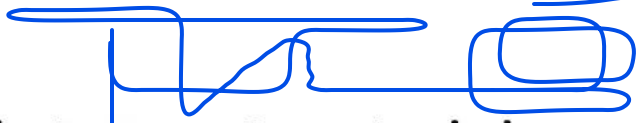
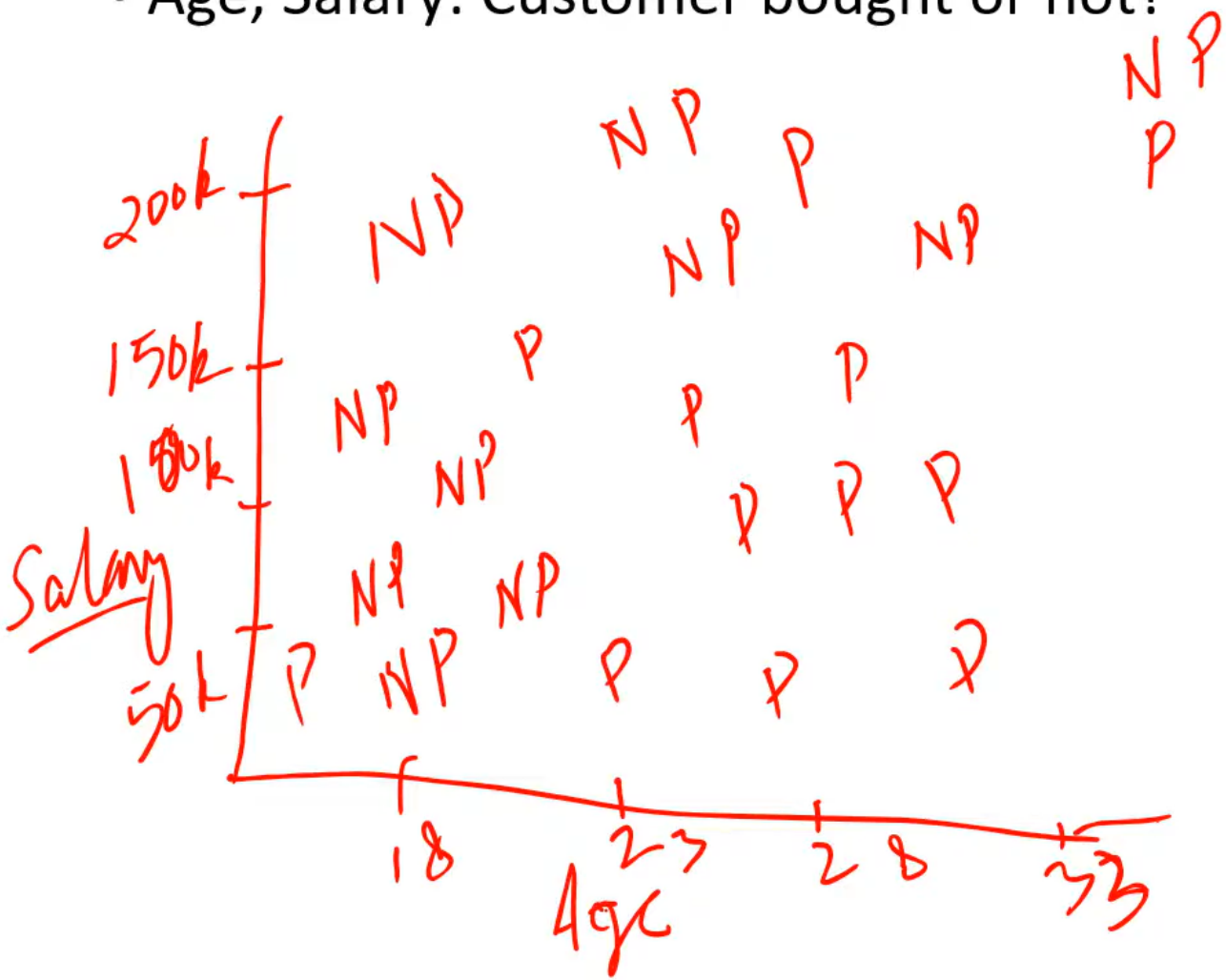
Intuition

- Decision Tree algorithm belongs to the family of supervised learning algorithms.
- The decision tree algorithm can be used for solving **regression and classification problems** too.
- The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by **learning simple decision rules** inferred from prior data(training data).
- The algorithm creates a flowchart of decision branches.



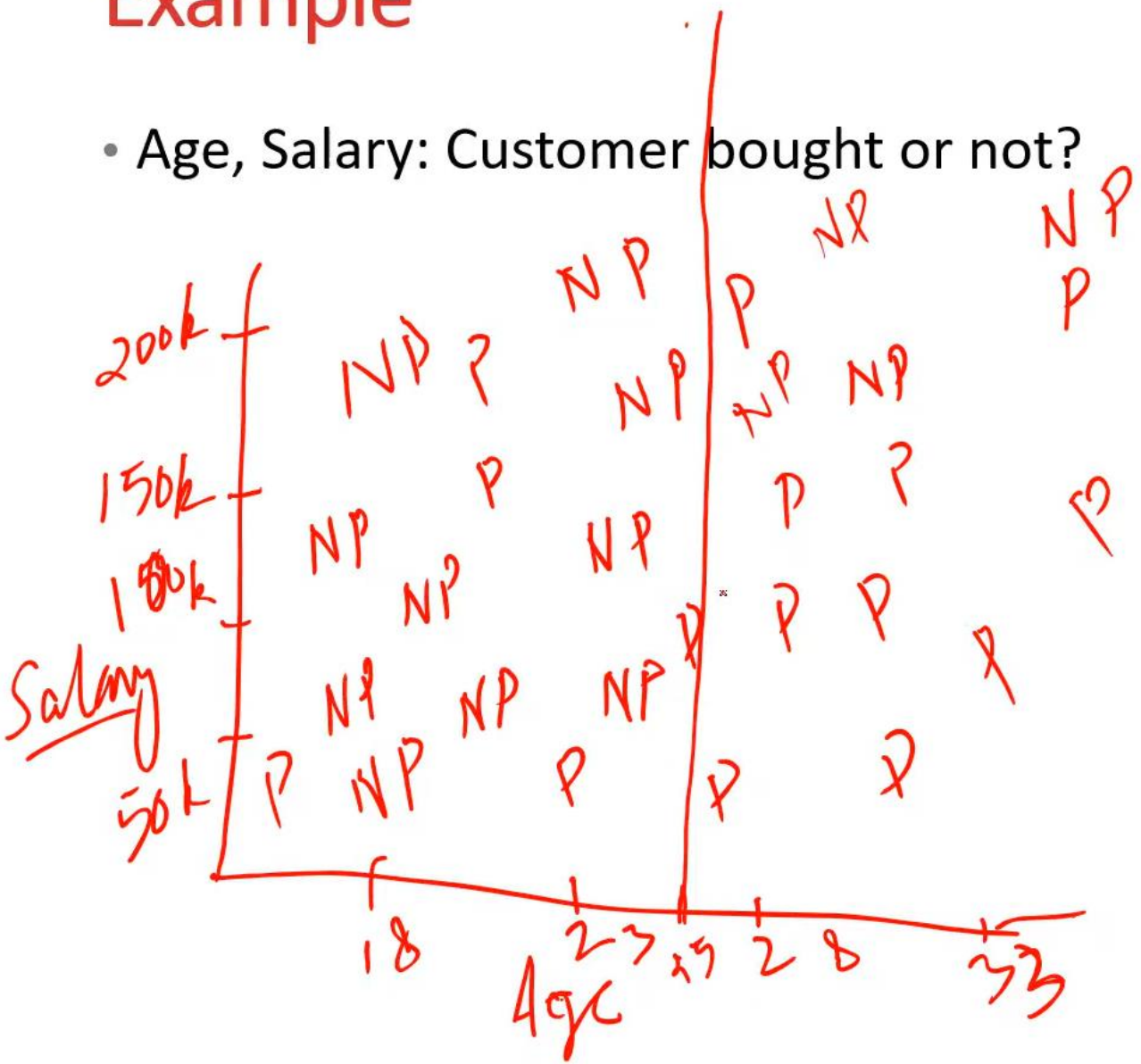
Example

- Age, Salary: Customer bought or not?



