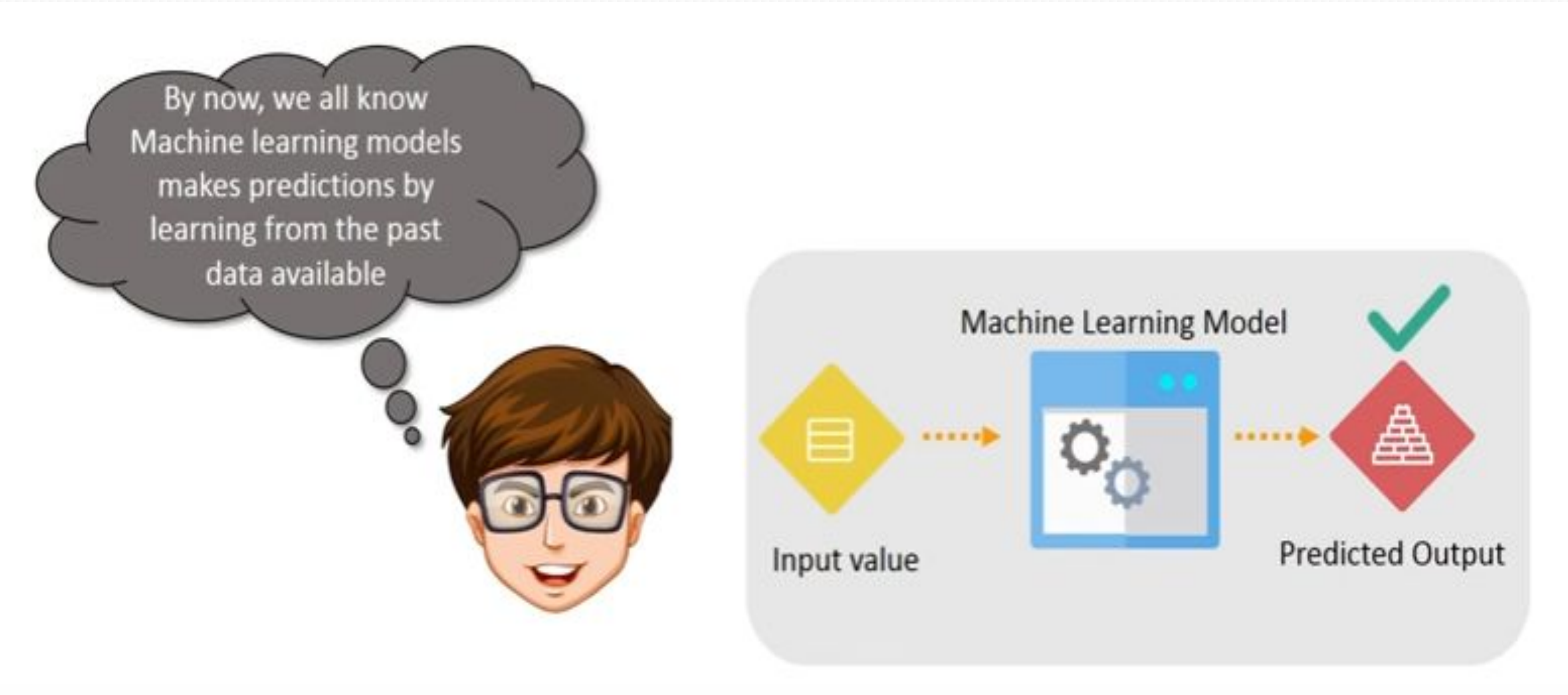
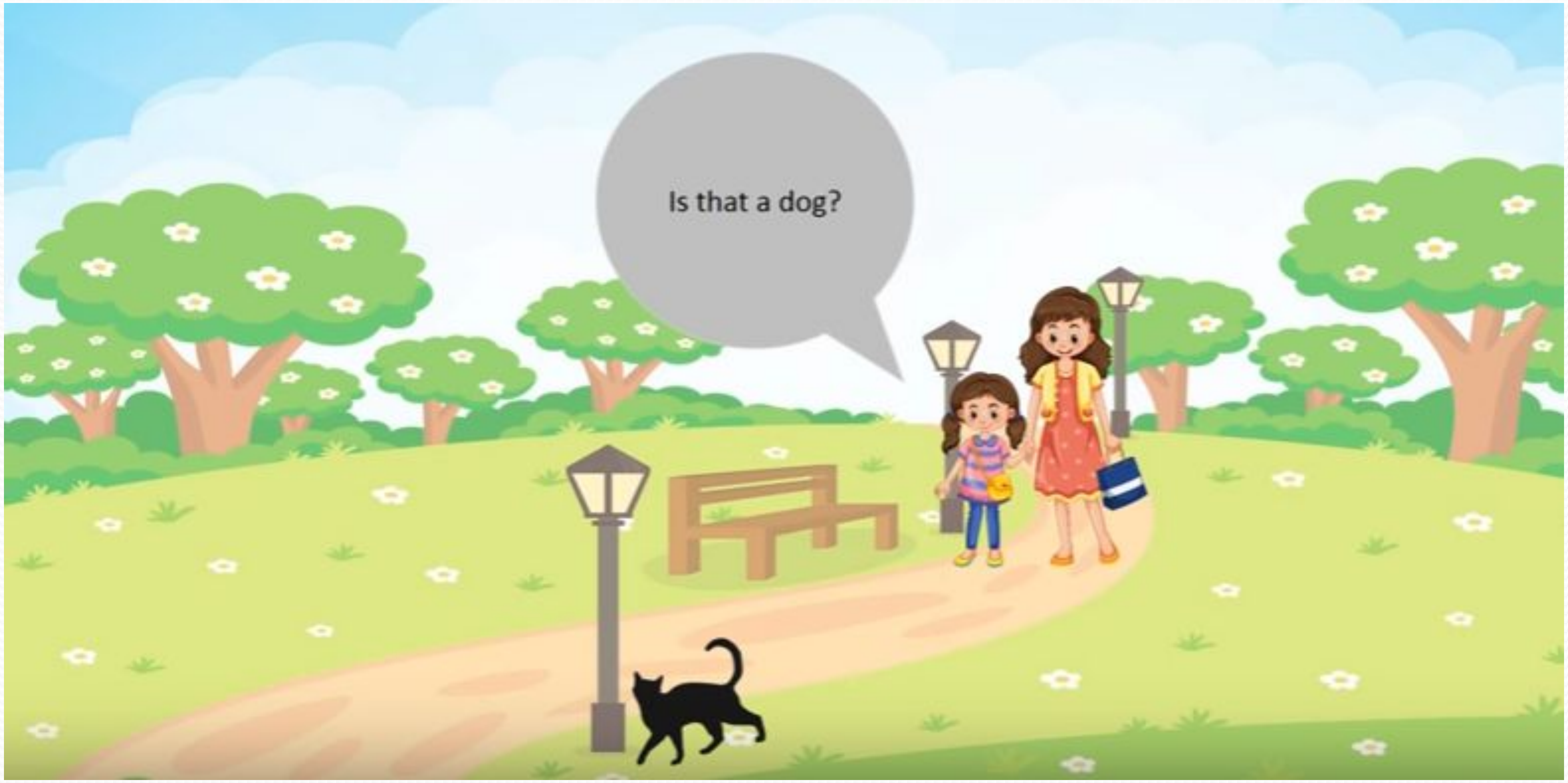
KNN Algorithm

Contents

- Why do we need KNN?
- What is KNN?
- How do we use KNN?
- How does KNN Algorithm work?
- Use Case: Predict whether a person will have diabetes or not

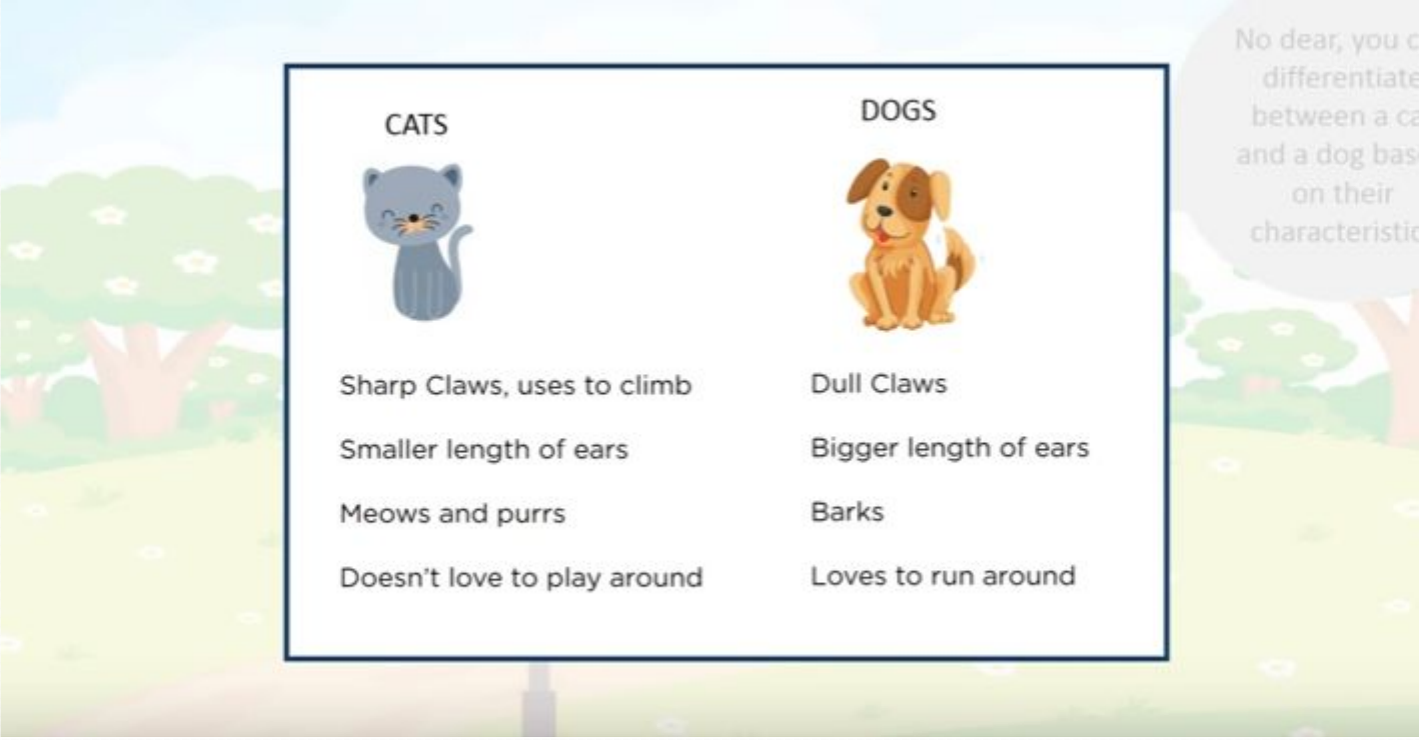
Why KNN?


By now, we all know Machine learning models makes predictions by learning from the past data available
Machine Learning Model
Input value
Predicted Output



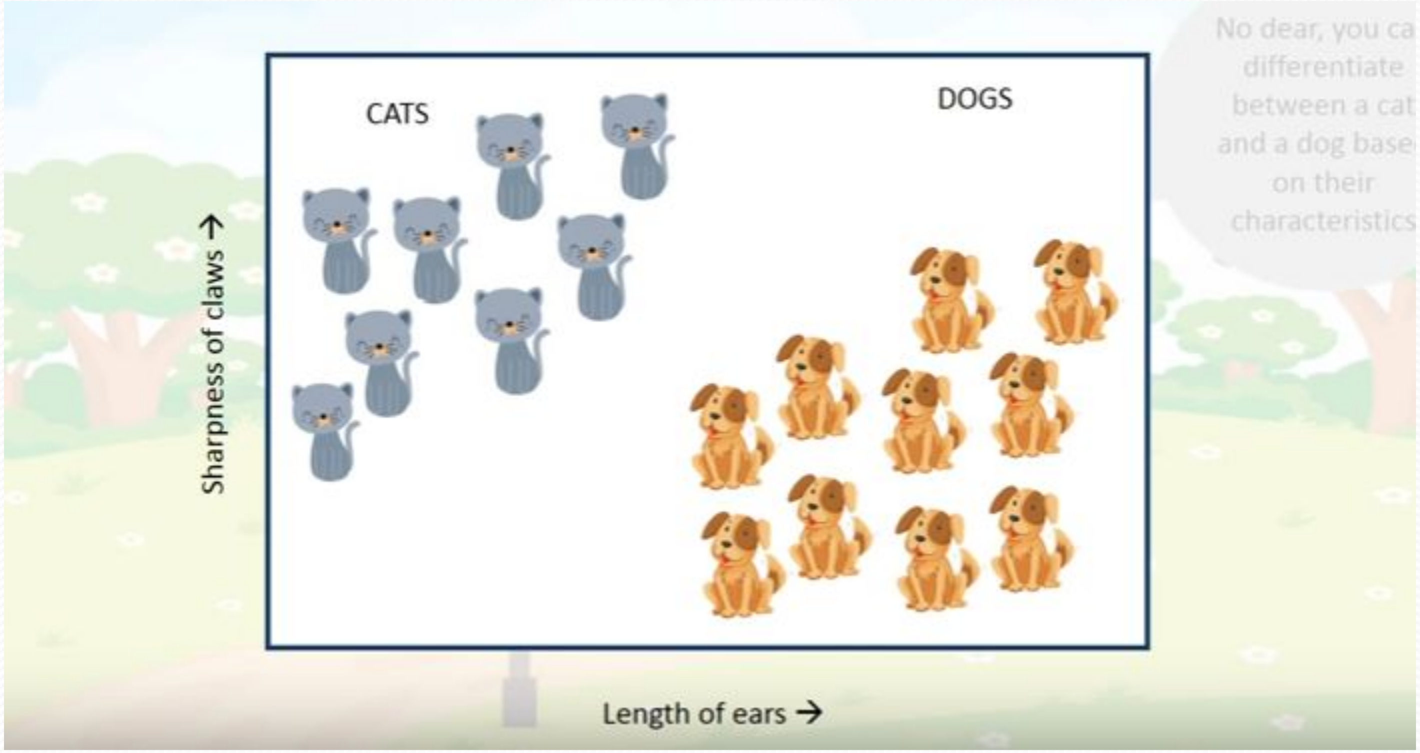
Is that a dog?