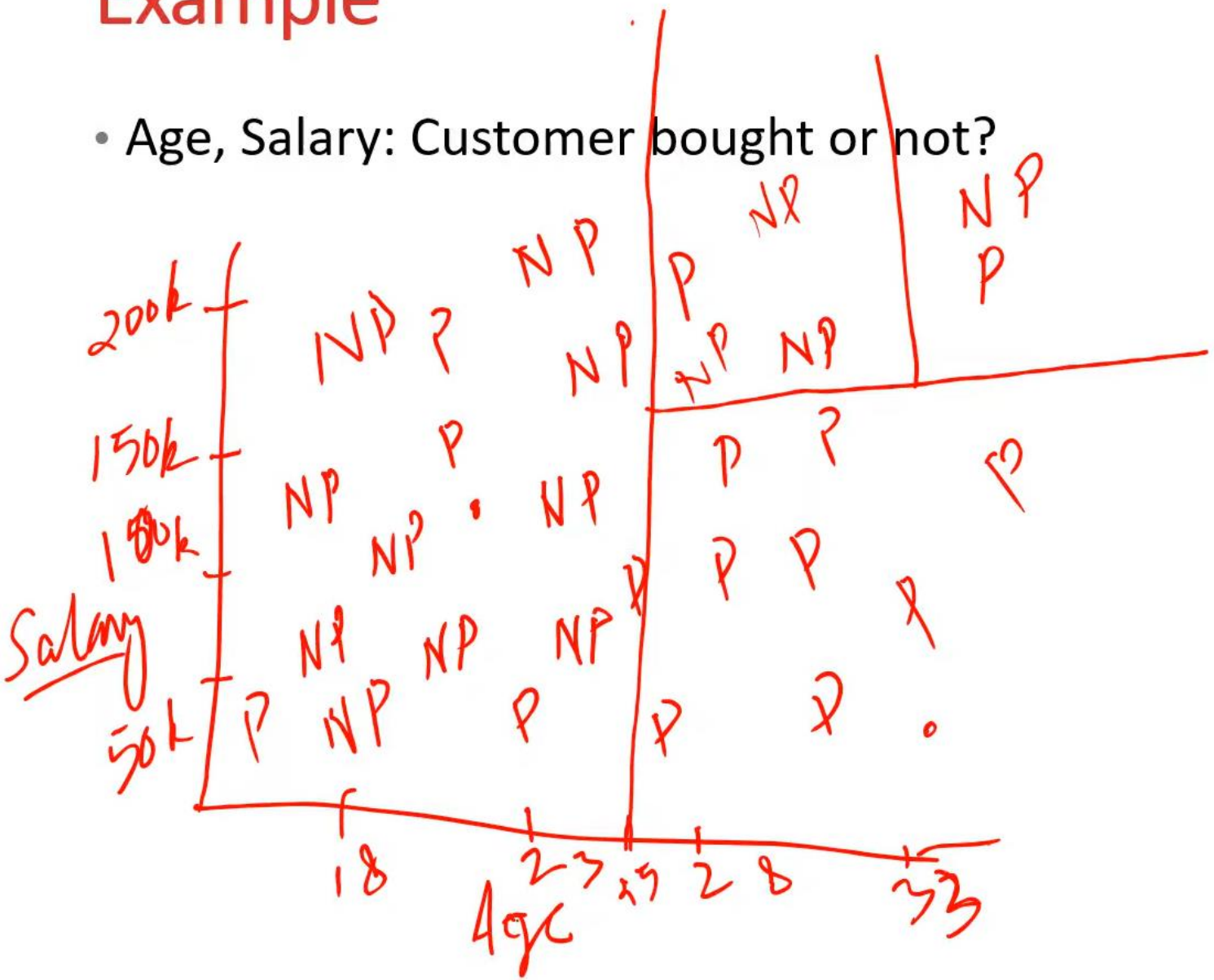
Example

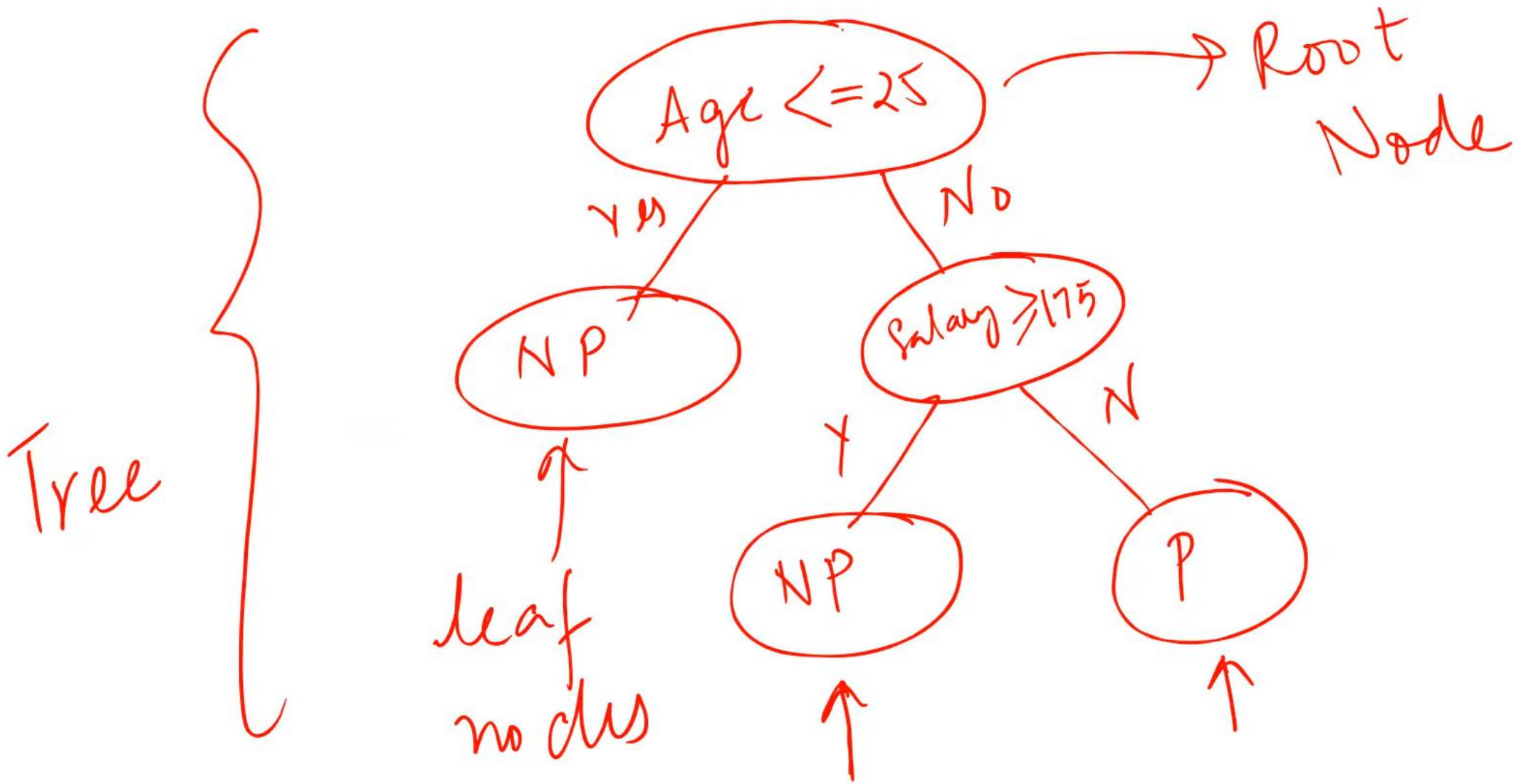
- Age, Salary: Customer bought or not?



Example

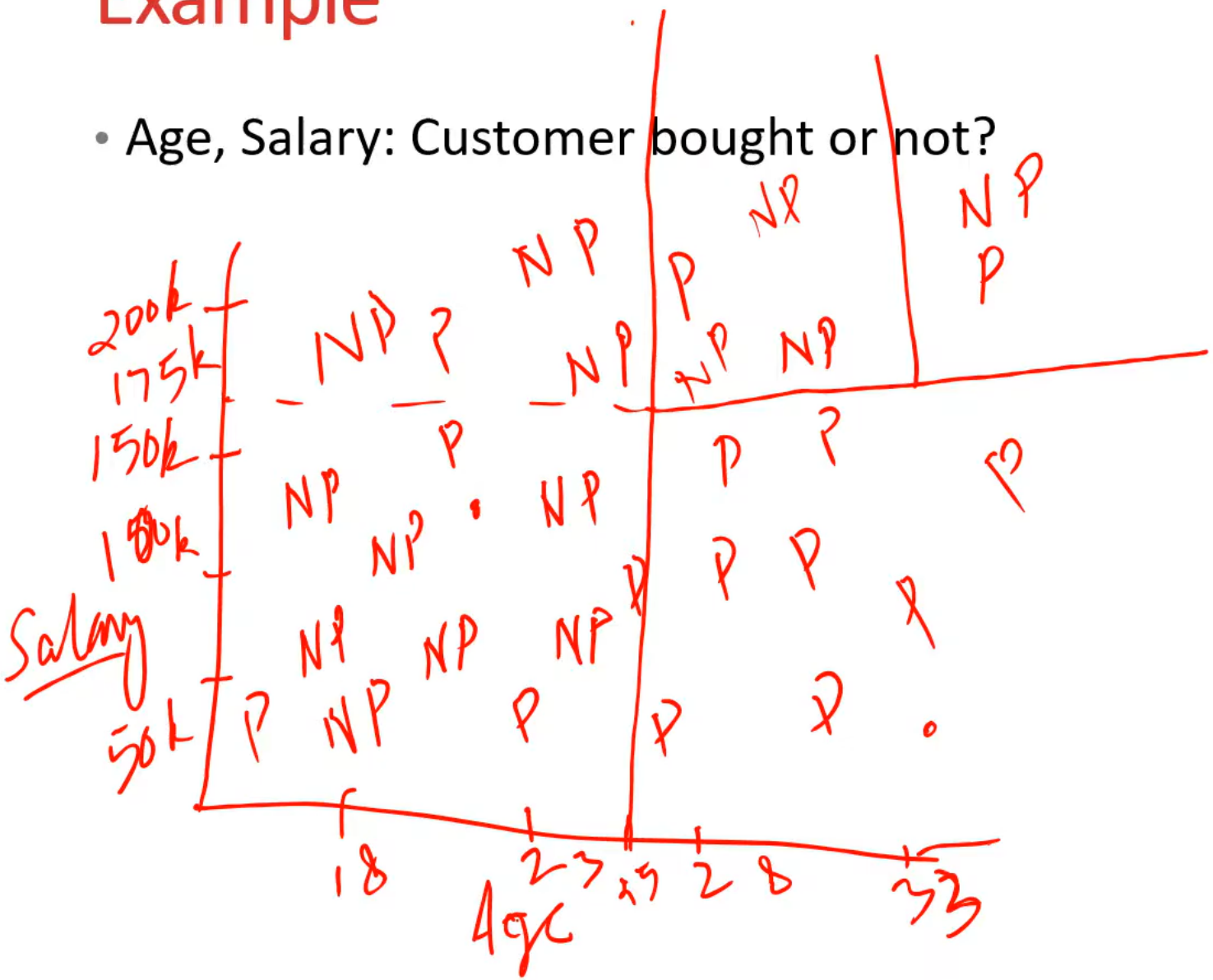
- Age, Salary: Customer bought or not?

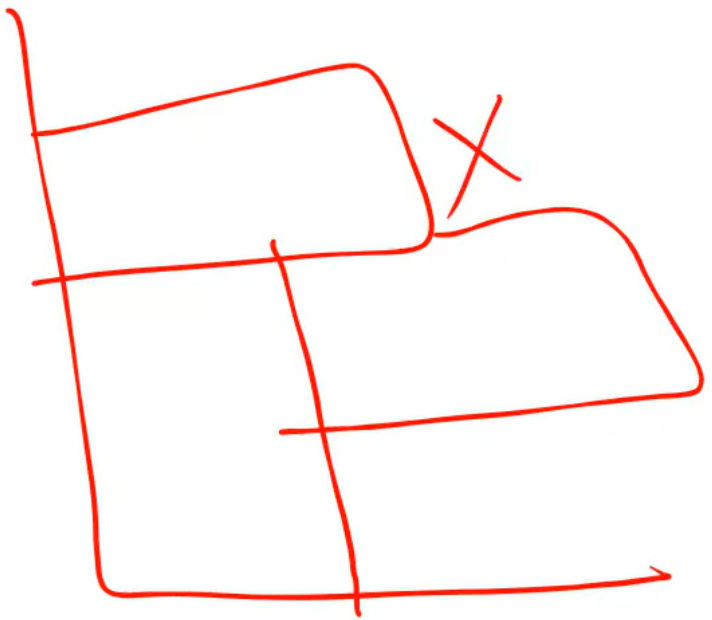



Tree
Age <=25
Root Node
Yes
No
NP
Salary >=175
Y
N
NP
P
leaf nodes

Example

- Age, Salary: Customer bought or not?





Details of the Tree Building Process

- Divide predictor space into N non-overlapping regions
- Every observation that falls in the region Ri, we make a prediction which is the mean of the response values for the training observations.
- In theory the regions could have any shape, but we choose to divide the predictor space into rectangles/boxes (for ease of interpretation)
- Goal is:

$$\min \sum_{i=1}^{N} \sum_{j \in R_i} \left(y_j - \hat{y}_{R_i} \right)^2$$

$\hat{y}_{R_i}$: mean response

Basic definitions

- Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.
- Information gain is a statistical property that measures how well a given feature separates the training examples according to their target classification.
- Constructing a decision tree is all about finding a feature that returns the highest information gain and the smallest entropy.

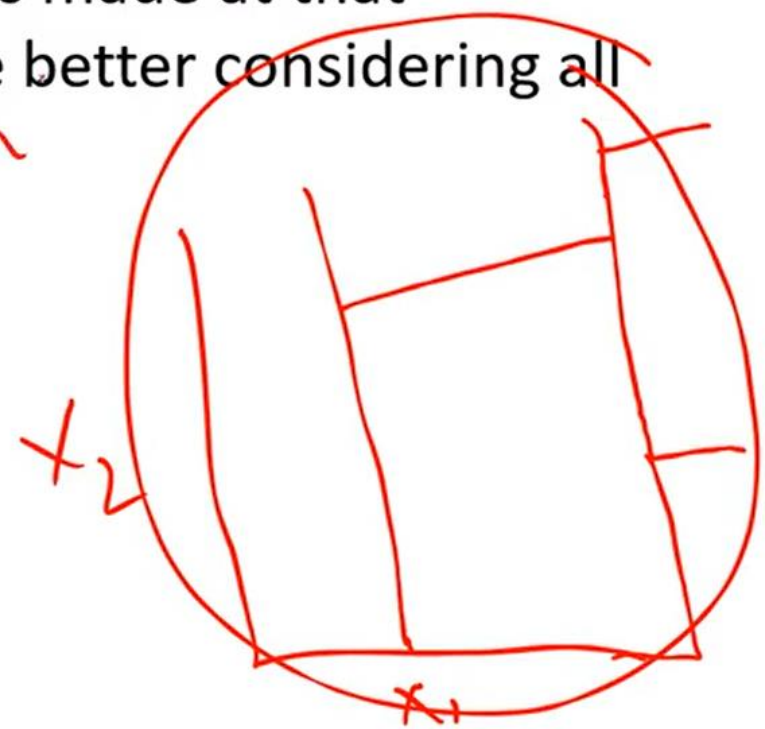
Intricacies

- On each iteration of the algorithm, it iterates through the most unused feature of the data and calculates **Entropy(H)** and **Information gain(IG)** of the feature.
- It then selects the feature which has the smallest Entropy or Largest Information gain.
- If the dataset consists of **N** features, then deciding which features to place at the root or at different levels of the tree is a complicated step. By just randomly selecting any node to be the root may give us bad results with low accuracy.
- For solving this feature selection problem, researchers worked and devised some solutions. They suggested using some *criteria* like :
- **Entropy**,
  Information gain,
  Gini index,
  Gain Ratio,
  Reduction in Variance
  Chi-Square