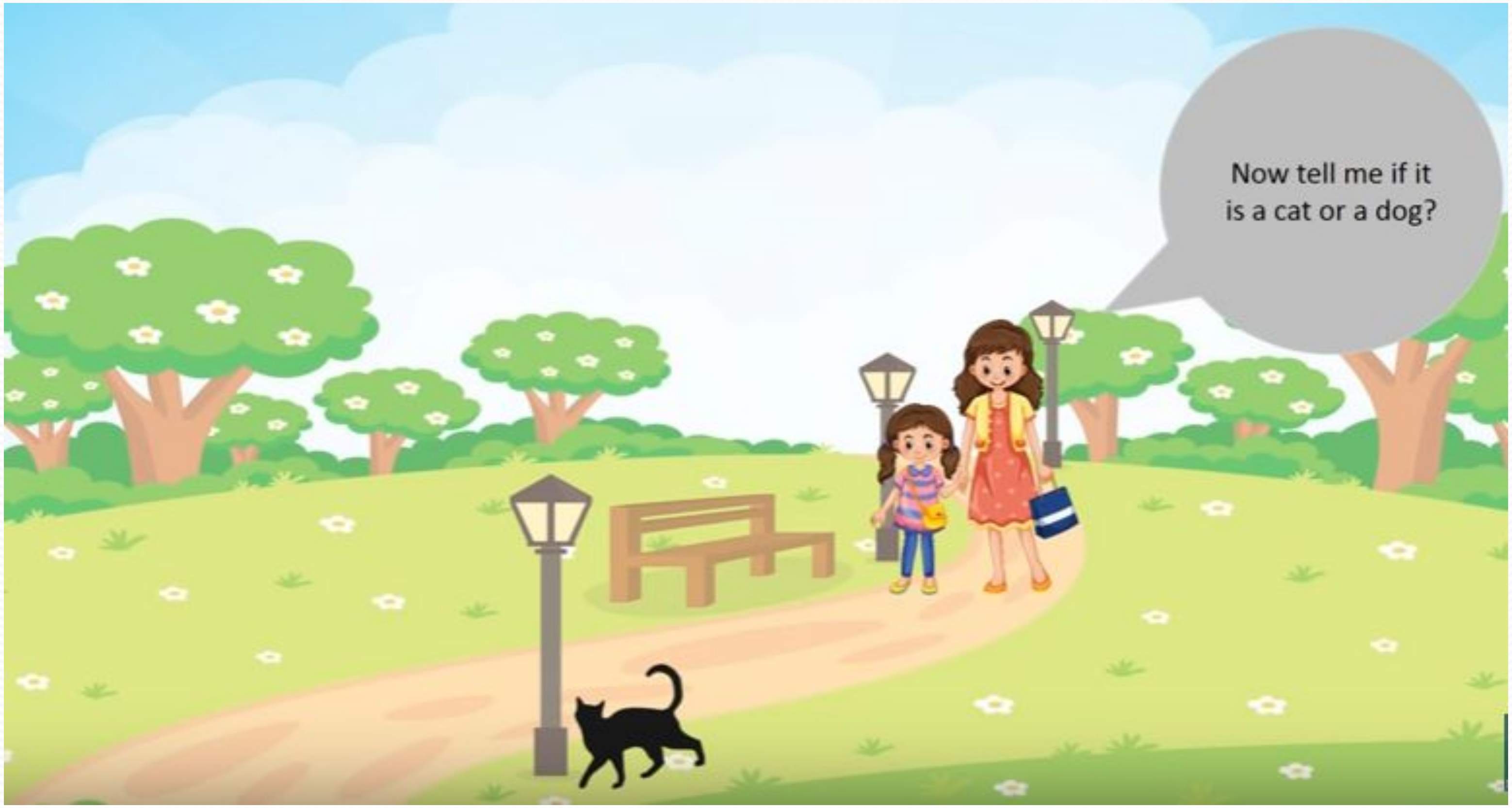
No dear, you can differentiate between a cat and a dog based on their characteristics



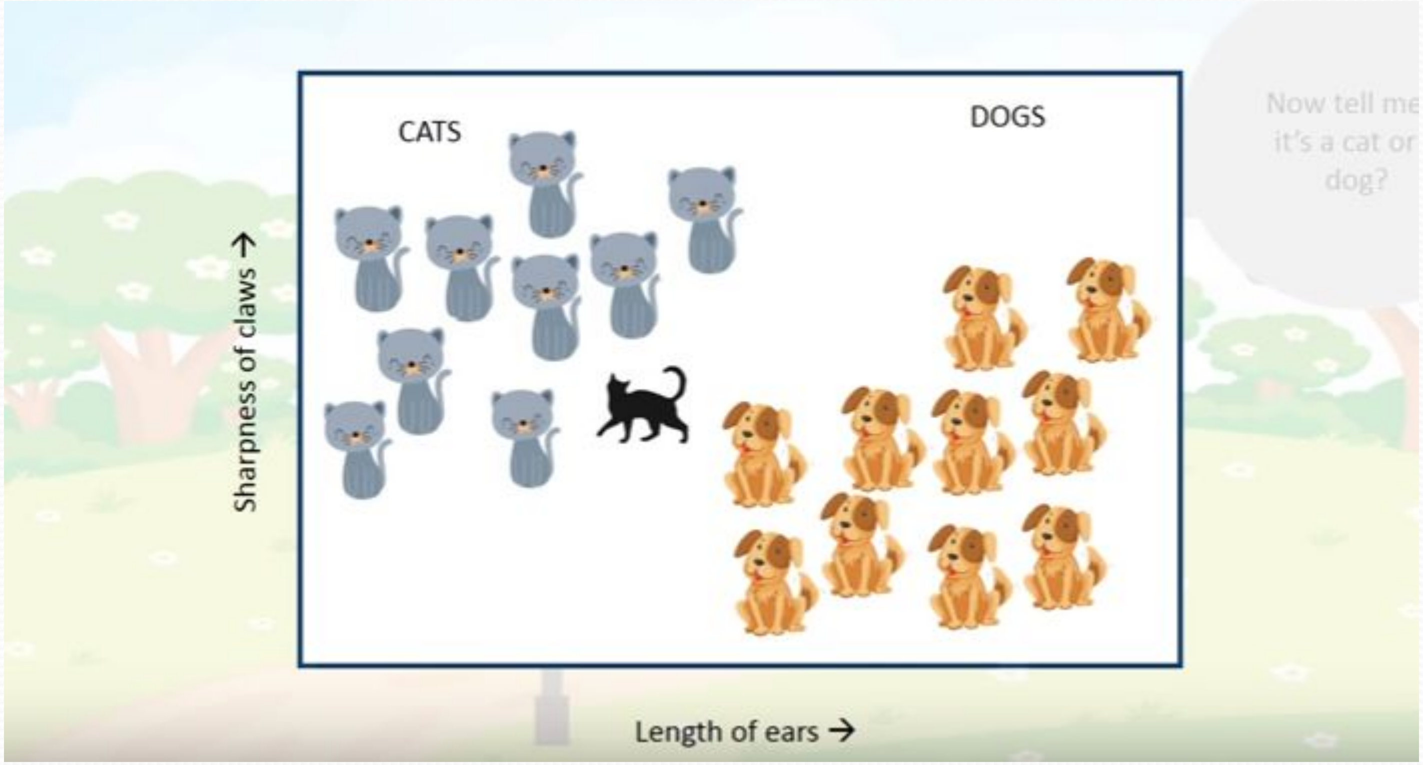
CATS	DOGS
Sharp Claws, uses to climb	Dull Claws
Smaller length of ears	Bigger length of ears
Meows and purrs	Barks
Doesn't love to play around	Loves to run around

No dear, you c
differentiate
between a ca
and a dog bas
on their
characteristic

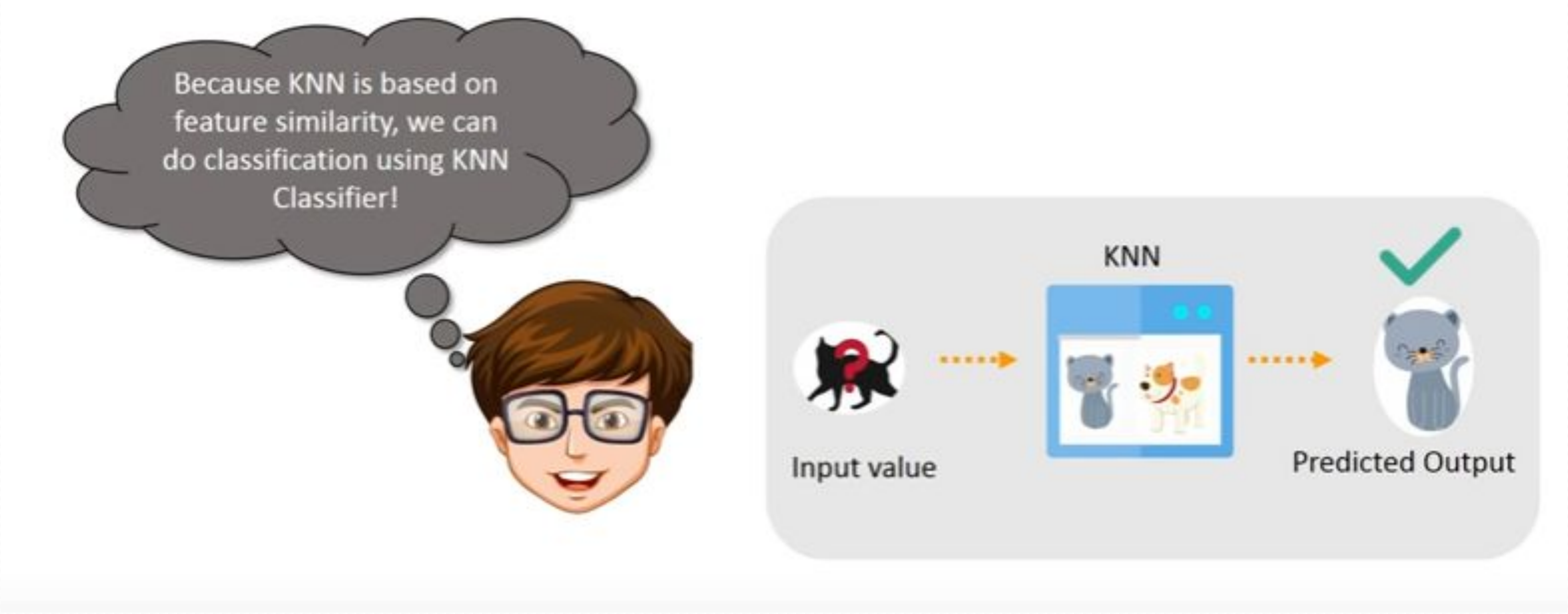

CATS
DOGS
Sharpness of claws →
Length of ears →
No dear, you ca
differentiate
between a cat
and a dog base
on their
characteristics



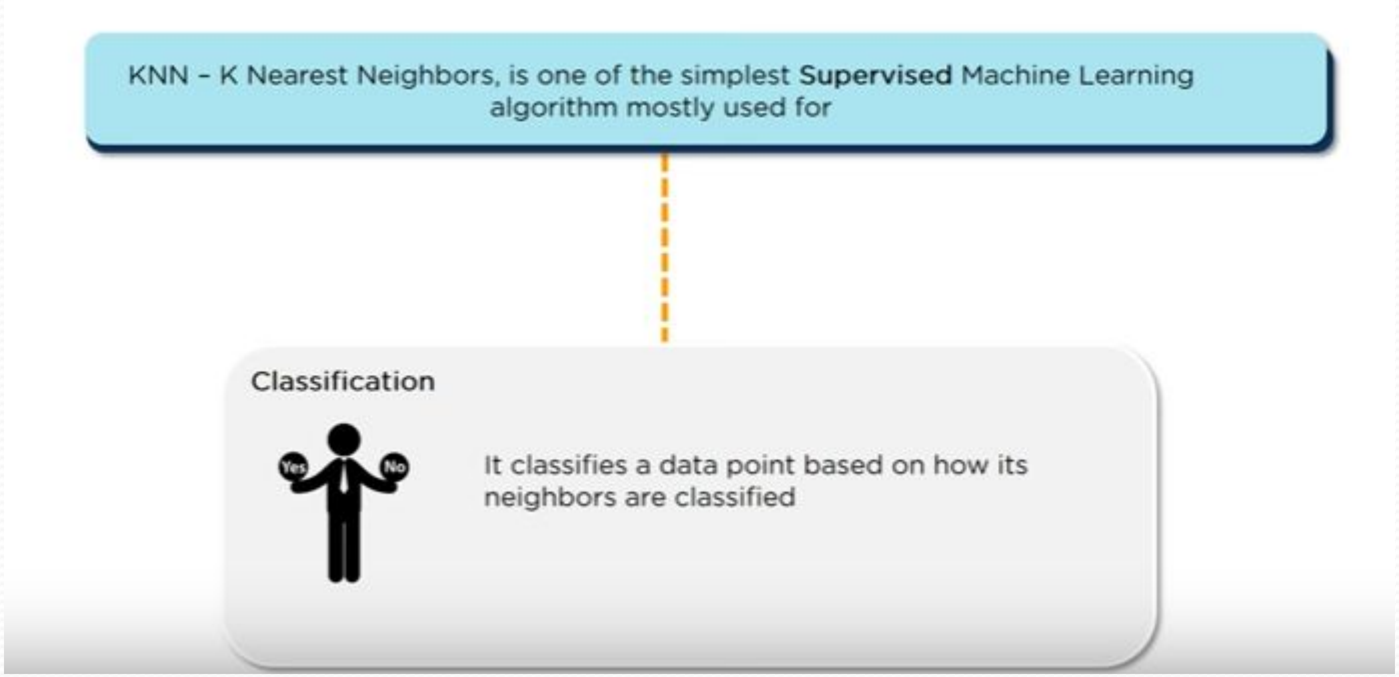
Now tell me if it is a cat or a dog?


CATS
DOGS
Sharpness of claws →
Length of ears →
Now tell me it's a cat or dog?

Why KNN?


Because KNN is based on feature similarity, we can do classification using KNN Classifier!
KNN
Input value
Predicted Output

What is knn algorithm?

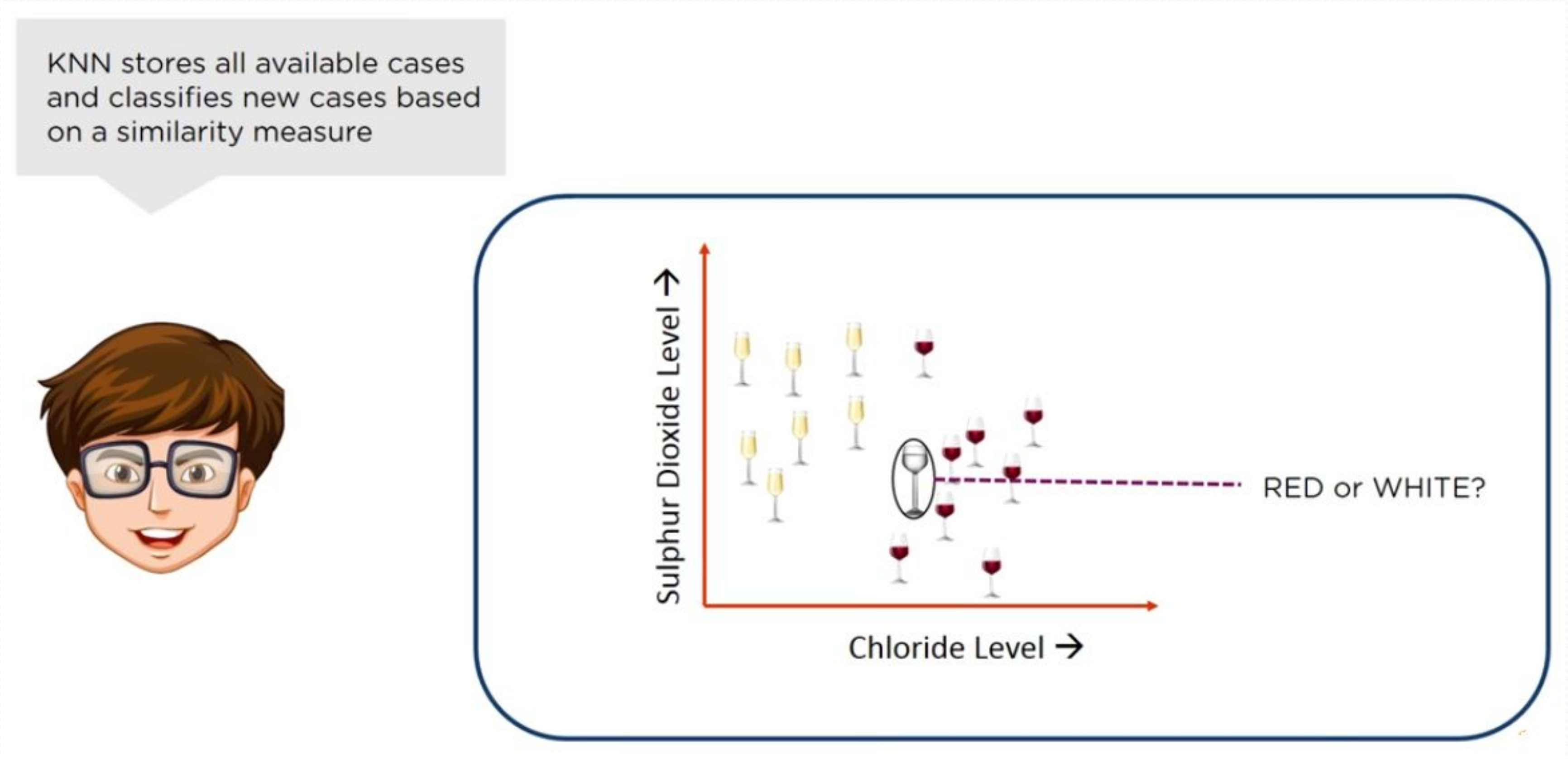


KNN – K Nearest Neighbors, is one of the simplest **Supervised** Machine Learning algorithm mostly used for