More details of the Algorithms

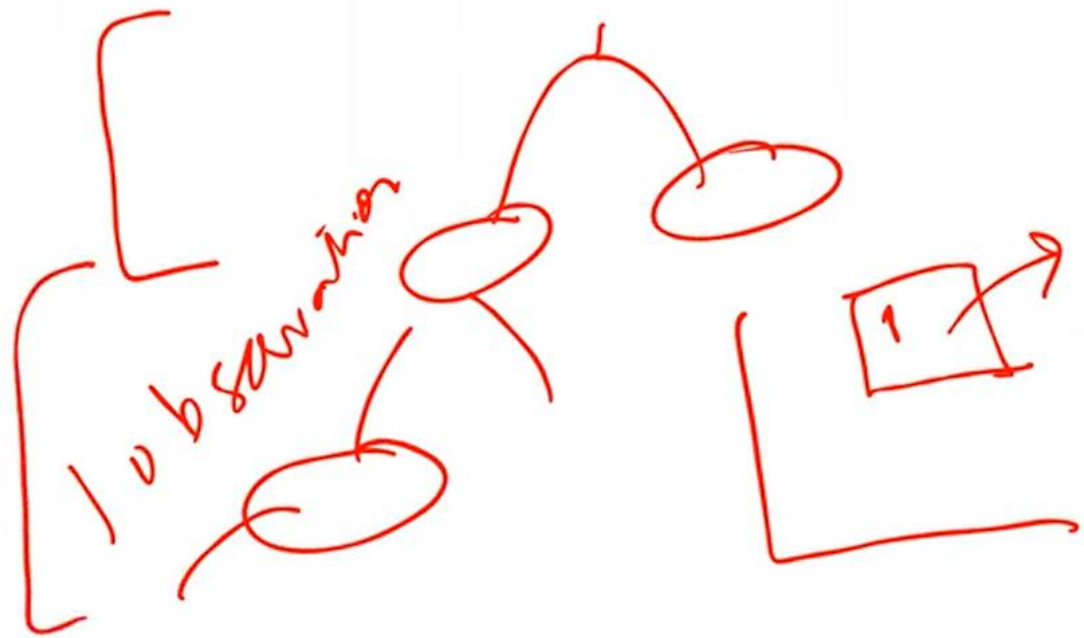
- It is practically impossible to consider every possible partition.
- This algorithm follows a top-down greedy approach called recursive binary splitting.
- At each step of the tree building process, the best split is made at that particular step, rather than looking at a step that will be better considering all future steps.

Greedy Algorithm



Improving Performance

- Decision-trees can create over-complex trees that do not generalize the data well. This is called overfitting.
- Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.



Python Implementation

- Using the DecisionTreeClassifier Class:
  - criterion{“gini”, “entropy”}, default=”gini”
  - The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy” for the information gain.

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\uzmam\Machine Learning Course\Classification

Editor - C:\Users\uzmam\Machine Learning Course\Classification\decision_tree_classification.py

decision_tree_classification.py

```
# Decision Tree Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =
print(X_train)
print(y_train)
print(X_test)
print(y_test)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)
```

Help

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front

Help Variable explorer File explorer

IPython console

Console 36/A

```
Python 3.7.3 (default, Apr 24 2019, 15:29:51)
[MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for
more information.

IPython 7.6.1 -- An enhanced Interactive
Python.

In [1]:
```

IPython console History log

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 7 Column: 1 Memory: 39 %

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\uzmam\Machine Learning Course\Classification

Editor - C:\Users\uzmam\Machine Learning Course\Classification\decision_tree_classification.py

decision_tree_classification.py

```
# Decision Tree Classification

# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('Social_Network_Ads.csv')
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =
print(X_train)
print(y_train)
print(X_test)
print(y_test)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
print(X_train)
print(X_test)

# Training the Decision Tree Classification model on the Training set
```

Help

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front

Help Variable explorer File explorer

IPython console

Console 36/A

```
Python 3.7.3 (default, Apr 24 2019, 15:29:51)
[MSC v.1915 64 bit (AMD64)]
Type "copyright", "credits" or "license" for
more information.

IPython 7.6.1 -- An enhanced Interactive
Python.

In [1]:
```

IPython console History log

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 27 Column: 14 Memory: 39 %

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\uzmam\Machine Learning Course\Classification

Editor - C:\Users\uzmam\Machine Learning Course\Classification\decision_tree_classification.py

decision_tree_classification.py

```
20
21 # Feature Scaling
22 from sklearn.preprocessing import StandardScaler
23 sc = StandardScaler()
24 X_train = sc.fit_transform(X_train)
25 X_test = sc.transform(X_test)
26 print(X_train)
27 print(X_test)
28
29 # Training the Decision Tree Classification model on the Training set
30 from sklearn.tree import DecisionTreeClassifier
31 classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
32 classifier.fit(X_train, y_train)
33
34 # Predicting a new result
35 print(classifier.predict(sc.transform([[35,100000]])))
36
37 # Predicting the Test set results
38 y_pred = classifier.predict(X_test)
39 print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
40
41 # Making the Confusion Matrix
42 from sklearn.metrics import confusion_matrix, accuracy_score
43 cm = confusion_matrix(y_test, y_pred)
44 print(cm)
45 print(accuracy_score(y_test, y_pred))
46
47
```

Help

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front

Help Variable explorer File explorer

IPython console

Console 36/A

```
[-1.39899564 -0.42281668]
[-1.20093113 -1.00267957]
[ 1.96810099 -0.91570013]
[ 0.38358493  0.30201192]
[ 0.18552042  0.1570462 ]
[ 2.06713324  1.75166912]
[ 0.77971394 -0.8287207 ]
[ 0.28455268 -0.27785096]
[ 0.38358493 -0.16187839]
[-0.11157634  2.21555943]
[-1.49802789 -0.62576869]
[-1.29996338 -1.06066585]
[-1.39899564  0.41798449]
[-1.10189888  0.76590222]
[-1.49802789 -0.19087153]
[ 0.97777845 -1.06066585]
[ 0.97777845  0.59194336]
[ 0.38358493  0.99784738]]

In [2]:
```

IPython console History log

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 32 Column: 33 Memory: 40 %

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\uzmam\Machine Learning Course\Classification

Editor - C:\Users\uzmam\Machine Learning Course\Classification\decision_tree_classification.py

decision_tree_classification.py

```
20
21 # Feature Scaling
22 from sklearn.preprocessing import StandardScaler
23 sc = StandardScaler()
24 X_train = sc.fit_transform(X_train)
25 X_test = sc.transform(X_test)
26 print(X_train)
27 print(X_test)
28
29 # Training the Decision Tree Classification model on the Training set
30 from sklearn.tree import DecisionTreeClassifier
31 classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
32 classifier.fit(X_train, y_train)
33
34 # Predicting a new result
35 print(classifier.predict(sc.transform([[35,100000]])))
36
37 # Predicting the Test set results
38 y_pred = classifier.predict(X_test)
39 print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
40
41 # Making the Confusion Matrix
42 from sklearn.metrics import confusion_matrix, accuracy_score
43 cm = confusion_matrix(y_test, y_pred)
44 print(cm)
45 print(accuracy_score(y_test, y_pred))
46
47
```

Help

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front

Help Variable explorer File explorer

IPython console

Console 36/A

```
Out[2]:
DecisionTreeClassifier(class_weight=None,
criterion='entropy', max_depth=None,
                       max_features=None,
max_leaf_nodes=None,
                       min_impurity_decrease=0.
0, min_impurity_split=None,
                       min_samples_leaf=1,
min_samples_split=2,
                       min_weight_fraction_leaf
=0.0, presort=False,
                       random_state=0,
splitter='best')

In [3]:
print(classifier.predict(sc.transform([[35,1000
00]])))
[1]

In [4]:
```

IPython console History log

Permissions: RW End-of-lines: LF Encoding: ASCII Line: 36 Column: 1 Memory: 40 %

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\uzmam\Machine Learning Course\Classification

Editor - C:\Users\uzmam\Machine Learning Course\Classification\decision_tree_classification.py

decision_tree_classification.py

```
20
21 # Feature Scaling
22 from sklearn.preprocessing import StandardScaler
23 sc = StandardScaler()
24 X_train = sc.fit_transform(X_train)
25 X_test = sc.transform(X_test)
26 print(X_train)
27 print(X_test)
28
29 # Training the Decision Tree Classification model on the Training set
30 from sklearn.tree import DecisionTreeClassifier
31 classifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
32 classifier.fit(X_train, y_train)
33
34 # Predicting a new result
35 print(classifier.predict(sc.transform([[35,100000]])))
36
37 # Predicting the Test set results
38 y_pred = classifier.predict(X_test)
39 print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
40
41 # Making the Confusion Matrix
42 from sklearn.metrics import confusion_matrix, accuracy_score
43 cm = confusion_matrix(y_test, y_pred)
44 print(cm)
45 print(accuracy_score(y_test, y_pred))
46
47
```

Help

Source Console Object

Usage

Here you can get help of any object by pressing **Ctrl+I** in front

Help Variable explorer File explorer

IPython console

Console 36/A

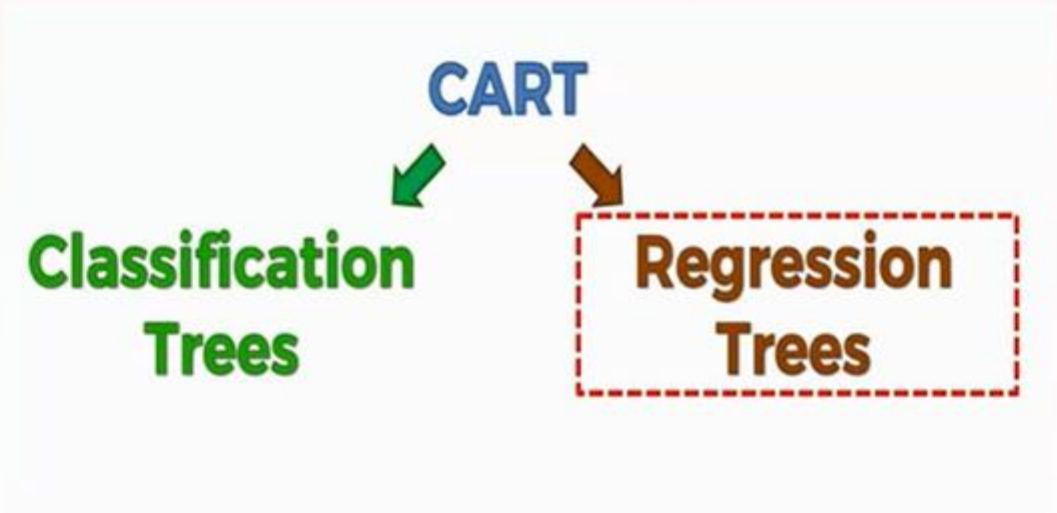
```
 [0 1]
 [0 0]
 [1 1]
 [1 1]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [0 0]
 [0 0]
 [1 1]
 [0 0]
 [1 1]
 [1 1]
 [1 1]]
[[62  6]
 [ 3 29]]
0.91

In [5]:
```

IPython console History log

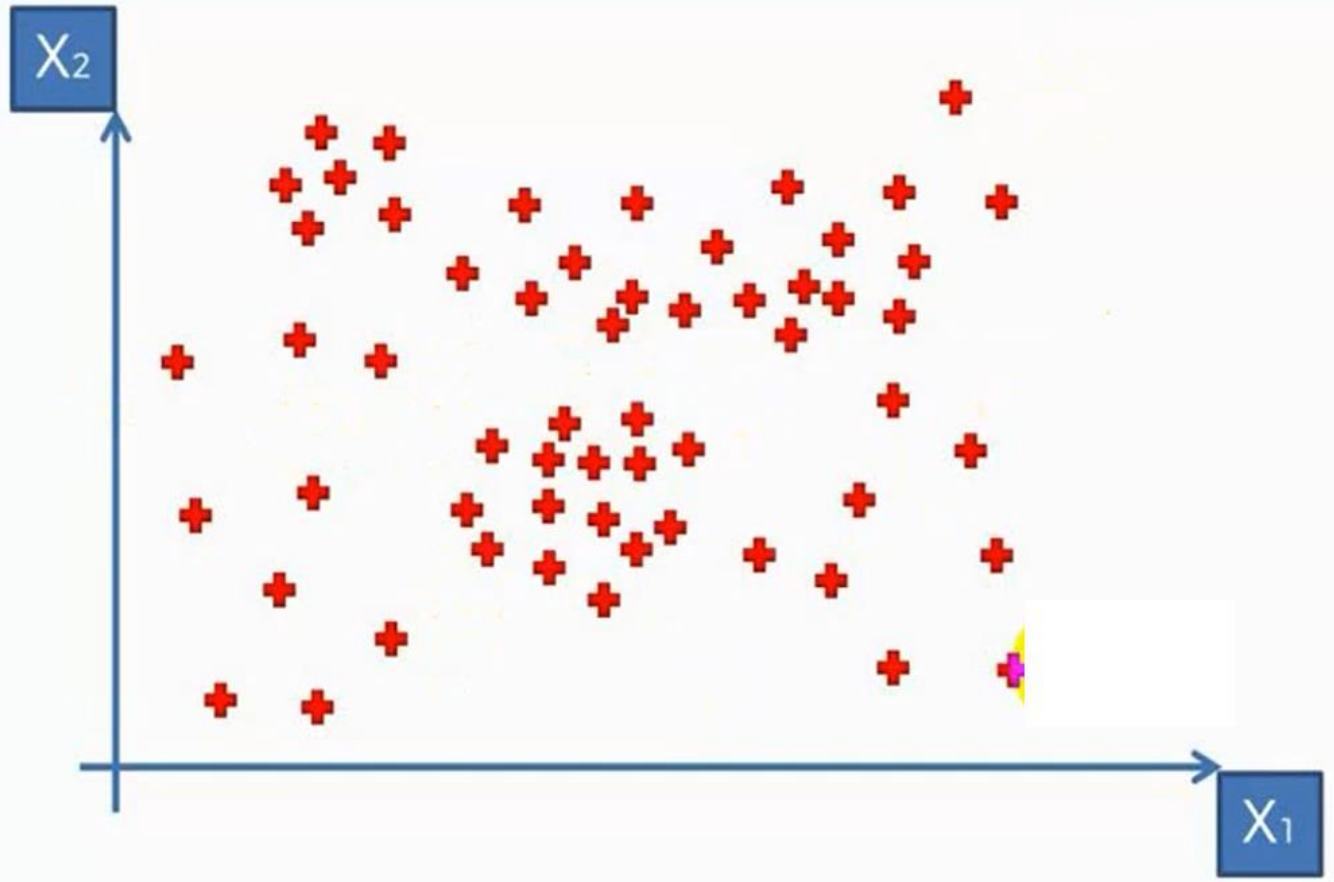
Permissions: RW End-of-lines: LF Encoding: ASCII Line: 46 Column: 1 Memory: 40 %