Classification

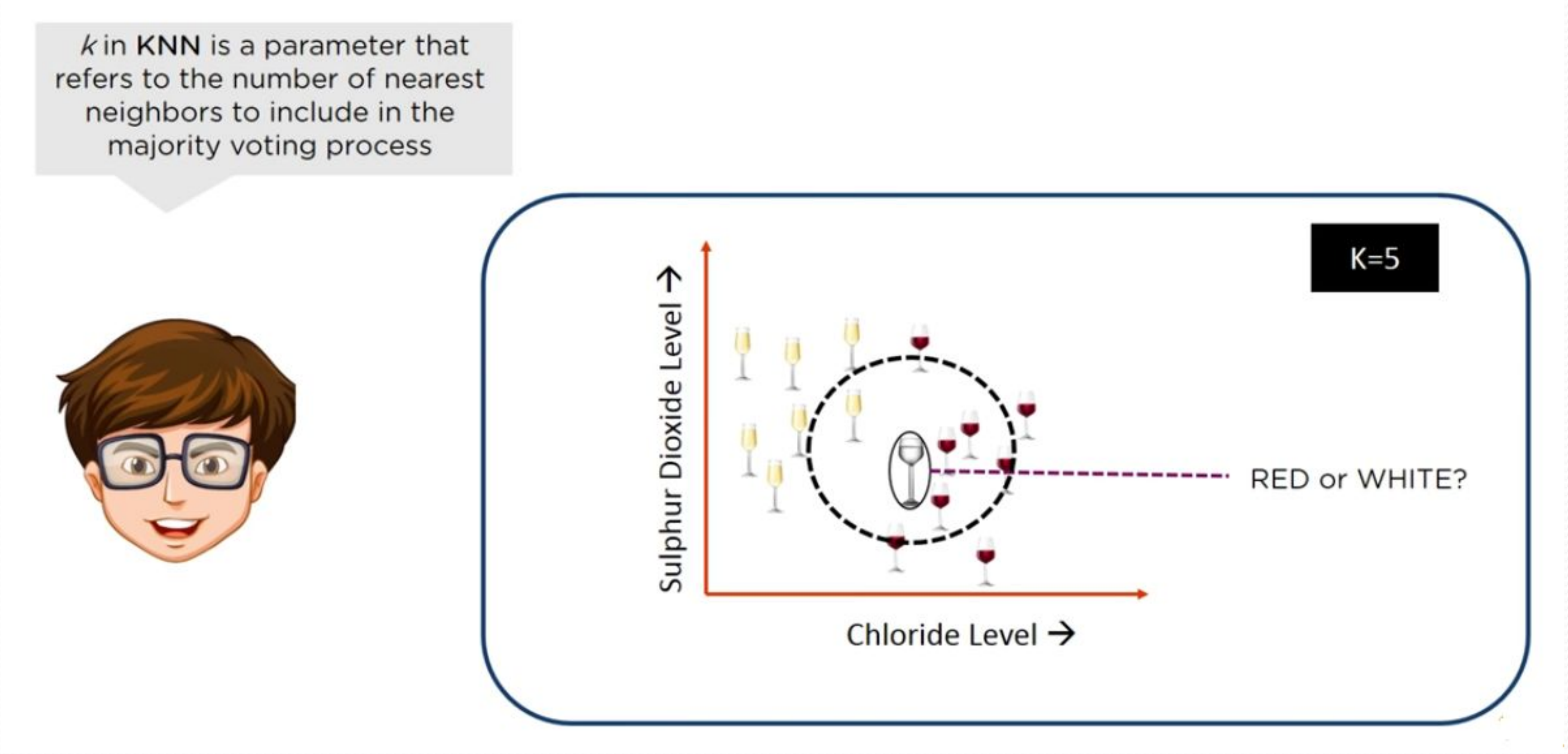
It classifies a data point based on how its neighbors are classified

What is knn algorithm?



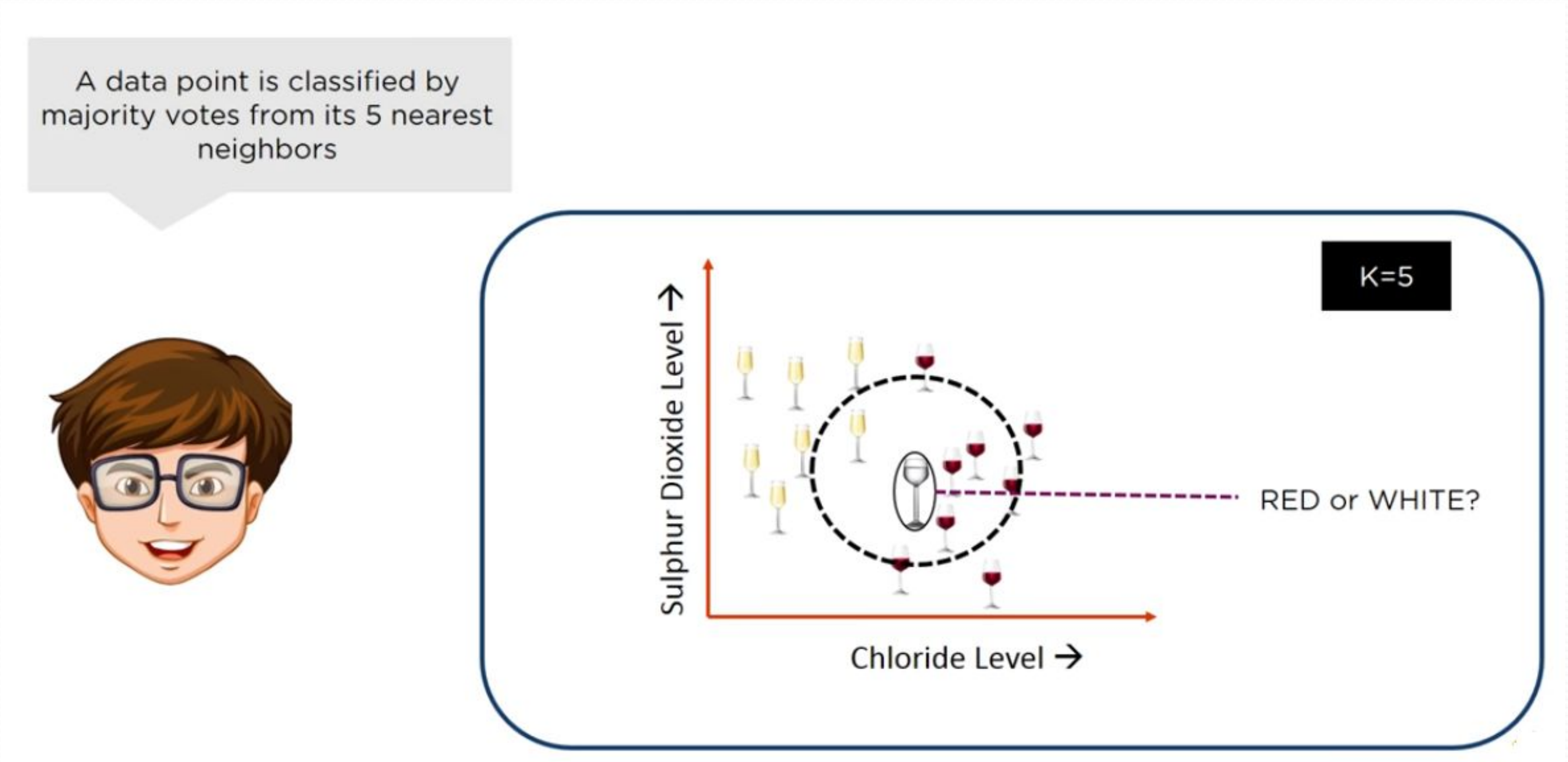
KNN stores all available cases and classifies new cases based on a similarity measure

What is knn algorithm?



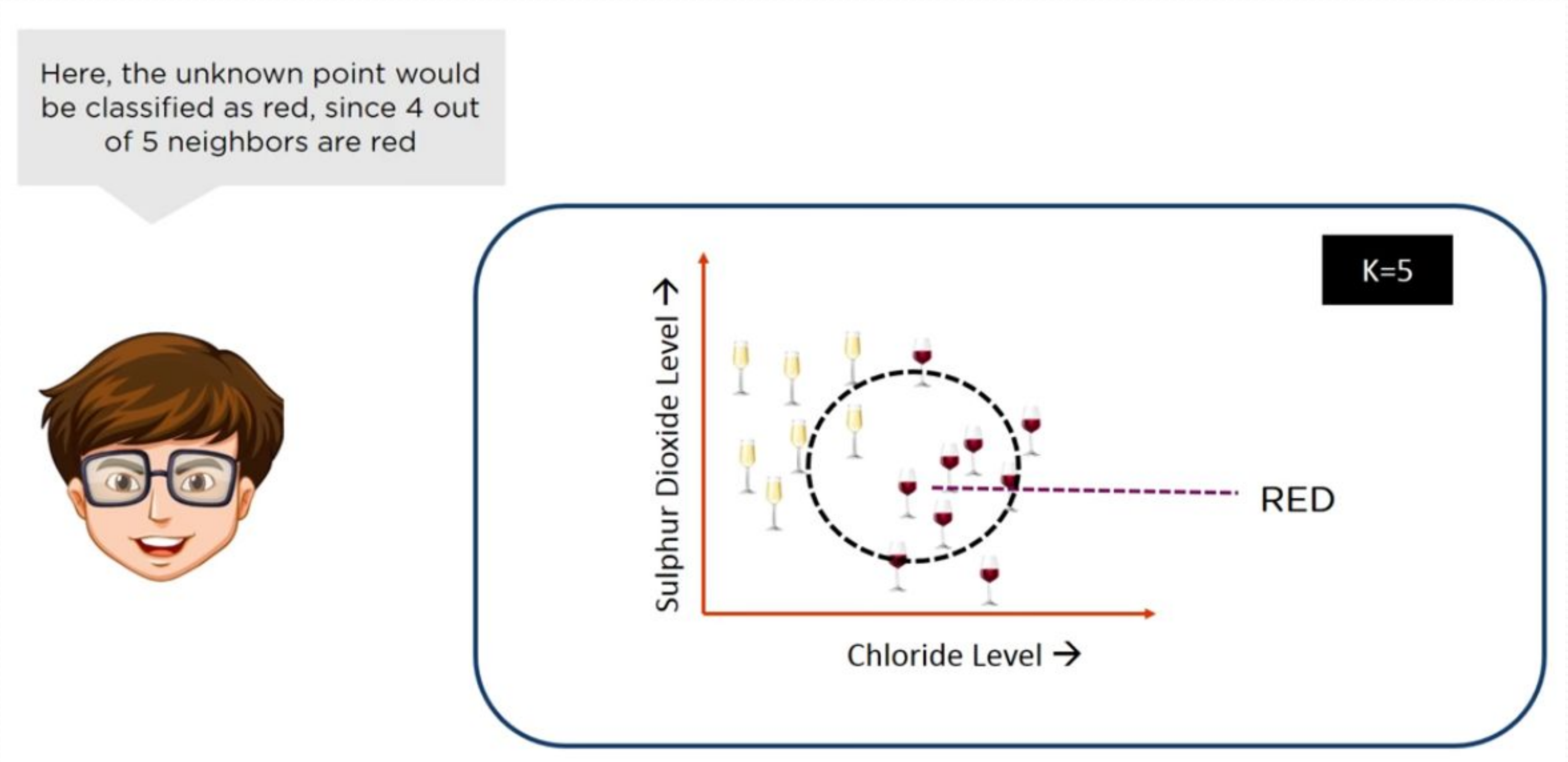
k in **KNN** is a parameter that refers to the number of nearest neighbors to include in the majority voting process

What is knn algorithm?



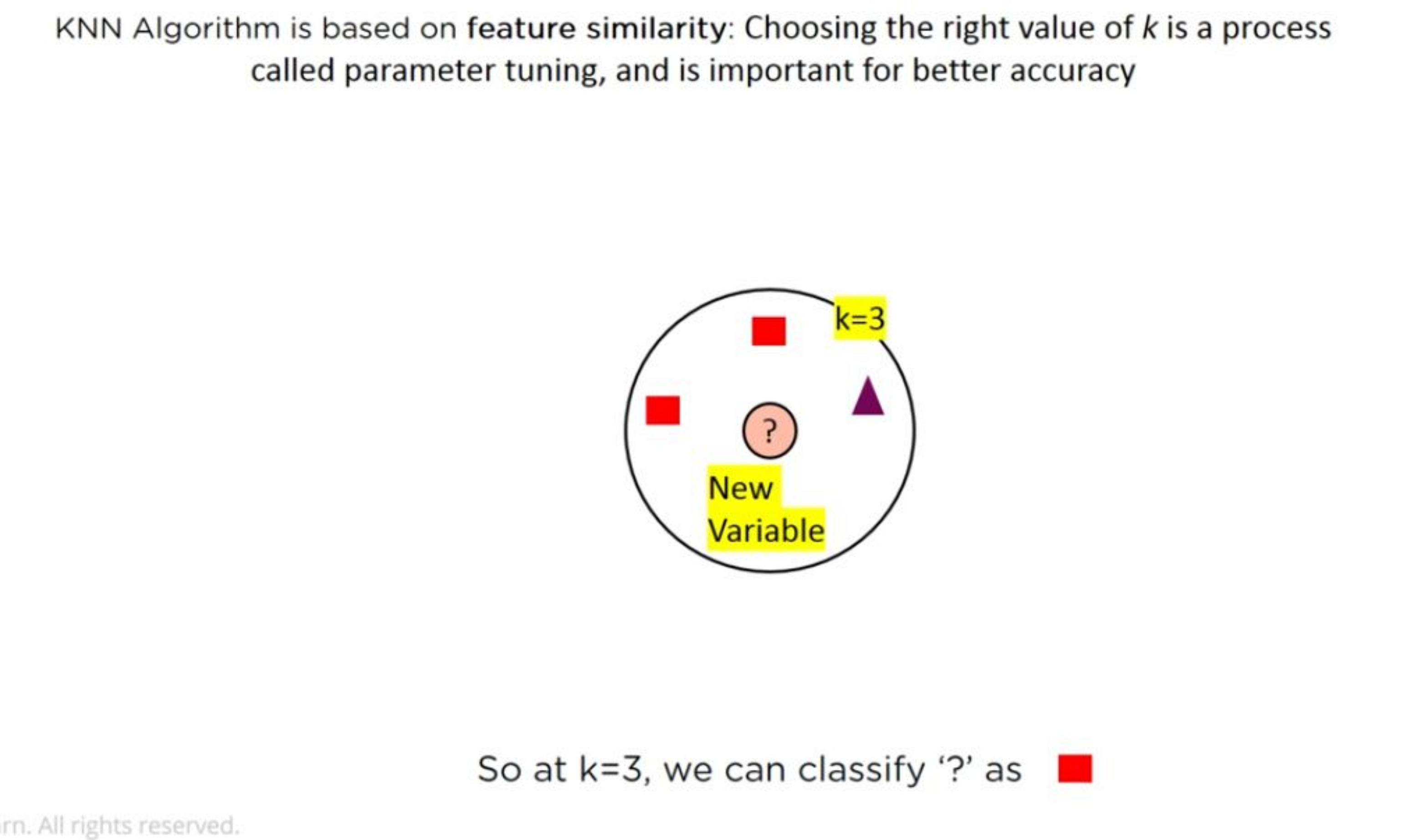
A data point is classified by majority votes from its 5 nearest neighbors

What is knn algorithm?