Decision Tree Classifier

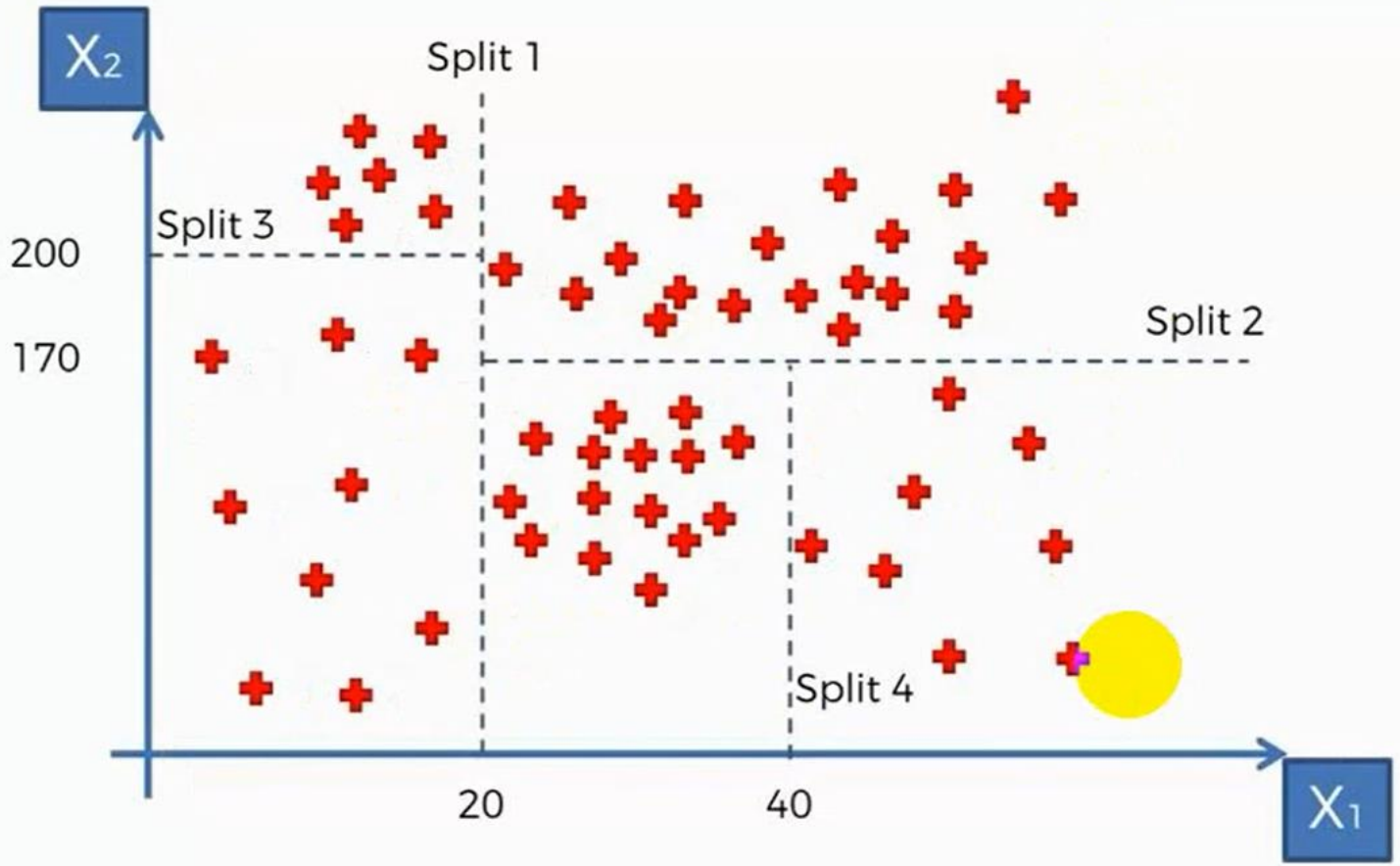
A Classification And Regression Tree (**CART**), is a predictive model, which explains how an outcome variable's values can be predicted based on other values. A **CART** output is a decision tree where each fork is a split in a predictor variable and each end node contains a prediction for the outcome variable



Support vector regression



Decision Tree Regression



Decision Tree Classifier

Decision Tree Learning Algorithm

- There are many specific decision-tree algorithms. Notable ones include:
- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- CHAID (Chi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.
- MARS: extends decision trees to handle numerical data better.

Concept to map Decision Tree

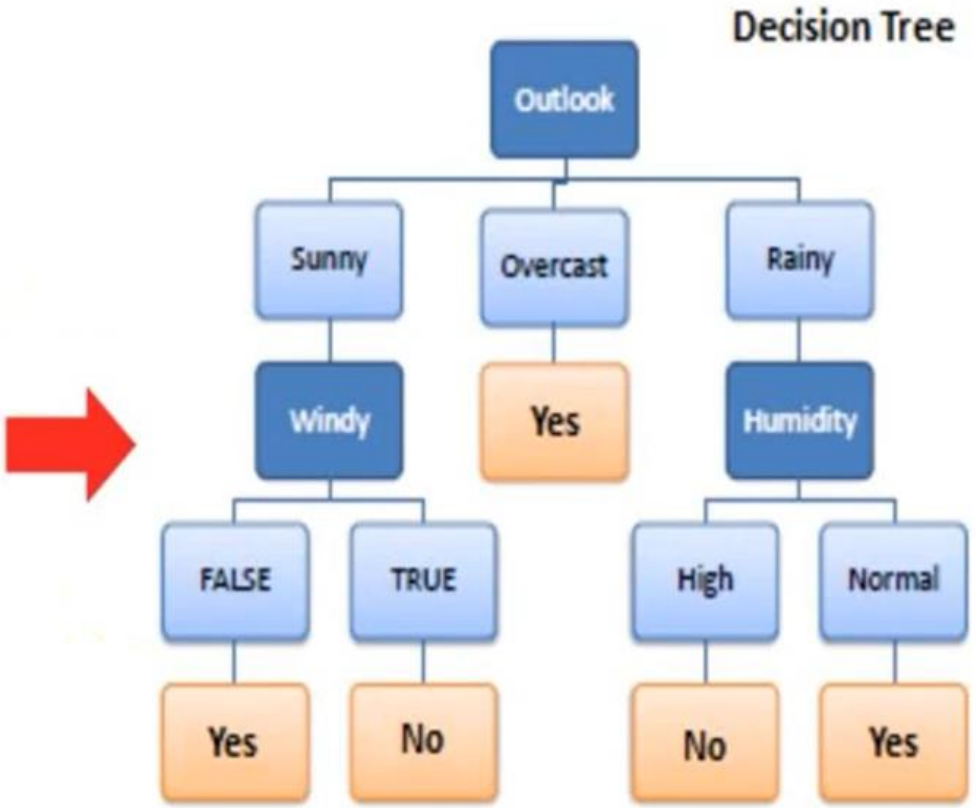
Information Gain: The **information gain** is based on the decrease in entropy after a data-set is split on an attribute. Constructing a **decision tree** is all about finding attribute that returns the highest **information gain** (i.e., the most homogeneous branches) ***OR [a measure of the decrease in disorder achieved by partitioning the original dataset]***

Entropy : Entropy, as it relates to machine learning, is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information. Flipping a coin is an example of an action that provides information that is random. ... This is the essence of entropy.

. ***Or [is a measure of disorder in a dataset]***

ID3 Decision Tree Classifier

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



Concept to map Decision Tree

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)
= 0.94

=-p / p + n * log (p / p+ n) - n / p + n * log(n / p + n)

=-9 / 5 + 5 * log (9 / 9+ 5) - 5 / 9 + 5 * log(5 / 9 + 5)

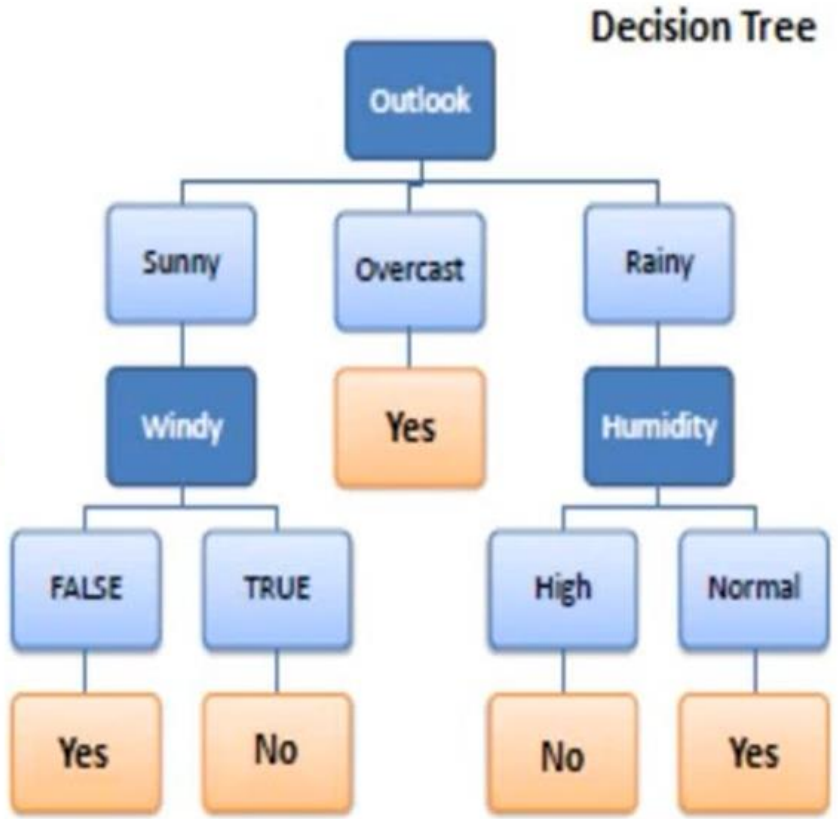
Here p = 9, n = 5 also for log2 = log(?) / log(2)