Here, the unknown point would be classified as red, since 4 out of 5 neighbors are red

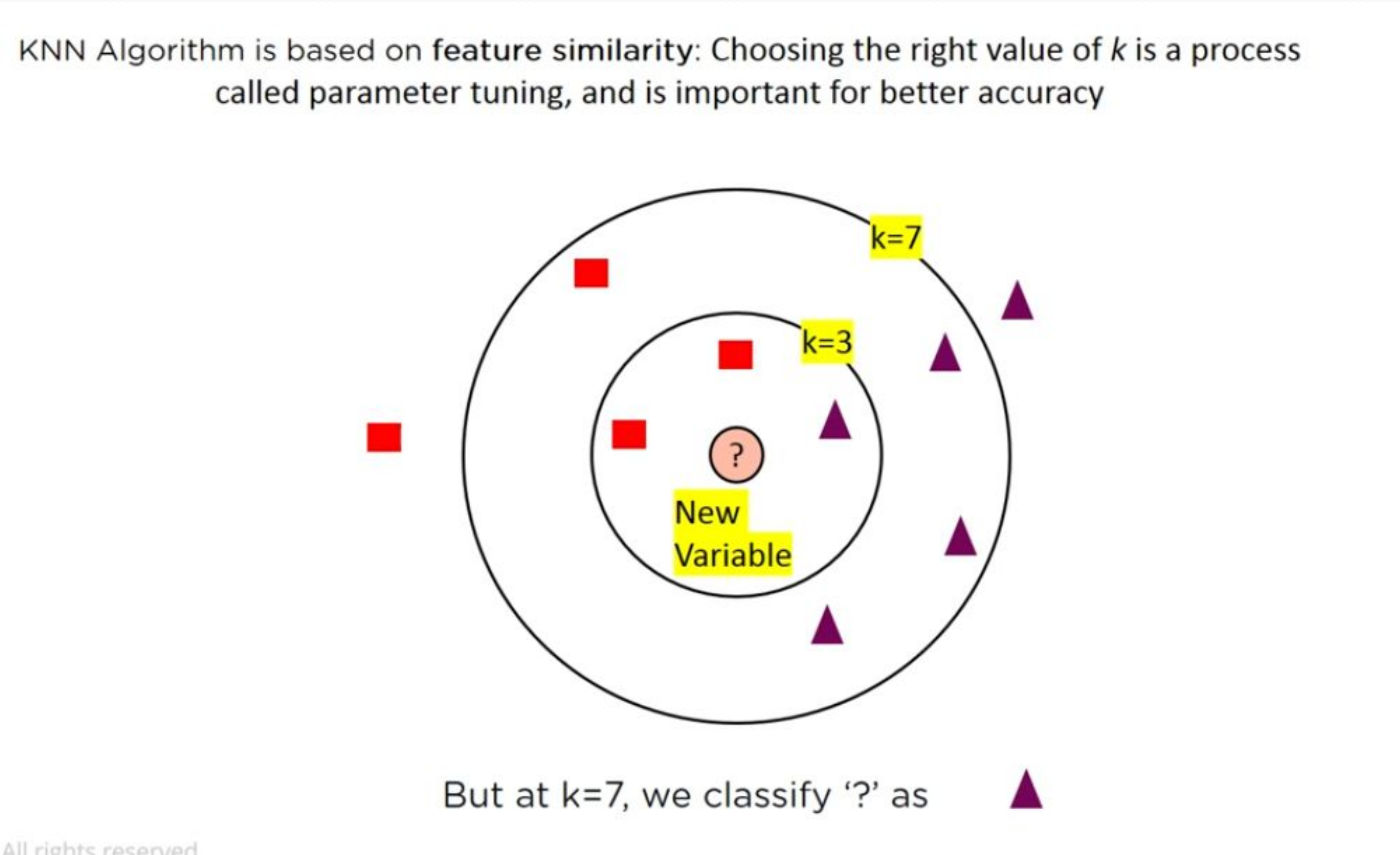
How do we choose the factor ‘k’?



KNN Algorithm is based on **feature similarity**: Choosing the right value of *k* is a process called parameter tuning, and is important for better accuracy

So at k=3, we can classify ‘?’ as 🟥

How do we choose the factor ‘k’?



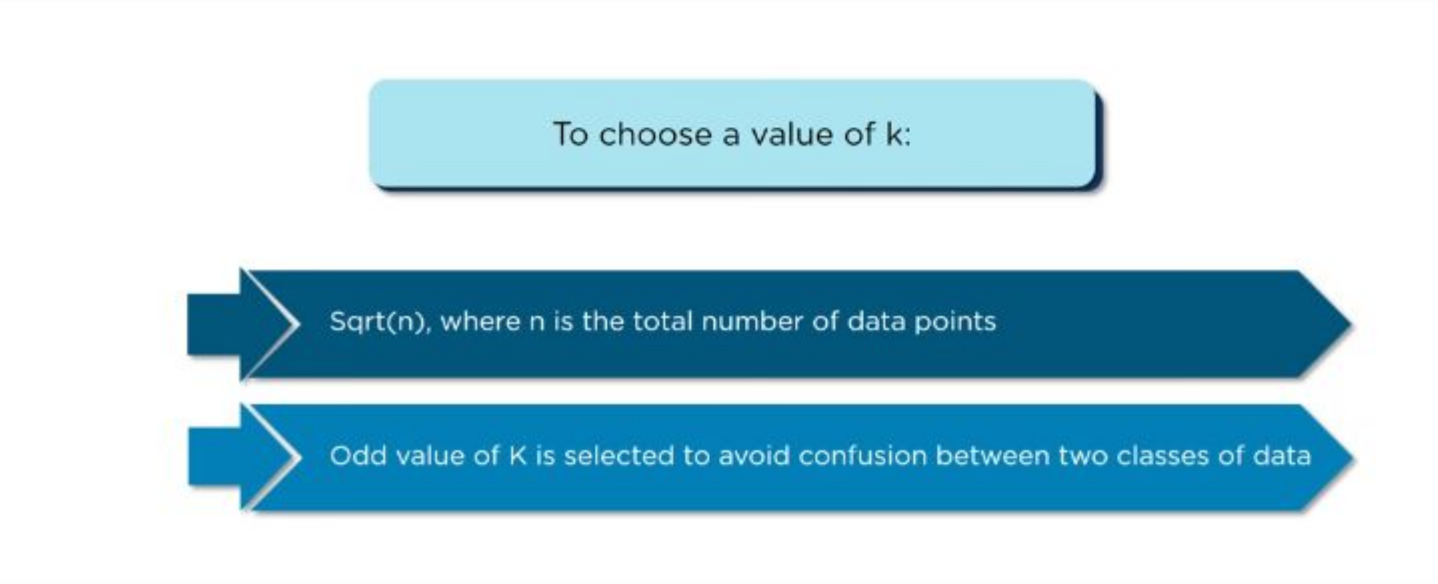
KNN Algorithm is based on **feature similarity**: Choosing the right value of *k* is a process called parameter tuning, and is important for better accuracy

But at k=7, we classify ‘?’ as ▲

How do we choose the factor ‘k’?

The class of unknown data point was 🟥 at k=3 but changed at k=7, so which k should we choose?
New Variable

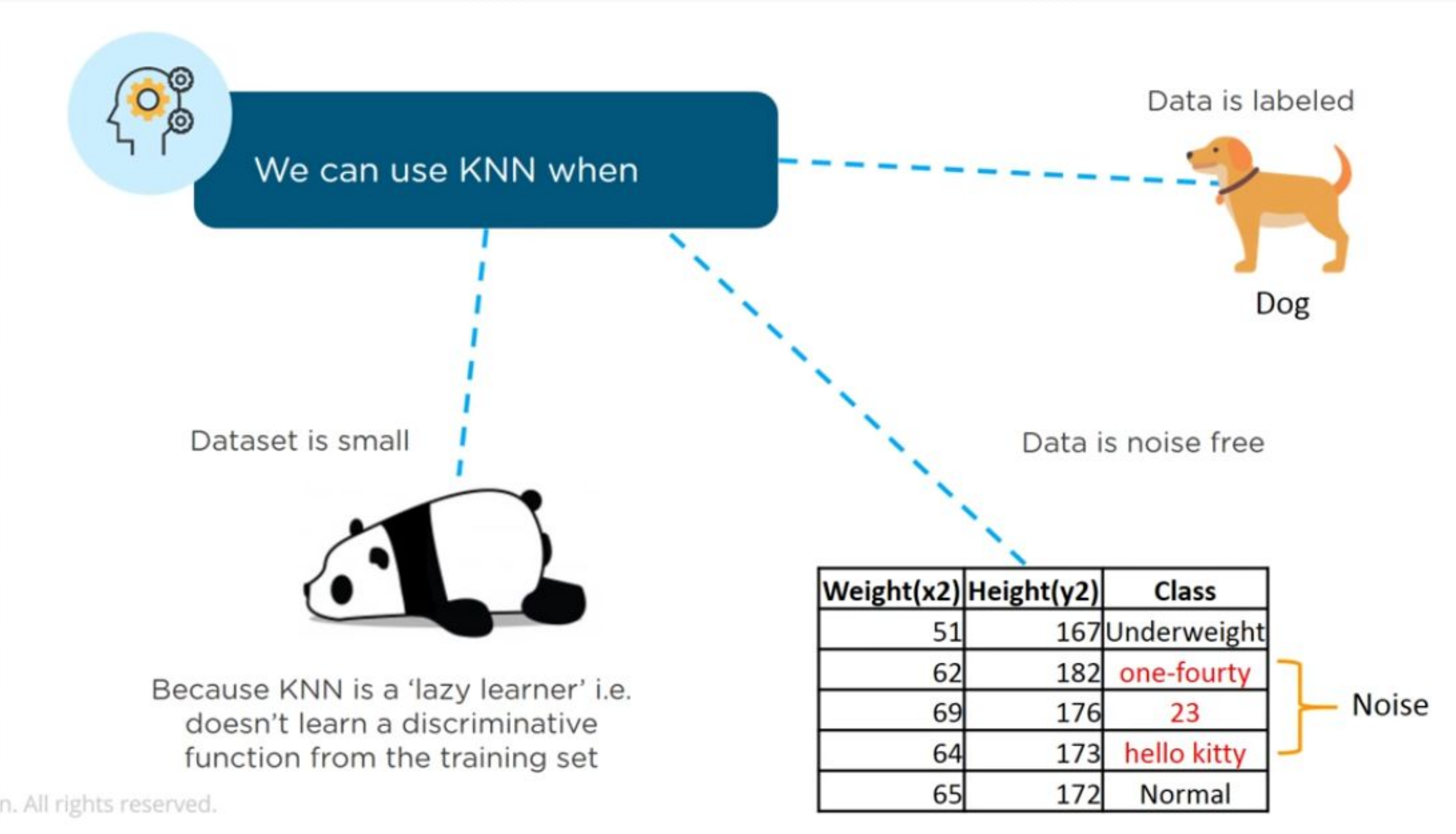
How do we choose the factor ‘k’?



To choose a value of k:

- Sqrt(n), where n is the total number of data points
- Odd value of K is selected to avoid confusion between two classes of data

When do we use knn algorithm?



We can use KNN when

Data is labeled

Dataset is small

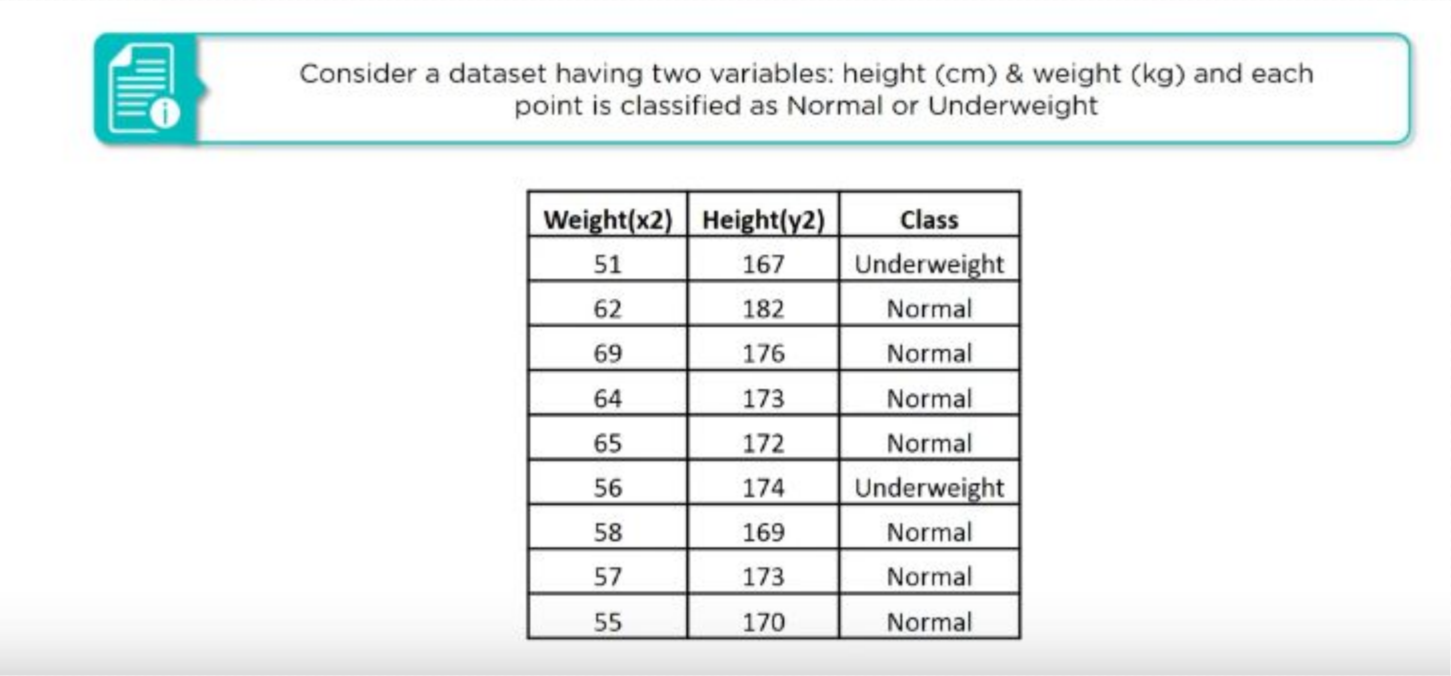
Because KNN is a 'lazy learner' i.e. doesn't learn a discriminative function from the training set

Data is noise free

Weight(x2)	Height(y2)	Class
51	167	Underweight
62	182	one-fourty
69	176	23
64	173	hello kitty
65	172	Normal

Noise

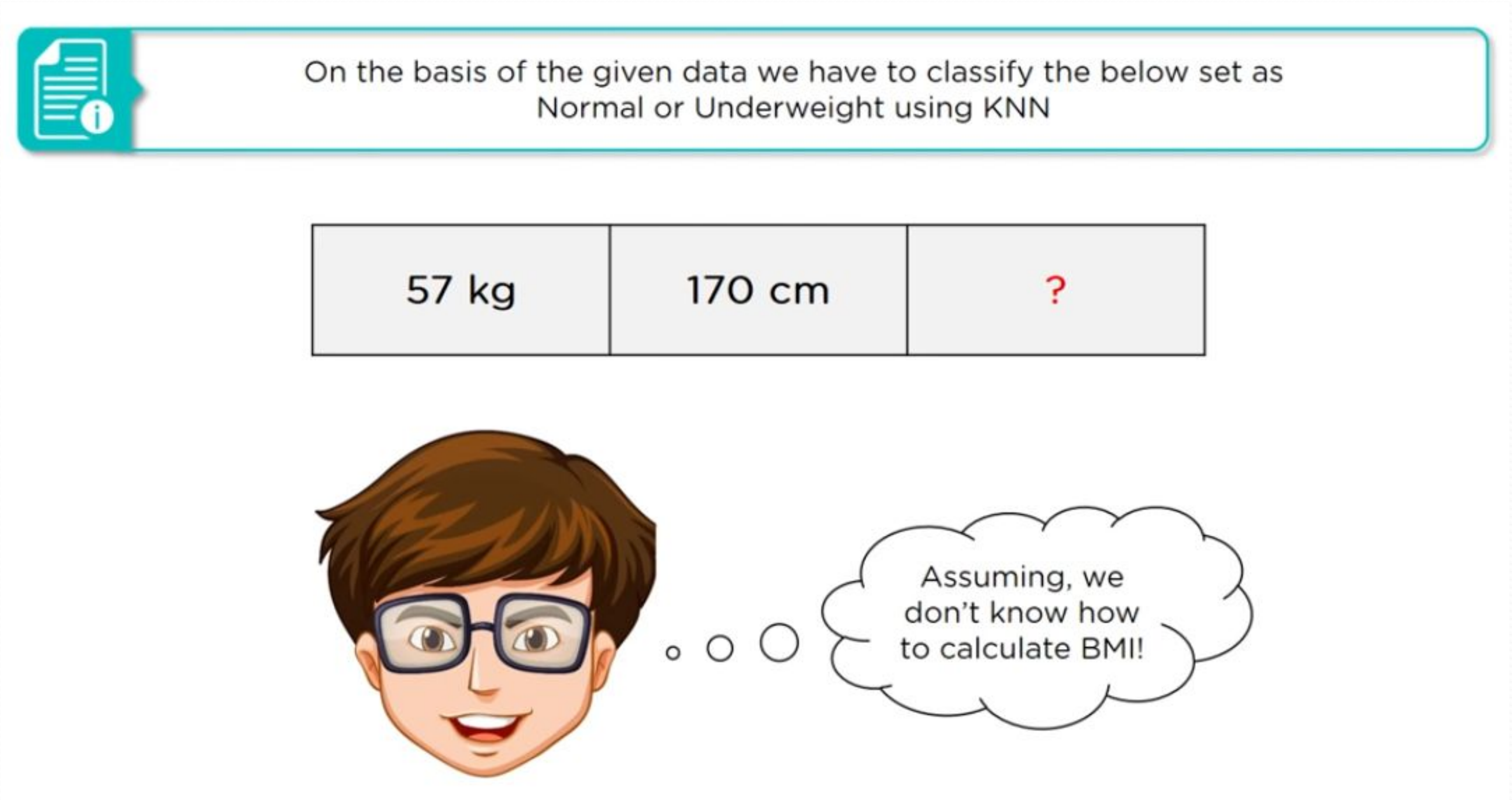
How does knn algorithm works?



Consider a dataset having two variables: height (cm) & weight (kg) and each point is classified as Normal or Underweight

Weight(x2)	Height(y2)	Class
51	167	Underweight
62	182	Normal
69	176	Normal
64	173	Normal
65	172	Normal
56	174	Underweight
58	169	Normal
57	173	Normal
55	170	Normal

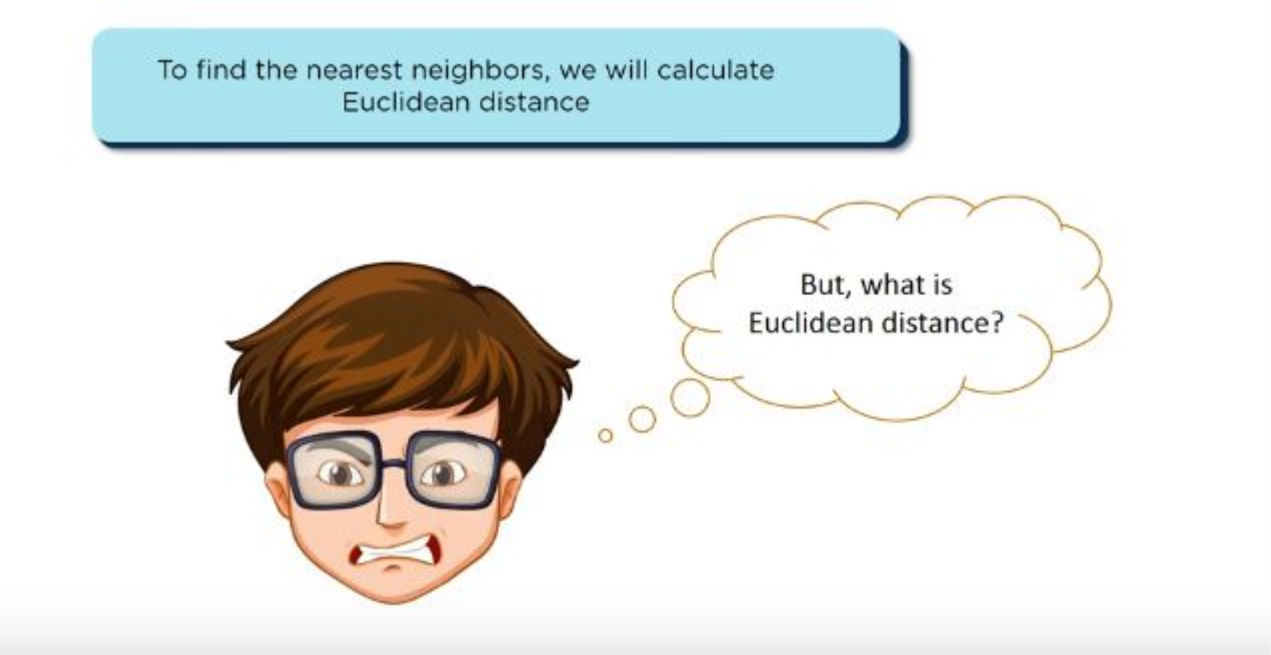
How does knn algorithm works?



On the basis of the given data we have to classify the below set as Normal or Underweight using KNN

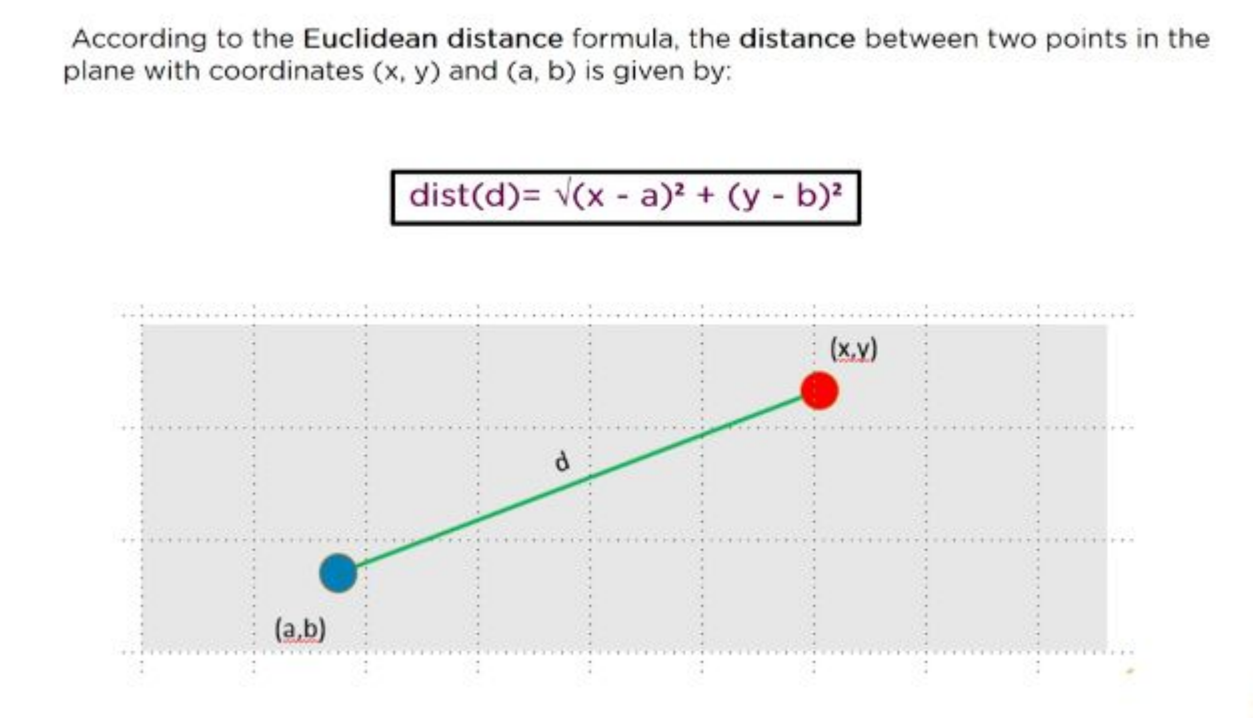
57 kg	170 cm	?

How does knn algorithm works?



To find the nearest neighbors, we will calculate Euclidean distance

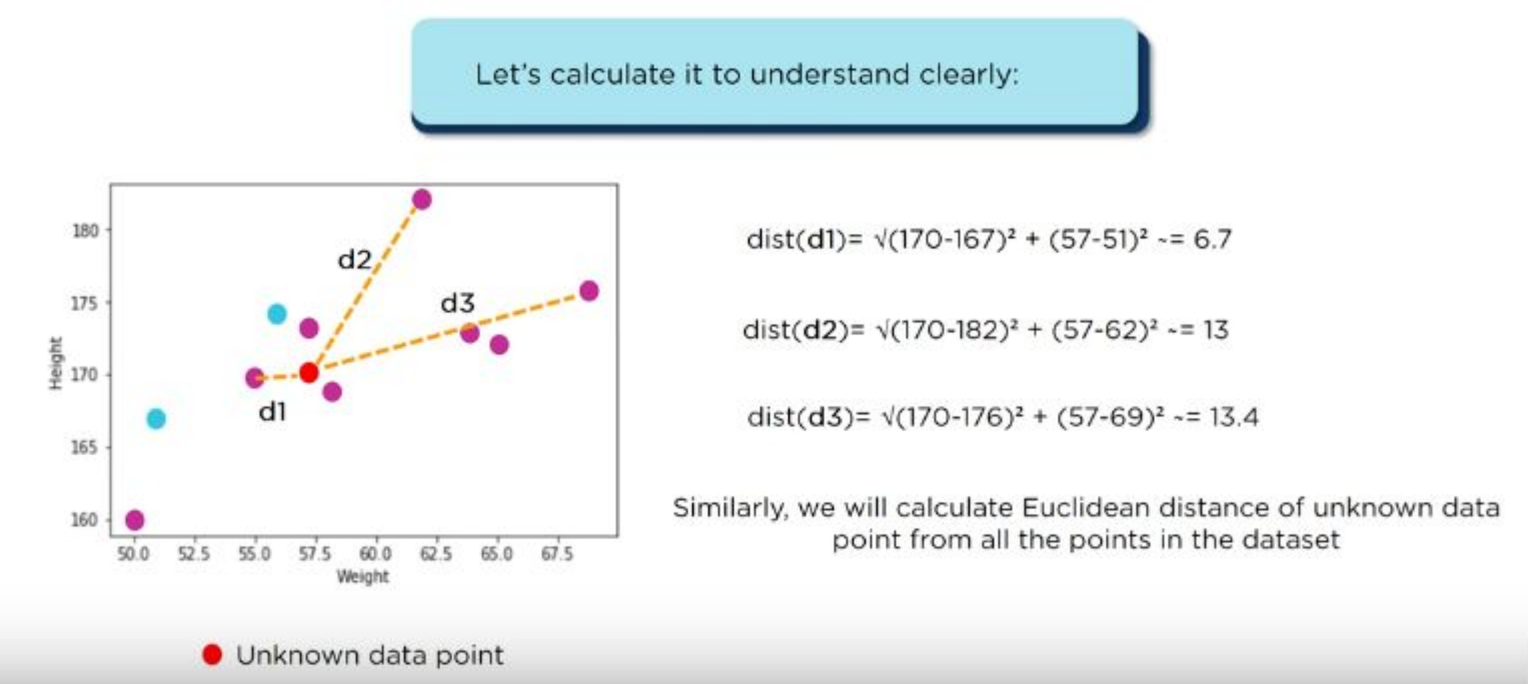
How does knn algorithm works?



According to the **Euclidean distance** formula, the **distance** between two points in the plane with coordinates (x, y) and (a, b) is given by:

$$\boxed{dist(d)= \sqrt{}(x - a)^2 + (y - b)^2}$$

How does knn algorithm works?



Let's calculate it to understand clearly:

dist(**d1**)= $\sqrt{(170-167)^2 + (57-51)^2}$ ~= 6.7

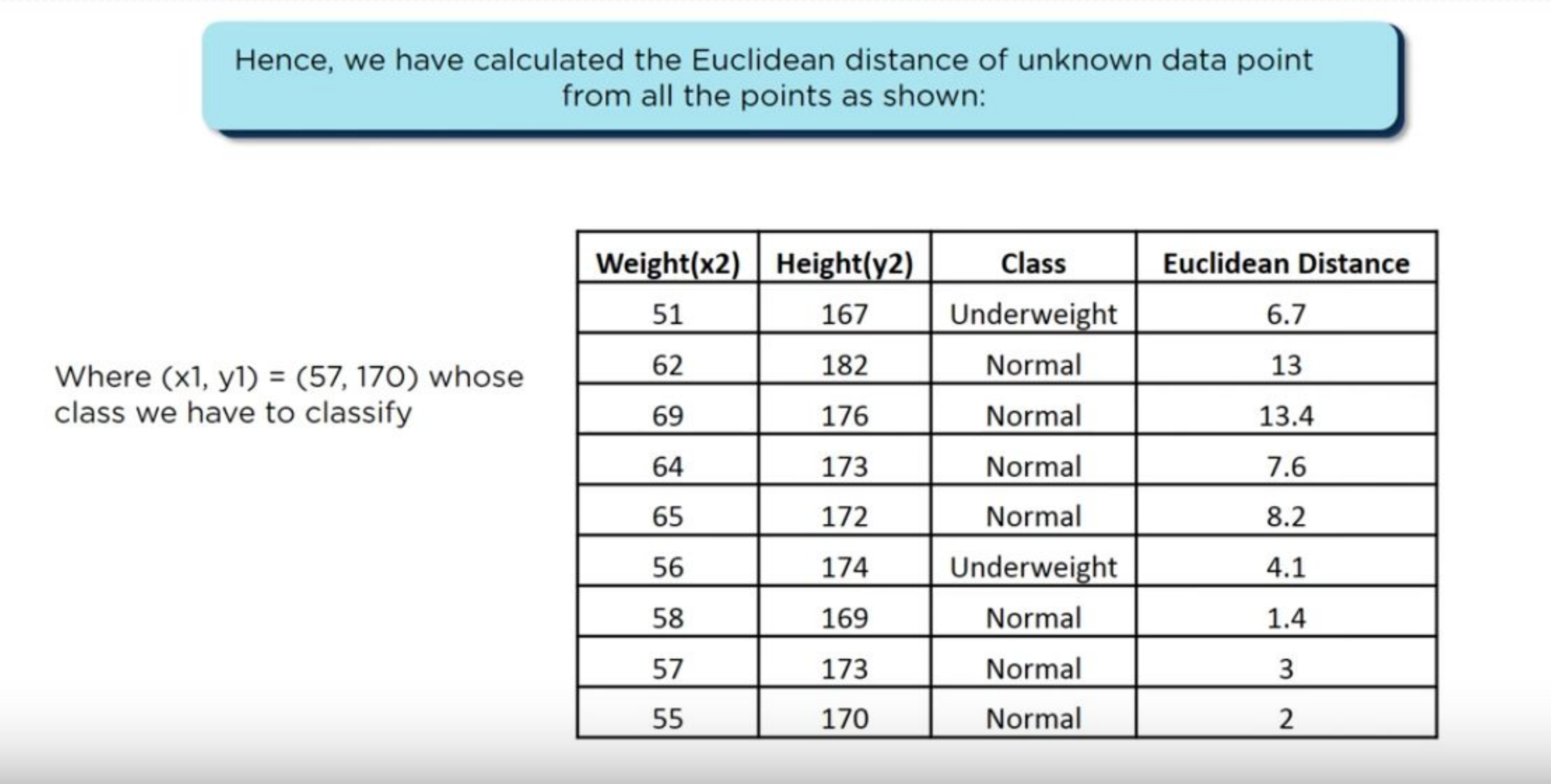
dist(**d2**)= $\sqrt{(170-182)^2 + (57-62)^2}$ ~= 13

dist(**d3**)= $\sqrt{(170-176)^2 + (57-69)^2}$ ~= 13.4

Similarly, we will calculate Euclidean distance of unknown data point from all the points in the dataset

● Unknown data point

How does knn algorithm works?