ID3 Decision Tree Classifier

Predictors

Target

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



Concept to map Decision Tree

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

$$-\frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log 3/5 = 0.970$$

$$-\frac{4}{5}\log\frac{4}{5} - \frac{3}{5}\log 0 = 0$$

$$-\frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log 2/5 = 0.970$$

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Concept to map Decision Tree

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)*__E__(3,2) + **P**(Overcast)*__E__(4,0) + **P**(Rainy)*__E__(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Entropy E(A) = pi + ni / p + n * (I (p,n))

Concept to map Decision Tree

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Information Gain

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Concept to map Decision Tree

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

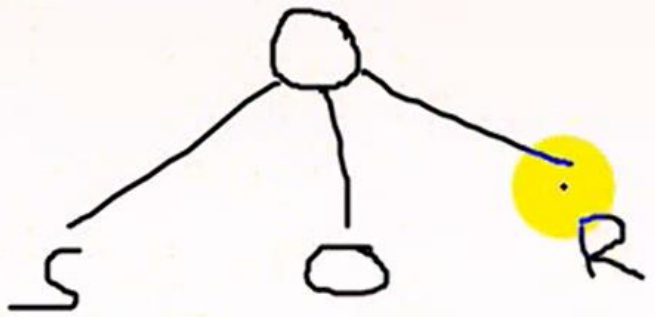
		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Information Gain

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

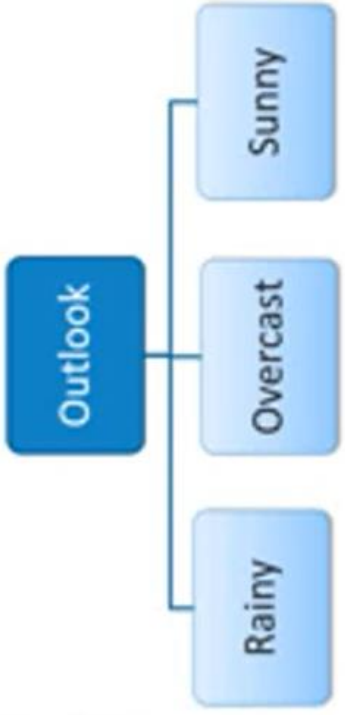
G(PlayGolf, Outlook) = E(PlayGolf) – E(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247



Concept to map Decision Tree

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



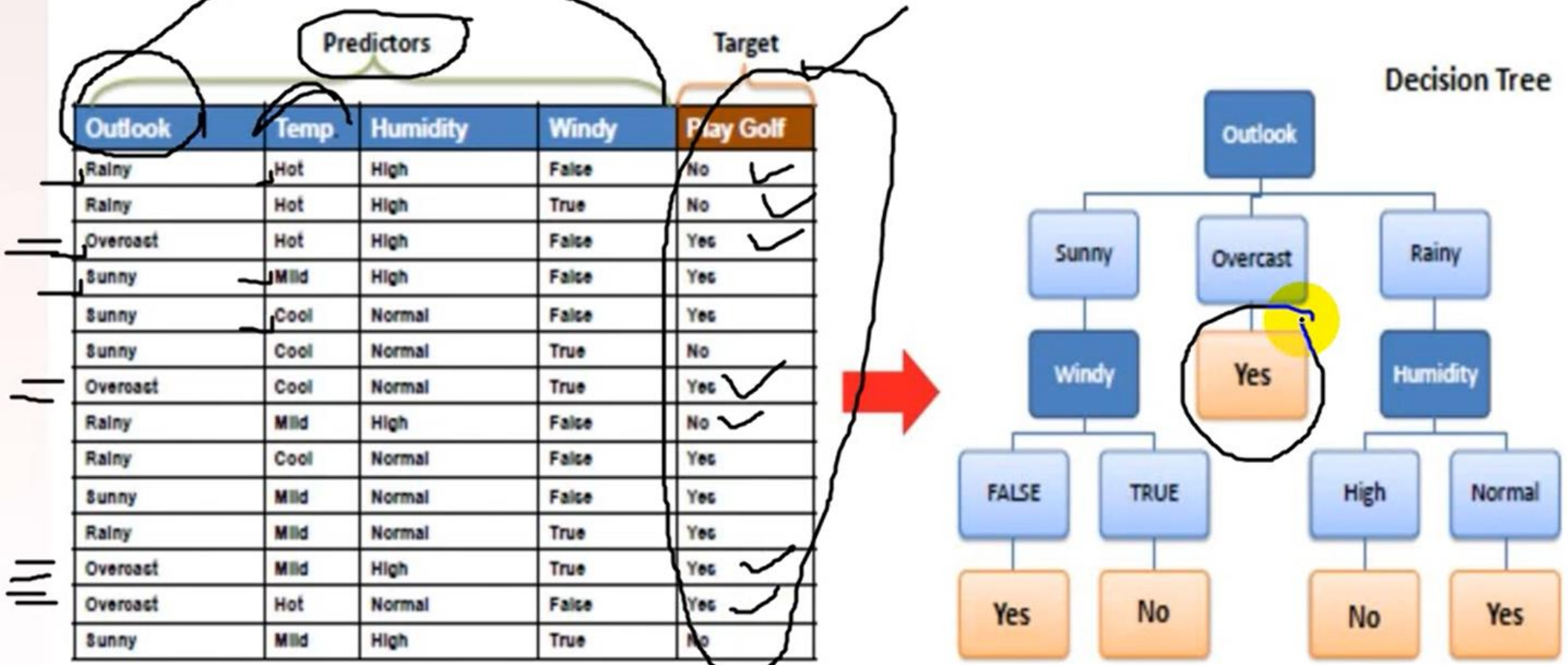
Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

ID3 Decision Tree Classifier

Predictors

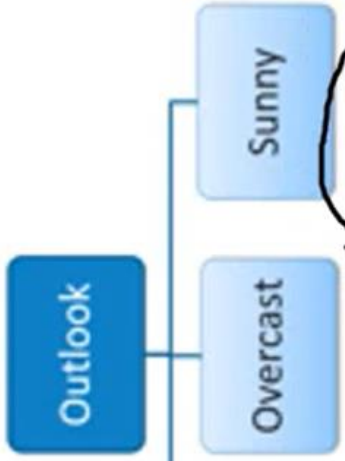
Target

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



Concept to map Decision Tree

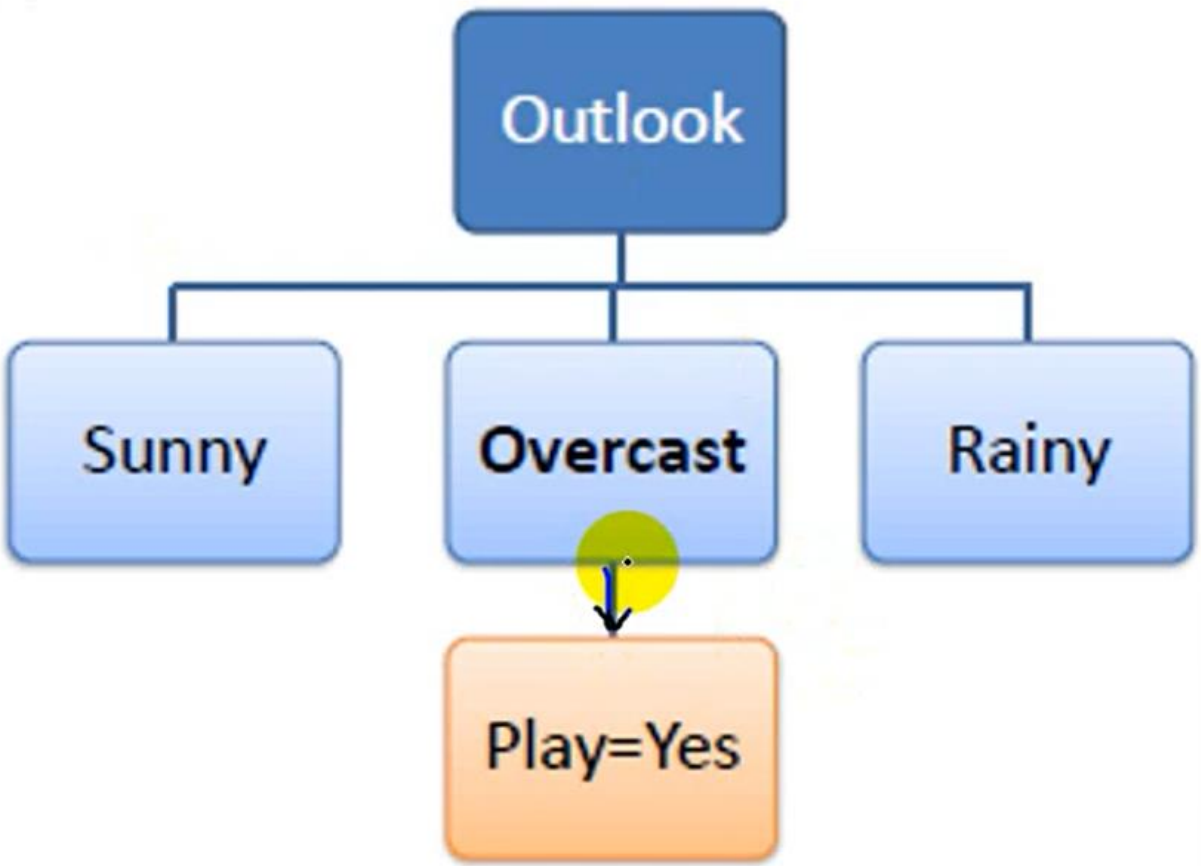
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Concept to map Decision Tree