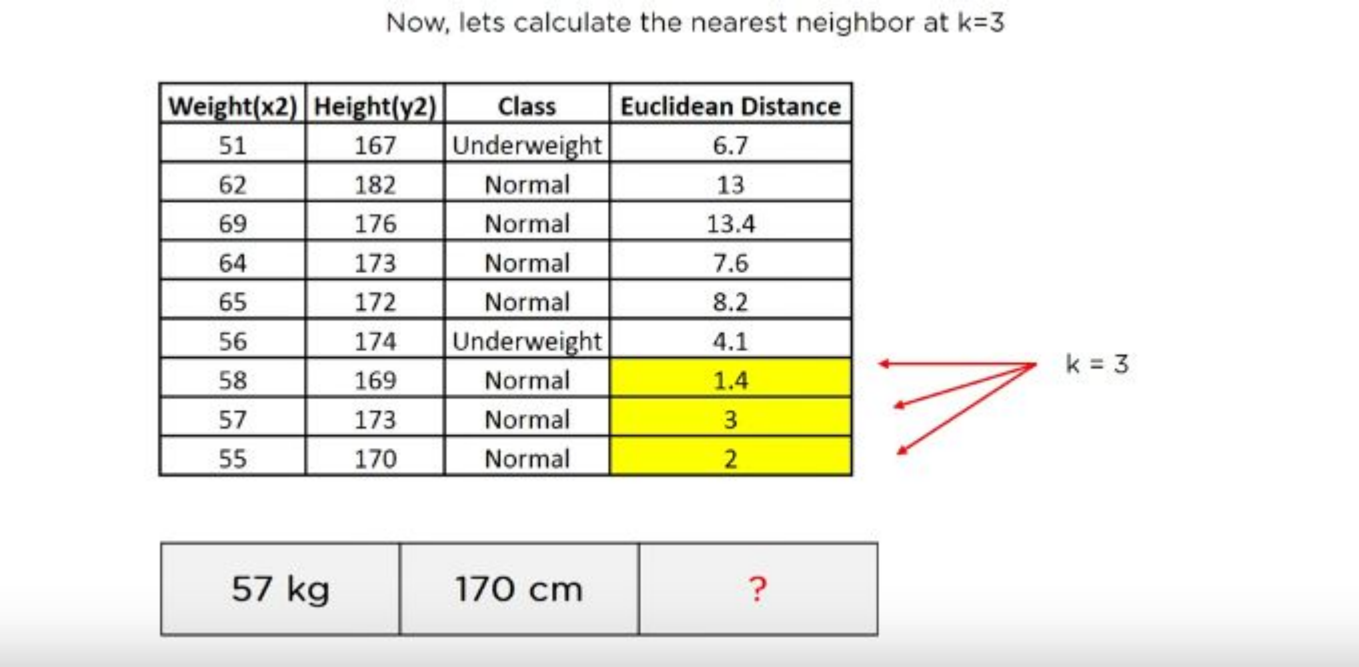


Hence, we have calculated the Euclidean distance of unknown data point from all the points as shown:

Where (x1, y1) = (57, 170) whose class we have to classify

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

How does knn algorithm works?



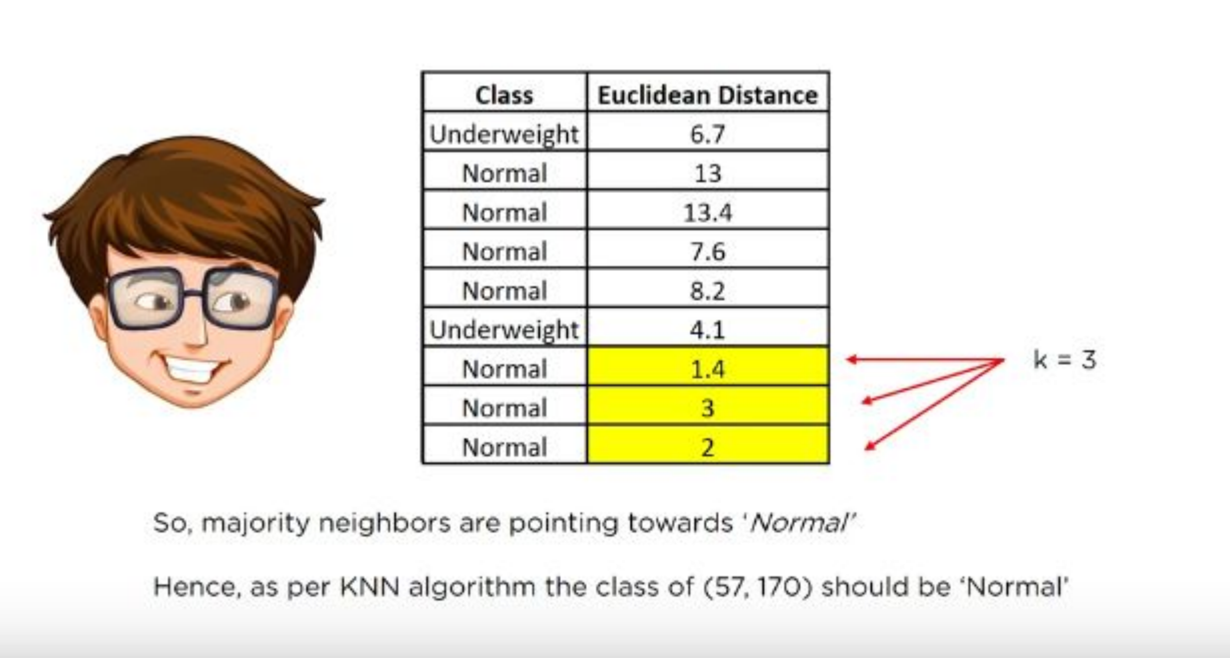
Now, lets calculate the nearest neighbor at k=3

Weight(x2)	Height(y2)	Class	Euclidean Distance
51	167	Underweight	6.7
62	182	Normal	13
69	176	Normal	13.4
64	173	Normal	7.6
65	172	Normal	8.2
56	174	Underweight	4.1
58	169	Normal	1.4
57	173	Normal	3
55	170	Normal	2

k = 3

57 kg	170 cm	?

How does knn algorithm works?

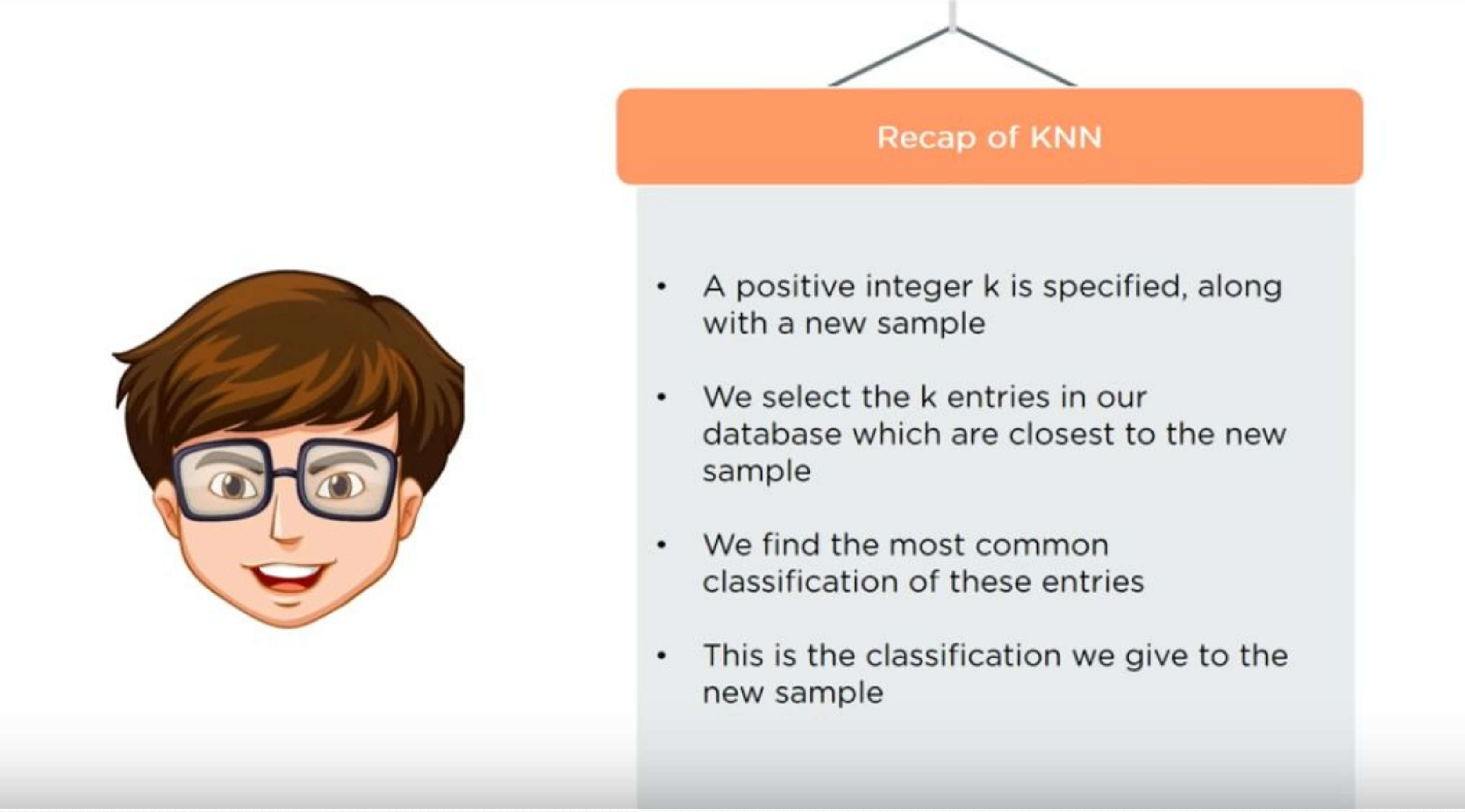


Class	Euclidean Distance
Underweight	6.7
Normal	13
Normal	13.4
Normal	7.6
Normal	8.2
Underweight	4.1
Normal	1.4
Normal	3
Normal	2

So, majority neighbors are pointing towards '*Normal*'

Hence, as per KNN algorithm the class of (57, 170) should be 'Normal'

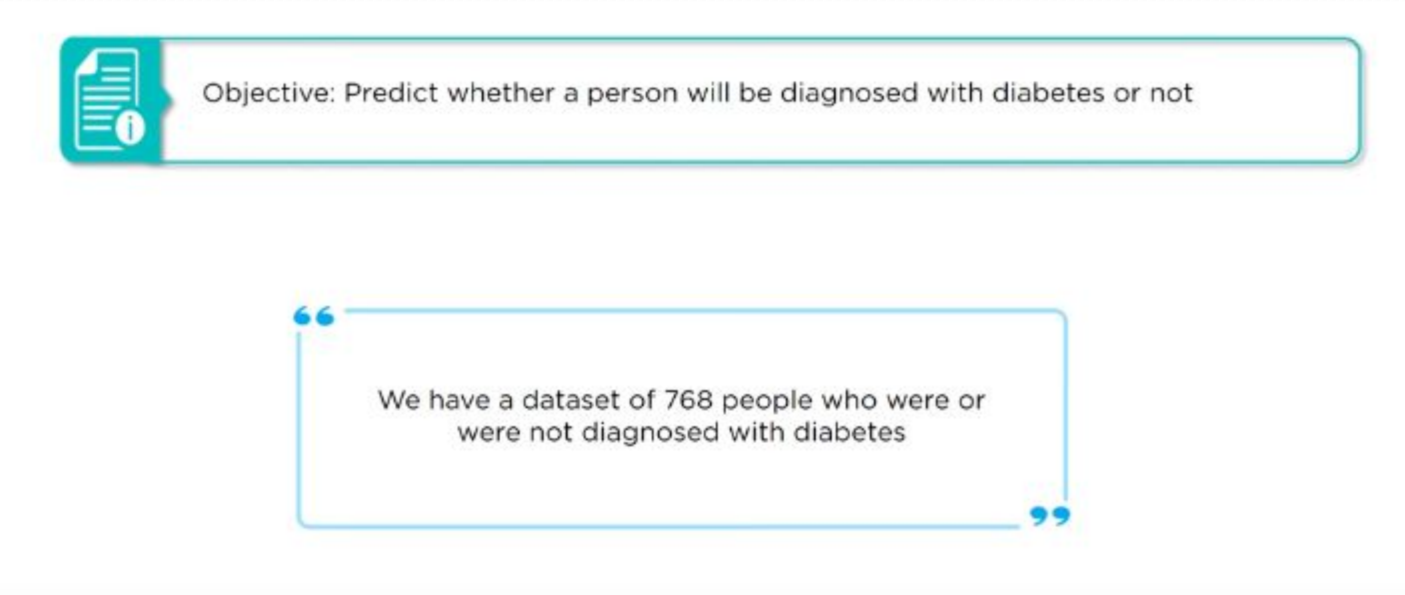
Recap of knn



Recap of KNN

- A positive integer k is specified, along with a new sample
- We select the k entries in our database which are closest to the new sample
- We find the most common classification of these entries
- This is the classification we give to the new sample

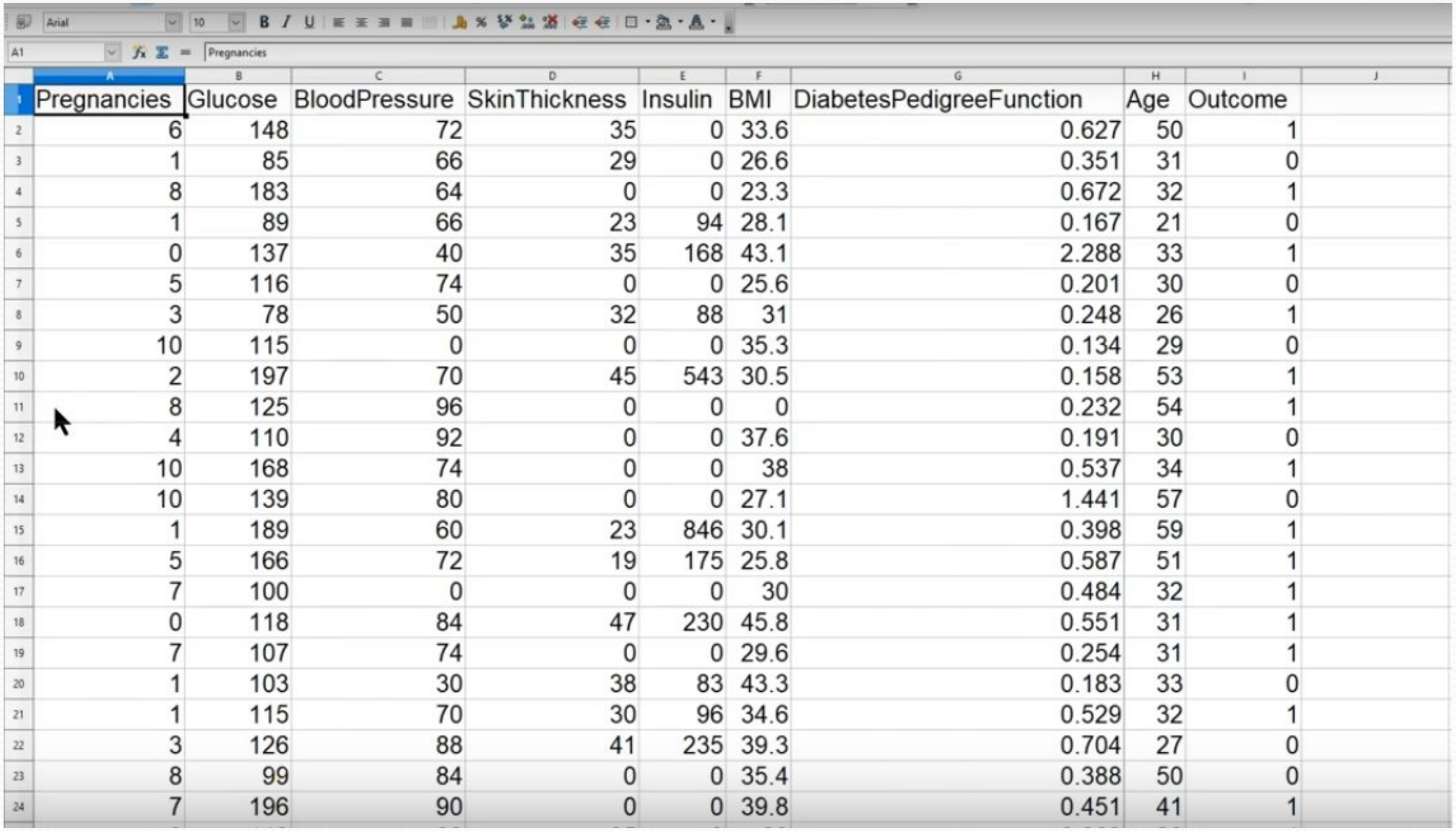
USE Case: predict diabetes



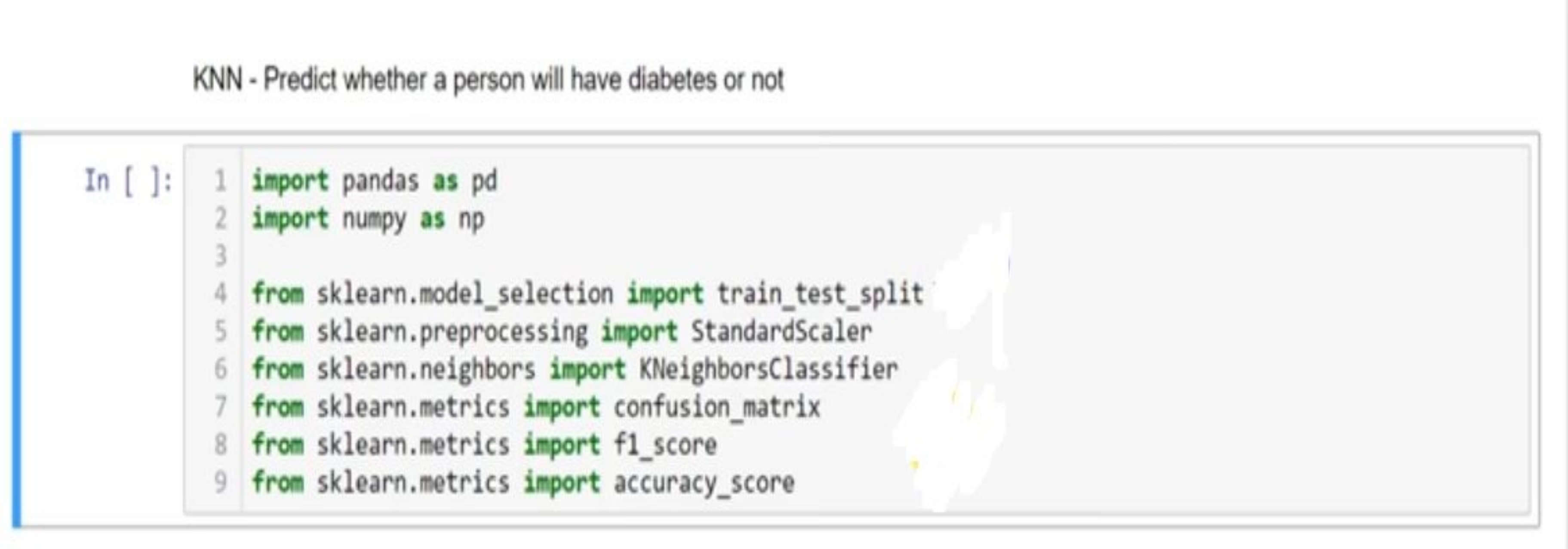
Objective: Predict whether a person will be diagnosed with diabetes or not

> We have a dataset of 768 people who were or were not diagnosed with diabetes

USE Case: predict diabetes



	A	B	C	D	E	F	G	H	I	J
1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome	
2	6	148	72	35	0	33.6	0.627	50	1	
3	1	85	66	29	0	26.6	0.351	31	0	
4	8	183	64	0	0	23.3	0.672	32	1	
5	1	89	66	23	94	28.1	0.167	21	0	
6	0	137	40	35	168	43.1	2.288	33	1	
7	5	116	74	0	0	25.6	0.201	30	0	
8	3	78	50	32	88	31	0.248	26	1	
9	10	115	0	0	0	35.3	0.134	29	0	
10	2	197	70	45	543	30.5	0.158	53	1	
11	8	125	96	0	0	0	0.232	54	1	
12	4	110	92	0	0	37.6	0.191	30	0	
13	10	168	74	0	0	38	0.537	34	1	
14	10	139	80	0	0	27.1	1.441	57	0	
15	1	189	60	23	846	30.1	0.398	59	1	
16	5	166	72	19	175	25.8	0.587	51	1	
17	7	100	0	0	0	30	0.484	32	1	
18	0	118	84	47	230	45.8	0.551	31	1	
19	7	107	74	0	0	29.6	0.254	31	1	
20	1	103	30	38	83	43.3	0.183	33	0	
21	1	115	70	30	96	34.6	0.529	32	1	
22	3	126	88	41	235	39.3	0.704	27	0	
23	8	99	84	0	0	35.4	0.388	50	0	
24	7	196	90	0	0	39.8	0.451	41	1	



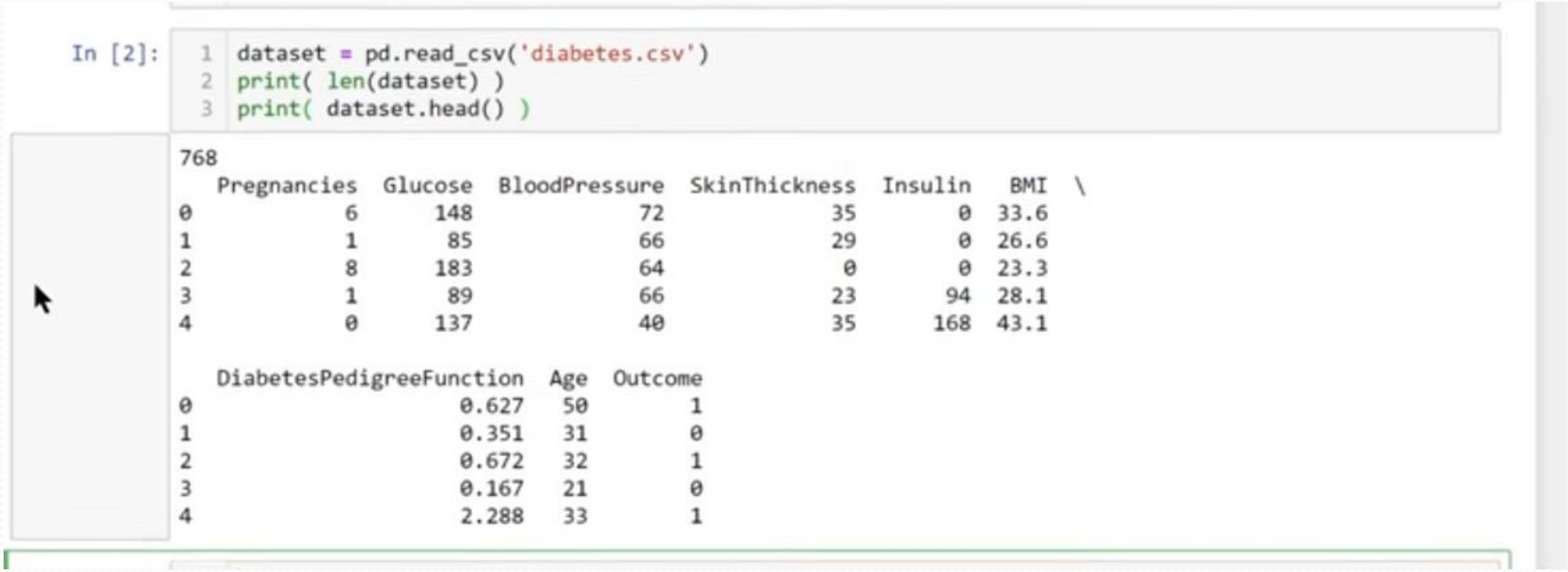
KNN - Predict whether a person will have diabetes or not

In [ ]:

```
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
```

USE Case: predict diabetes



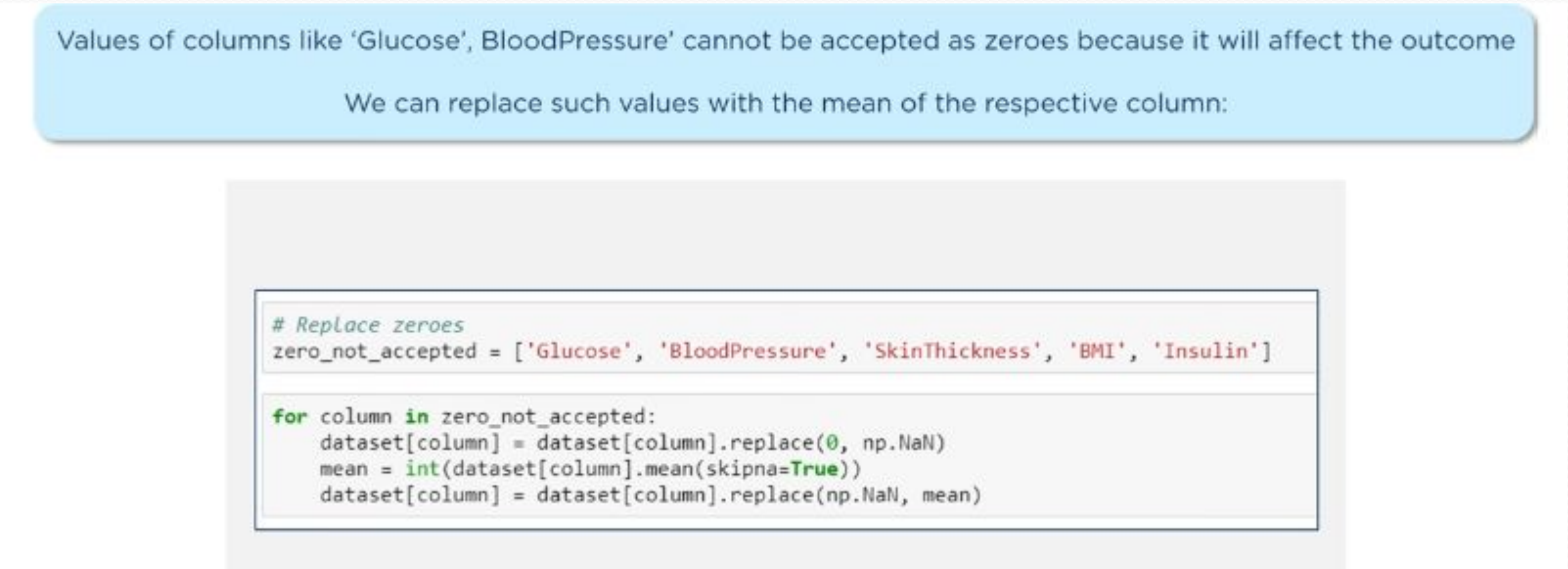
In [2]:

```python
dataset = pd.read_csv('diabetes.csv')
print( len(dataset) )
print( dataset.head() )
```

```
768
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

USE Case: predict diabetes



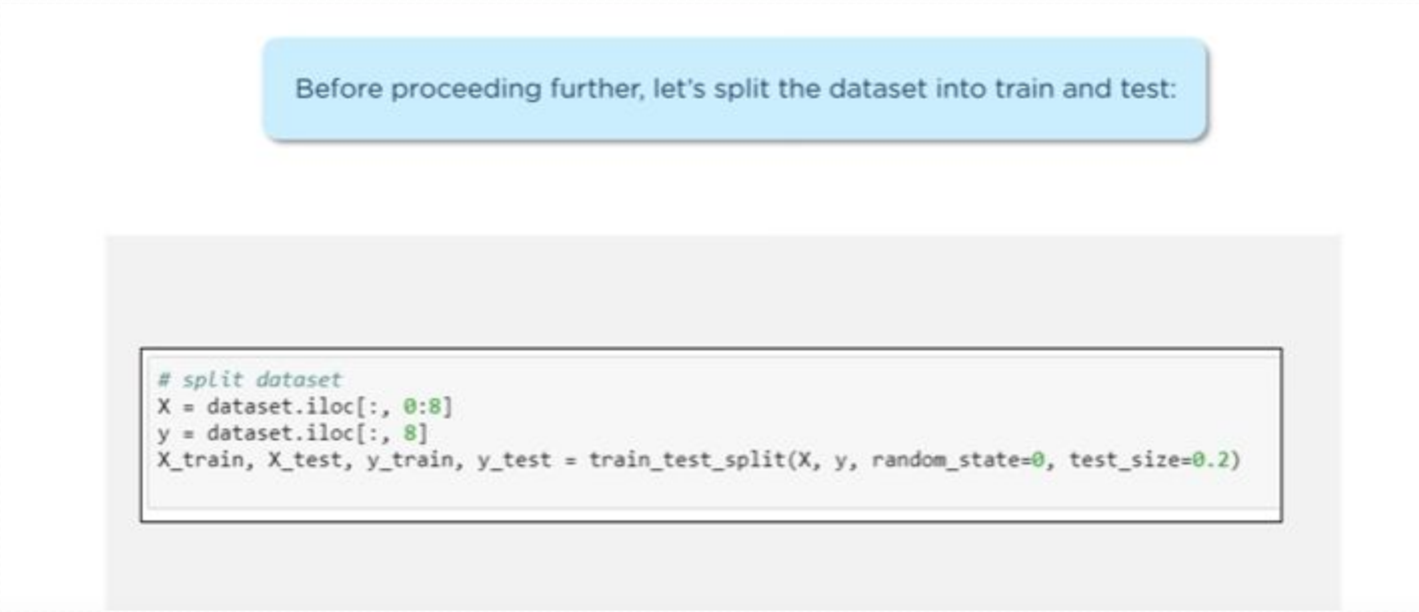
Values of columns like 'Glucose', BloodPressure' cannot be accepted as zeroes because it will affect the outcome

We can replace such values with the mean of the respective column:

```
# Replace zeroes
zero_not_accepted = ['Glucose', 'BloodPressure', 'SkinThickness', 'BMI', 'Insulin']
```

```
for column in zero_not_accepted:
    dataset[column] = dataset[column].replace(0, np.NaN)
    mean = int(dataset[column].mean(skipna=True))
    dataset[column] = dataset[column].replace(np.NaN, mean)
```

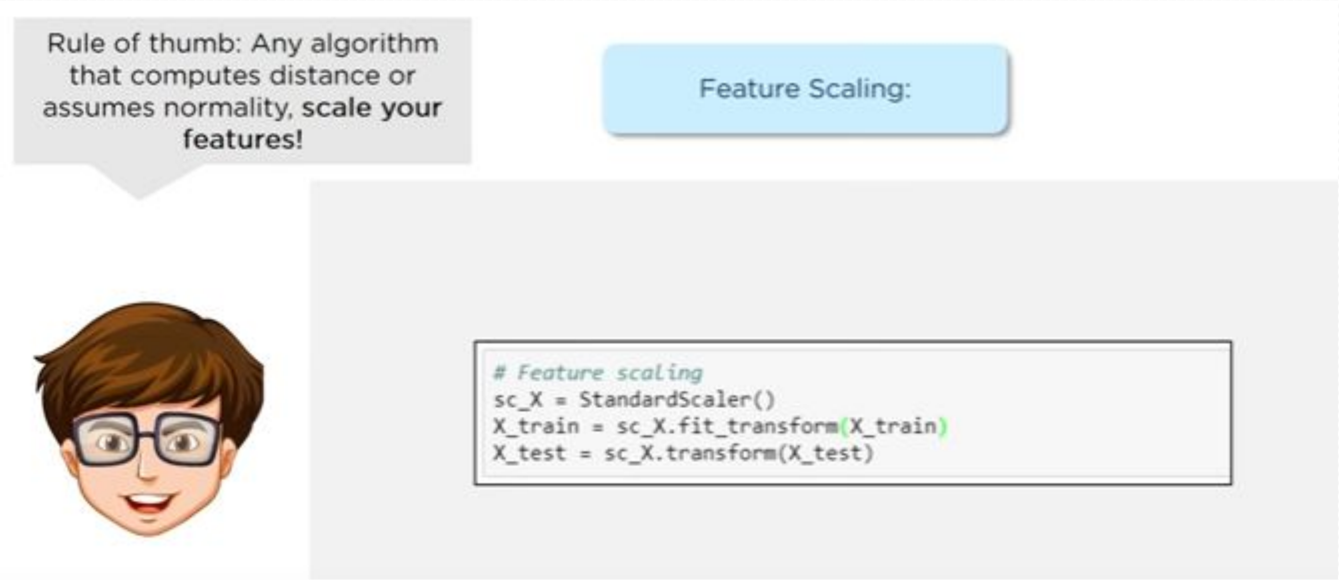
USE Case: predict diabetes



Before proceeding further, let's split the dataset into train and test:

```
# split dataset
X = dataset.iloc[:, 0:8]
y = dataset.iloc[:, 8]
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.2)
```

USE Case: predict diabetes

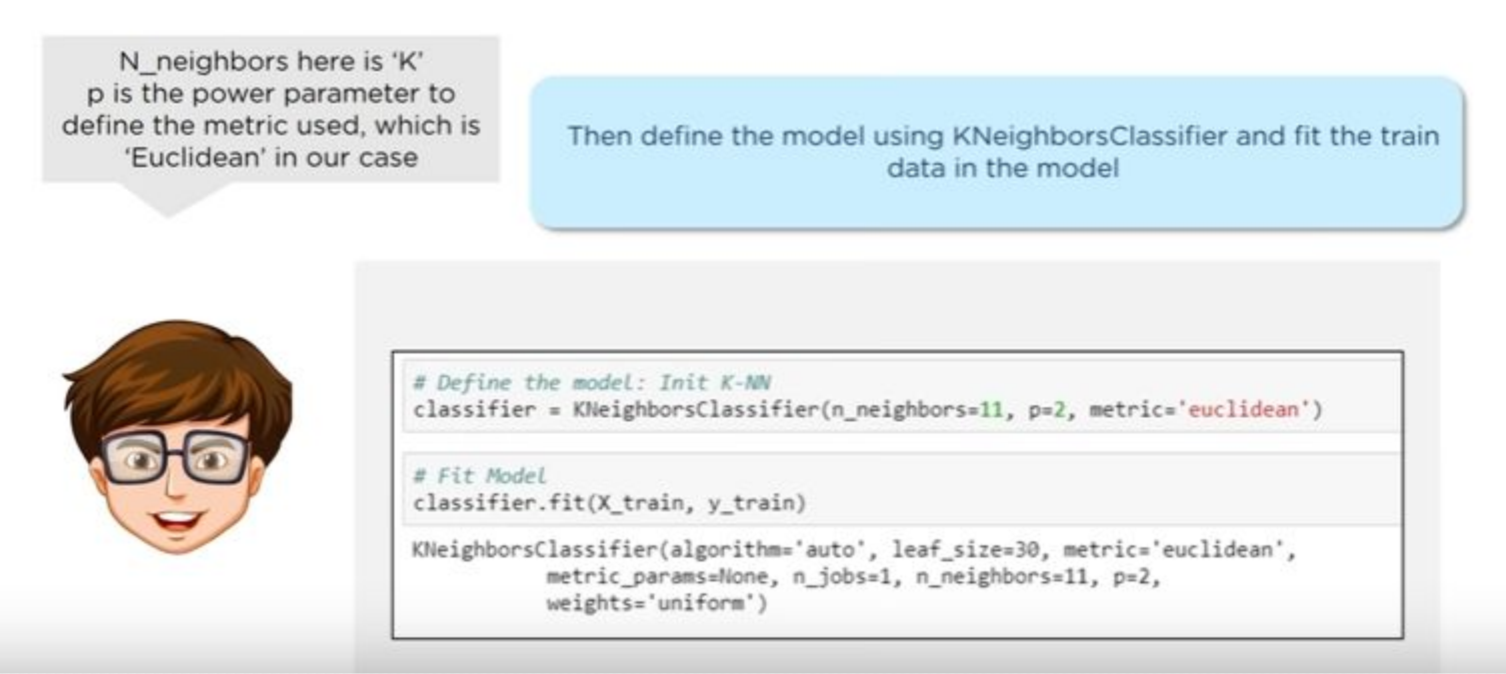


Rule of thumb: Any algorithm that computes distance or assumes normality, **scale your features!**

Feature Scaling:

```
# Feature scaling
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
```

USE Case: predict diabetes



N_neighbors here is ‘K’
p is the power parameter to define the metric used, which is ‘Euclidean’ in our case

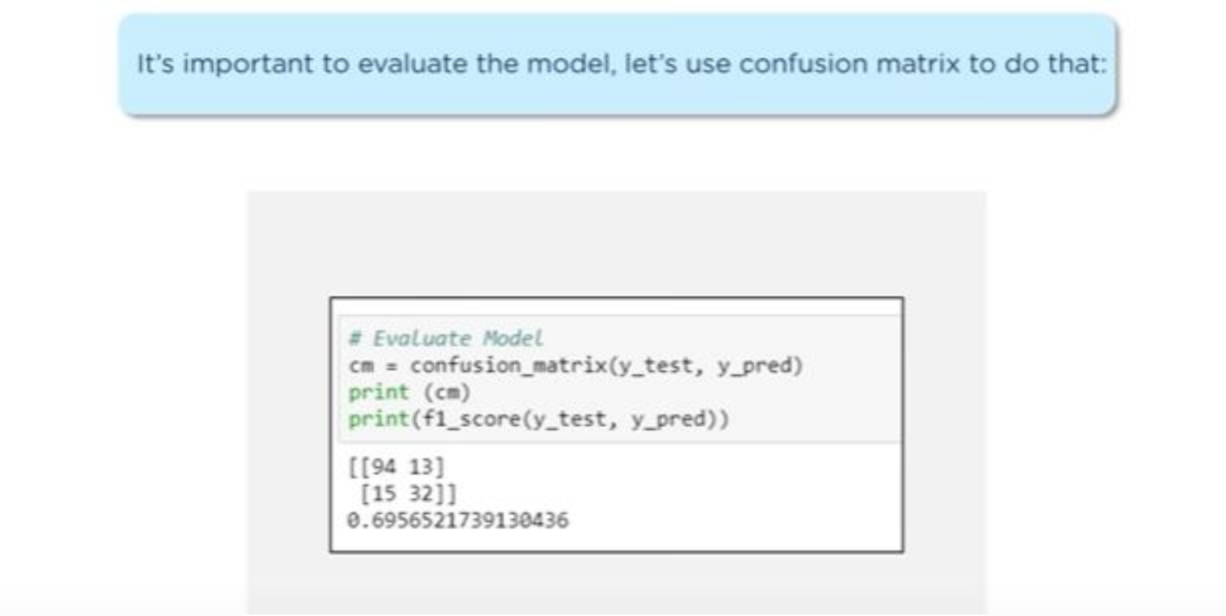
Then define the model using KNeighborsClassifier and fit the train data in the model

```
# Define the model: Init K-NN
classifier = KNeighborsClassifier(n_neighbors=11, p=2, metric='euclidean')

# Fit Model
classifier.fit(X_train, y_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
           metric_params=None, n_jobs=1, n_neighbors=11, p=2,
           weights='uniform')
```

USE Case: predict diabetes

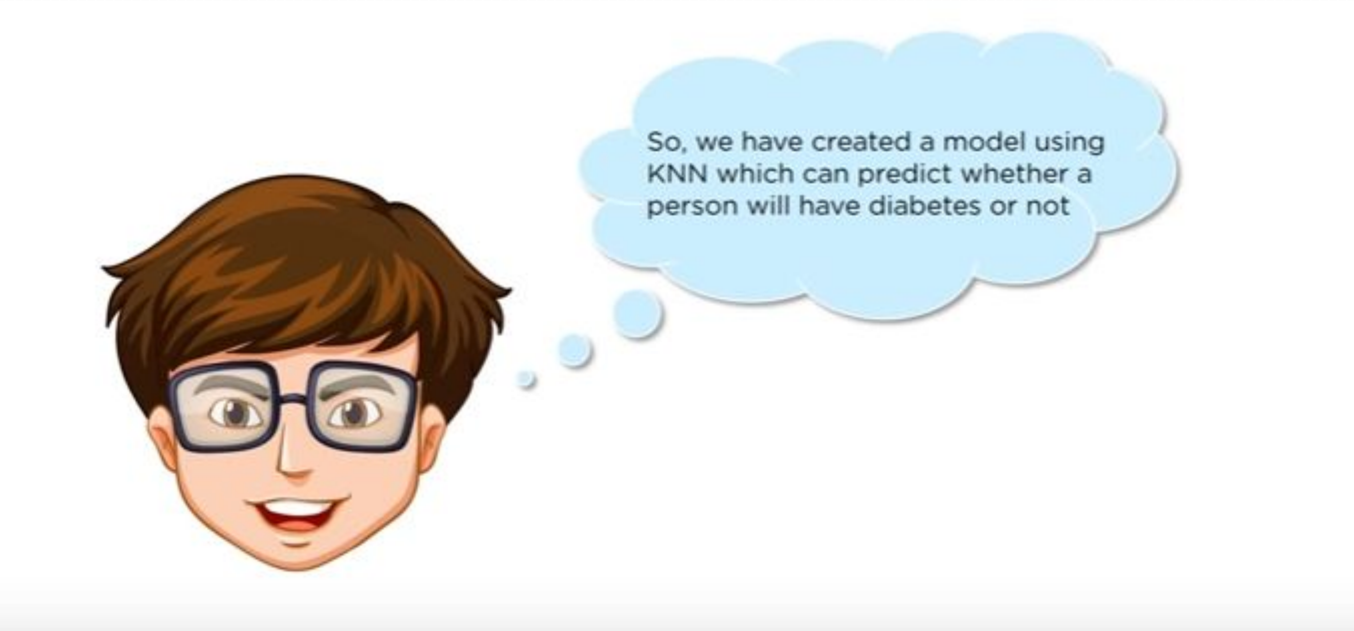


It's important to evaluate the model, let's use confusion matrix to do that:

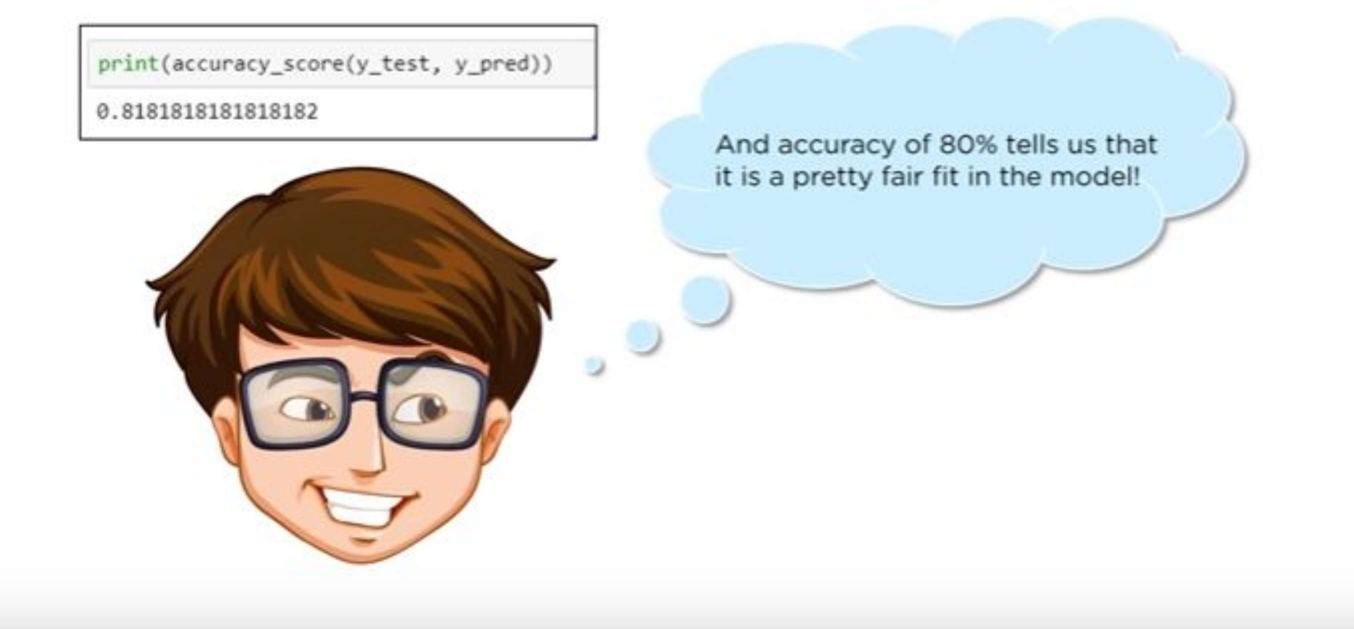
```
# Evaluate Model
cm = confusion_matrix(y_test, y_pred)
print (cm)
print(f1_score(y_test, y_pred))
```

```
[[94 13]
 [15 32]]
0.6956521739130436
```

USE Case: predict diabetes


So, we have created a model using KNN which can predict whether a person will have diabetes or not

USE Case: predict diabetes



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

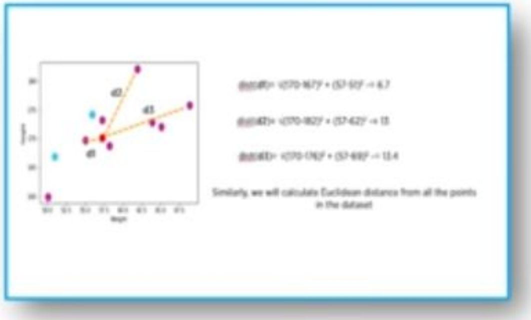
```
0.8181818181818182
```

And accuracy of 80% tells us that it is a pretty fair fit in the model!

summary

WHY WE NEED KNN?

EUCLEDIAN DISTANCE



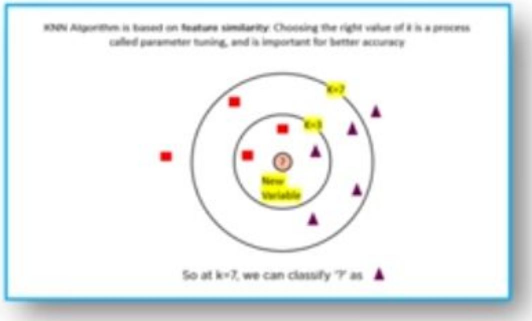
dist(d1)= √(170-167)² + (57-51)² ~= 6.7

dist(d2)= √(170-182)² + (57-62)² ~= 13

dist(d3)= √(170-176)² + (57-69)² ~= 13.4

Similarly, we will calculate Euclidean distance from all the points in the dataset

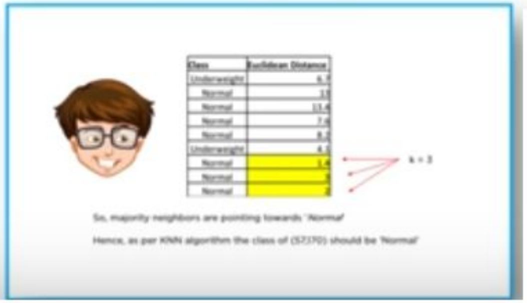
CHOOSING THE VALUE OF K



KNN Algorithm is based on **feature similarity**: Choosing the right value of k is a process called parameter tuning, and is important for better accuracy

So at k=7, we can classify '?' as ▲

HOW KNN WORKS?



Class	Euclidean Distance
Underweight	6.7
Normal	13
Normal	13.4
Normal	7.6
Normal	8.2
Underweight	4.1
Normal	1.4
Normal	2
Normal	3

So, majority neighbors are pointing towards 'Normal'

Hence, as per KNN algorithm the class of (57,170) should be 'Normal'

KNN CLASSIFIER FOR DIABETES PREDICTION

N_neighbors here is 'K'
p is the power parameter to define the metric used, which is 'Euclidean' in our case

Then define the model using KNeighborsClassifier and fit the train data in the model

```
# Define the model: Init K-NN
classifier = KNeighborsClassifier(n_neighbors=11, p=2, metric='euclidean')

# Fit Model
classifier.fit(X_train, y_train)

KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
           metric_params=None, n_jobs=1, n_neighbors=11, p=2,
           weights='uniform')
```

THANKS!