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes

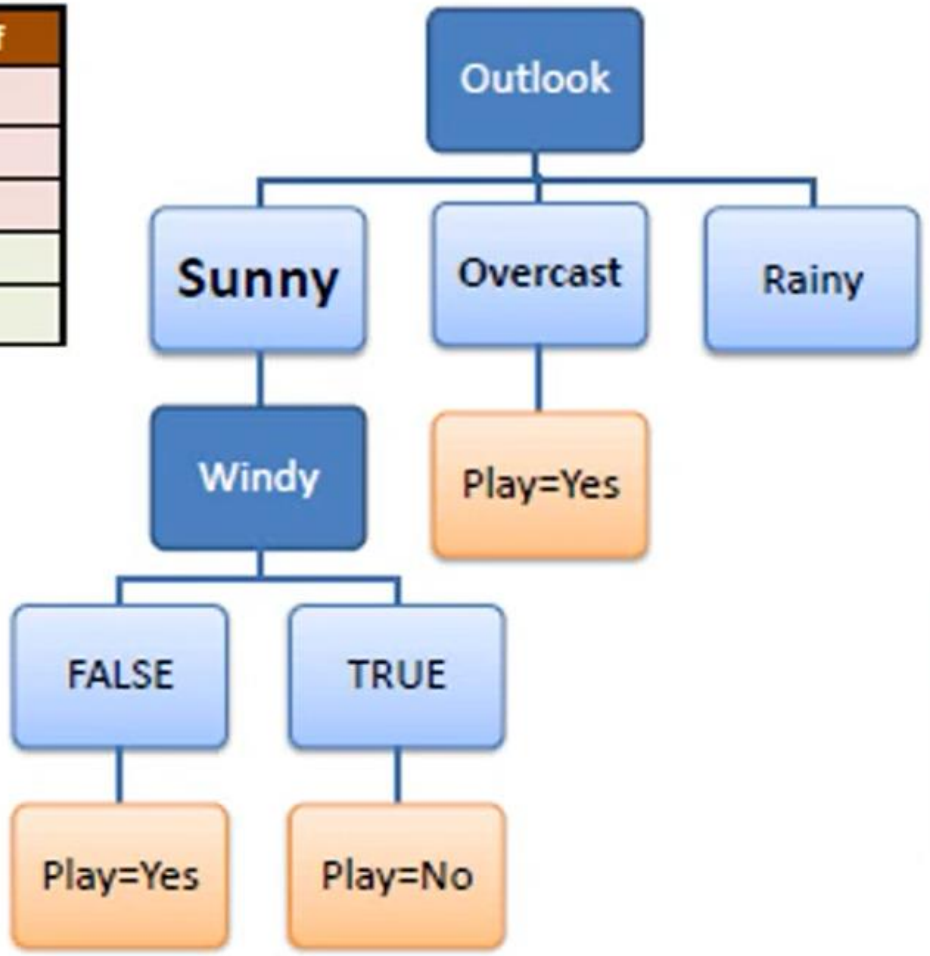


Concept to map Decision Tree

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No

	WIND		
wind	pi	ni	E
TRUE	0	2	0
FALSE	3	0	0
E	0		
G	0.97		

	Temp		
Temp	pi	ni	E
hot	0	0	0
mild	2	1	0.9
cold	1	1	1
E	0.951		
G	0.019		



Concept to map Decision Tree

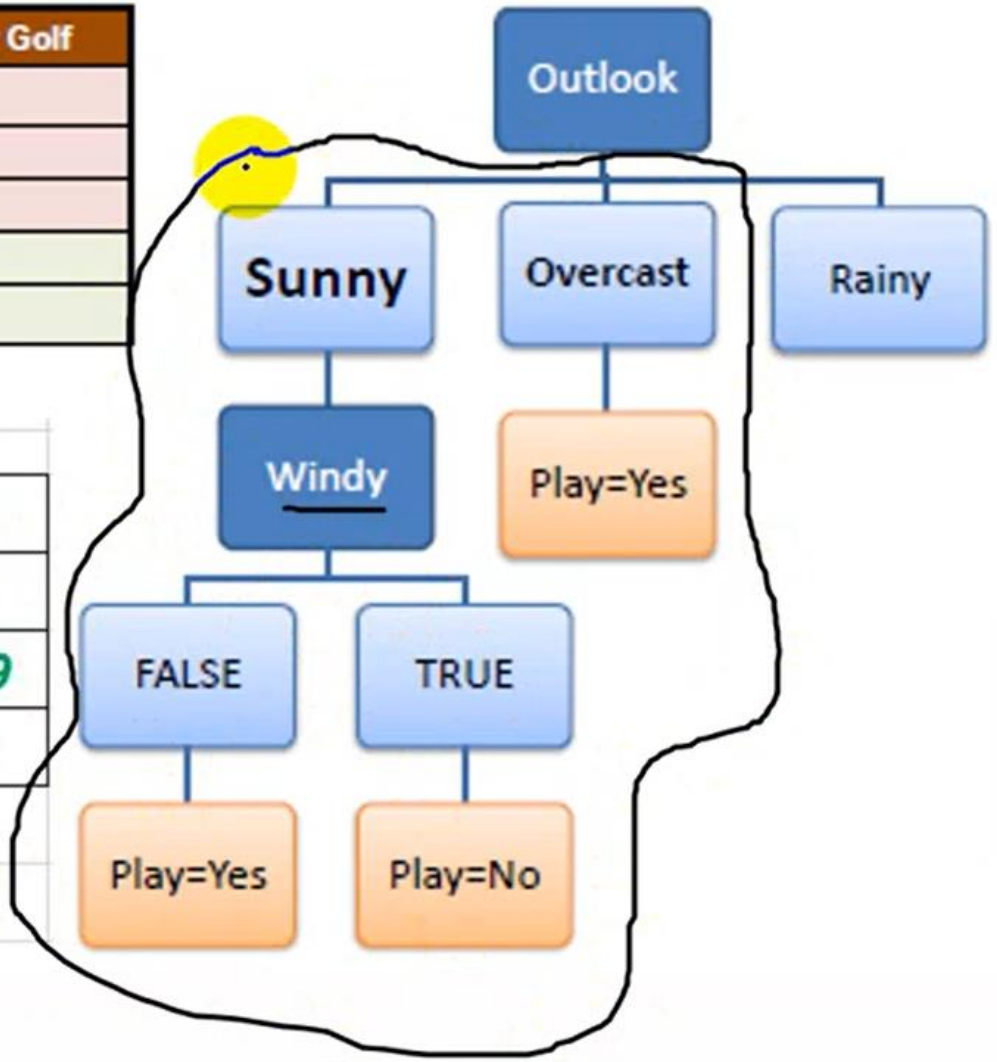
Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No

WIND

wind	pi	ni	E
TRUE	0	2	0
FALSE	3	0	0
E	0		
G	0.97		

Temp

Temp	pi	ni	E
hot	0	0	0
mild	2	1	0.9
cold	1	1	1
E	0.951		
G	0.019		



Concept to map Decision Tree

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes