Machine Learning

Lesson 5: Supervised Learning–Classification

Concepts Covered

- Classification: A supervised learning algorithm
- Decision Tree
- Random Forest
- Naïve Bayes
- Confusion Matrix vs Cost Matrix
- Kernel SVM

Learning Objectives

By the end of this lesson, you will be able to:

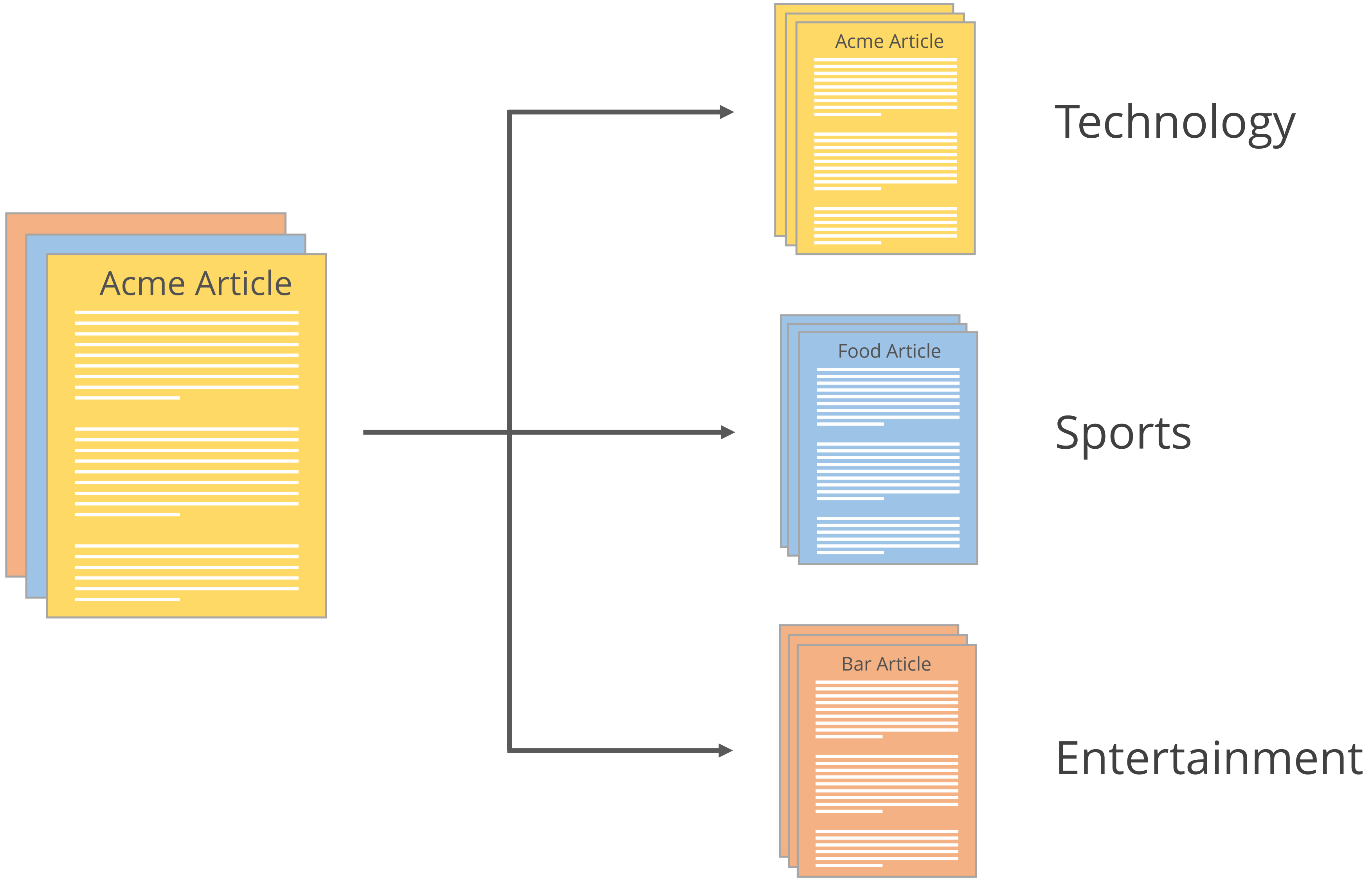
- Understand classification as part of supervised learning
- Demonstrate different classification techniques in Python
- Evaluate classification models

Classification

Topic 1: Definition of Classification

What Is Classification?

A machine learning task that identifies the class to which an instance belongs



Classification: Example

Training a classifier model with respect to the available data

Training Data

Classification Algorithms

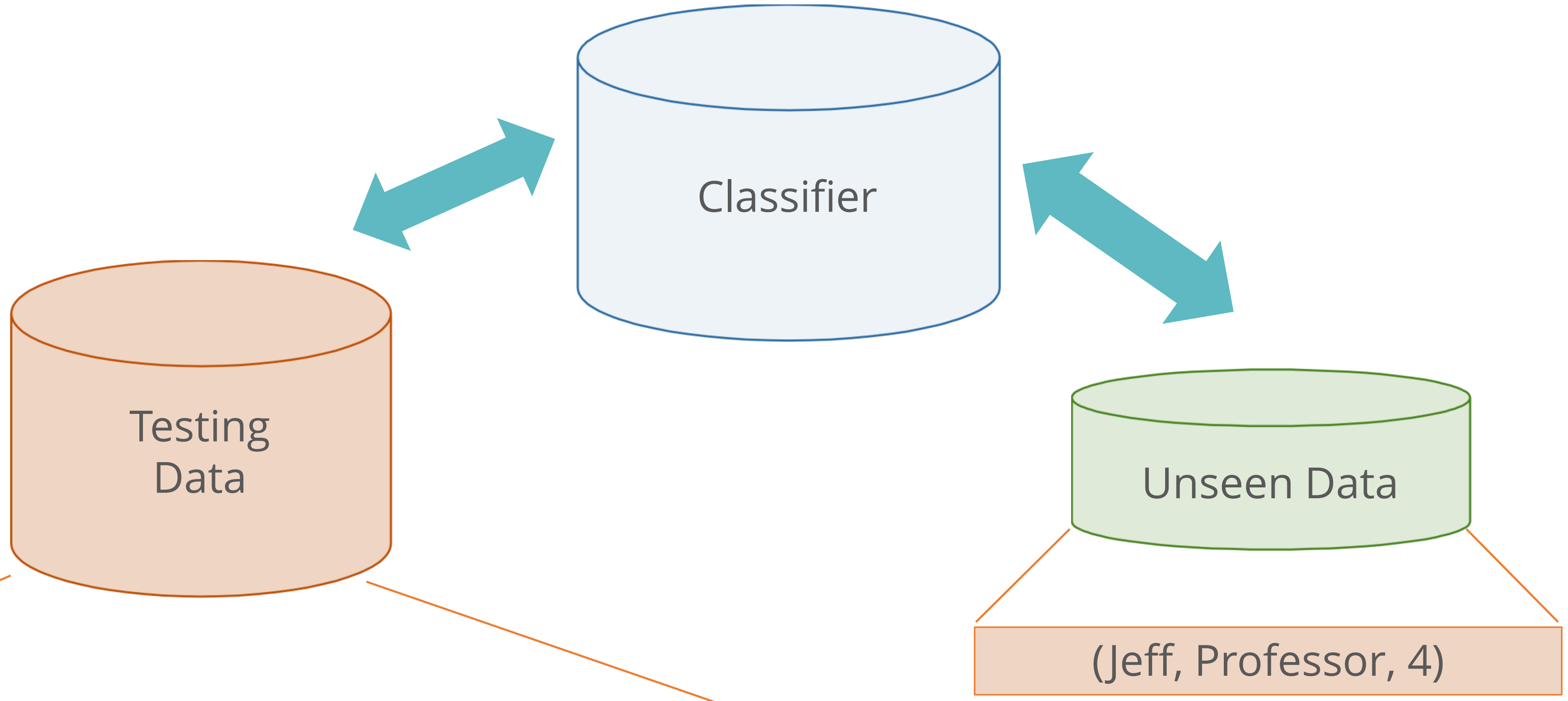
Name	Rank	Years	Tenured
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Associate Prof	6	no
Anne	Associate Prof	3	no

Classifier (Model)

IF rank = “professor”
OR years > 6
THEN tenured = “yes”

Classification: Example

The model will classify if the professors are tenured or not



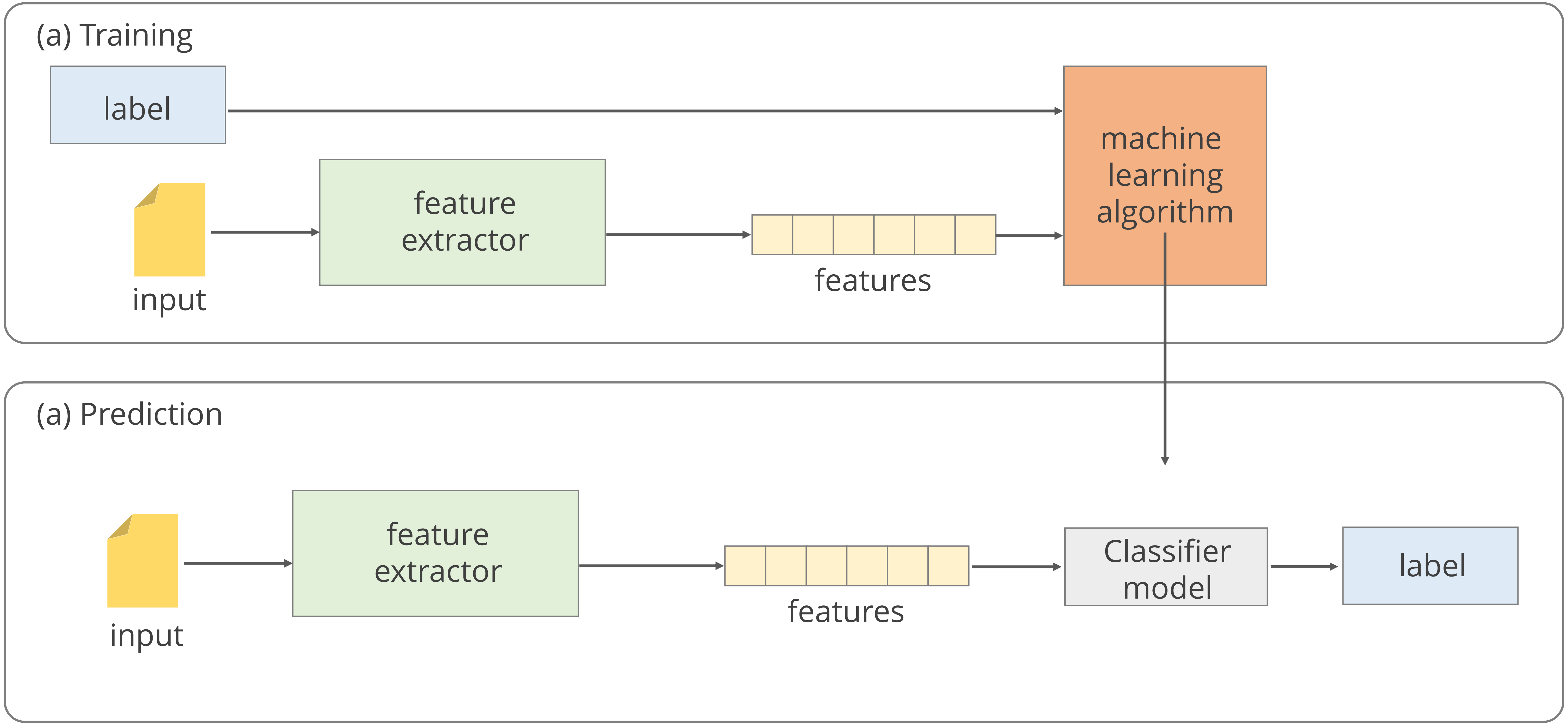
Name	Rank	Years	Tenured
Tom	Assistant Prof	2	no
Merlisa	Assistant Prof	7	no
George	Professor	5	yes
Joseph	Assistant Prof	7	yes

Tenured?

Yes

Classification: Work Flow

A typical classifier model workflow with input training data and output labels

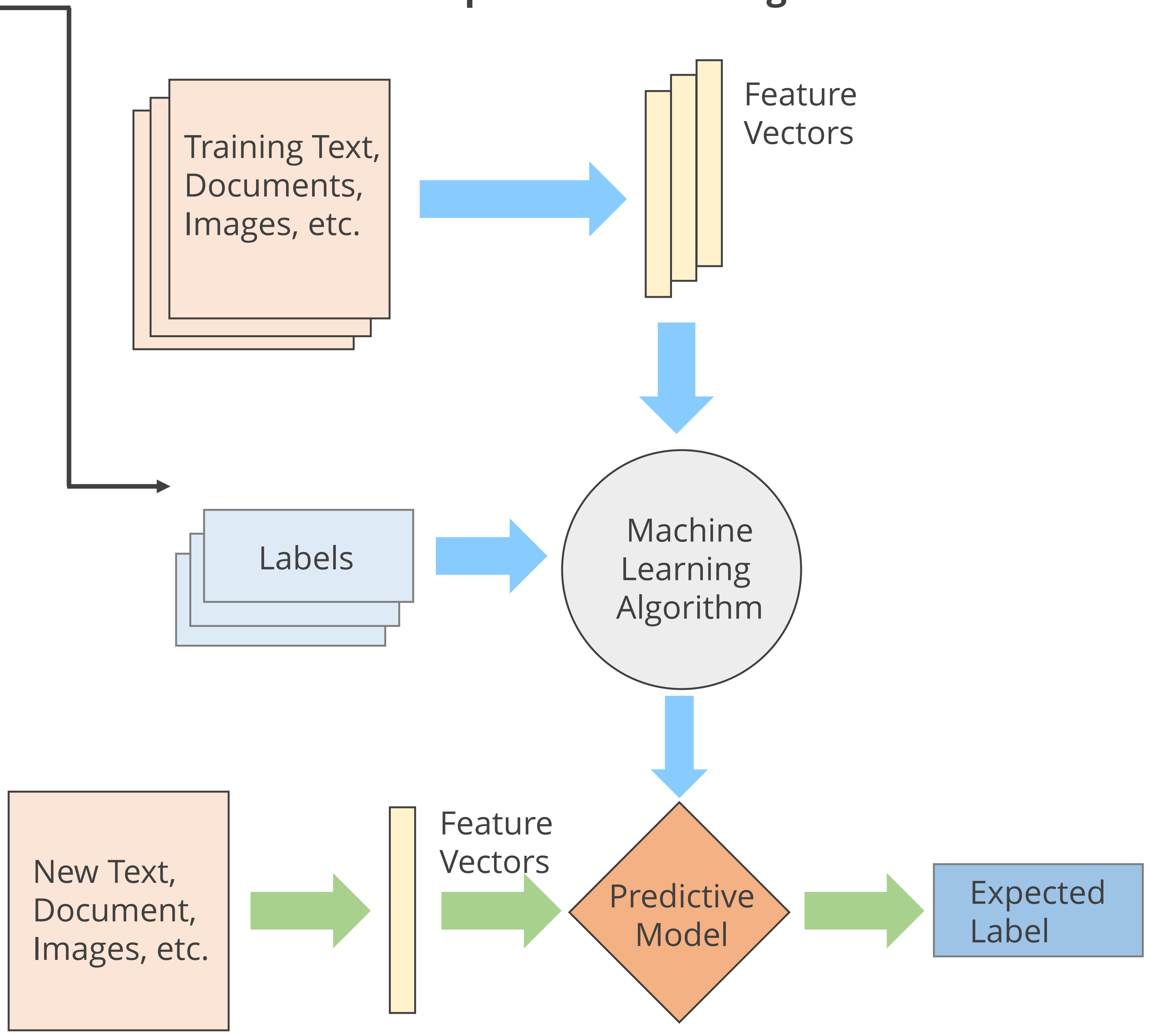


Classification: A Supervised Learning Algorithm

Classification is a supervised learning algorithm as the training data contains labels

Record ID	Age	Spectacle Prescription	Astigmatic	Tear production Rate	Class Label Lenses
1	Young	Myope	No	Reduced	Noncontact
2	Young	Myope	No	Normal	Soft contact
3	Young	Myope	Yes	Reduced	Noncontact
4	Young	Myope	Yes	Normal	Hard contact
5	Young	Hypermetrope	No	Reduced	Noncontact
6	Young	Hypermetrope	No	Normal	Soft contact
7	Young	Hypermetrope	Yes	Reduced	Noncontact
8	Young	Hypermetrope	Yes	Normal	Hard contact
9	Pre-presbyopic	Myope	No	Reduced	Noncontact
10	Pre-presbyopic	Myope	No	Normal	Soft contact
11	Pre-presbyopic	Myope	Yes	Reduced	Noncontact
12	Pre-presbyopic	Myope	Yes	Normal	Hard contact
13	Pre-presbyopic	Hypermetrope	No	Reduced	Noncontact
14	Pre-presbyopic	Hypermetrope	No	Normal	Soft contact
15	Pre-presbyopic	Hypermetrope	Yes	Reduced	Noncontact
16	Pre-presbyopic	Hypermetrope	Yes	Normal	Noncontact
17	Presbyopic	Myope	No	Reduced	Noncontact
18	Presbyopic	Myope	No	Normal	Noncontact
19	Presbyopic	Myope	Yes	Reduced	Noncontact
20	Presbyopic	Myope	Yes	Normal	Hard contact

Supervised Learning Model

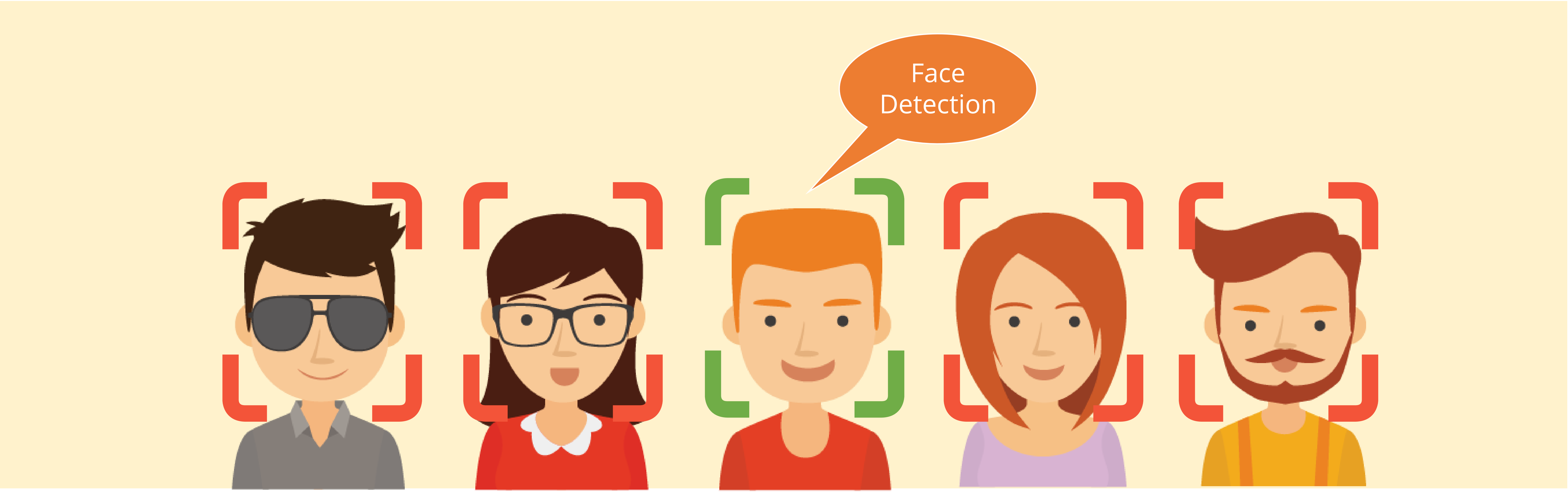


Classification

Topic 2: Use Cases and Algorithms


Sentiment Analysis


Fraud
Detection


Face
Detection

Classification Algorithms

Few of the most commonly used classification algorithms:

Decision Tree

Support Vector Machines (SVM)

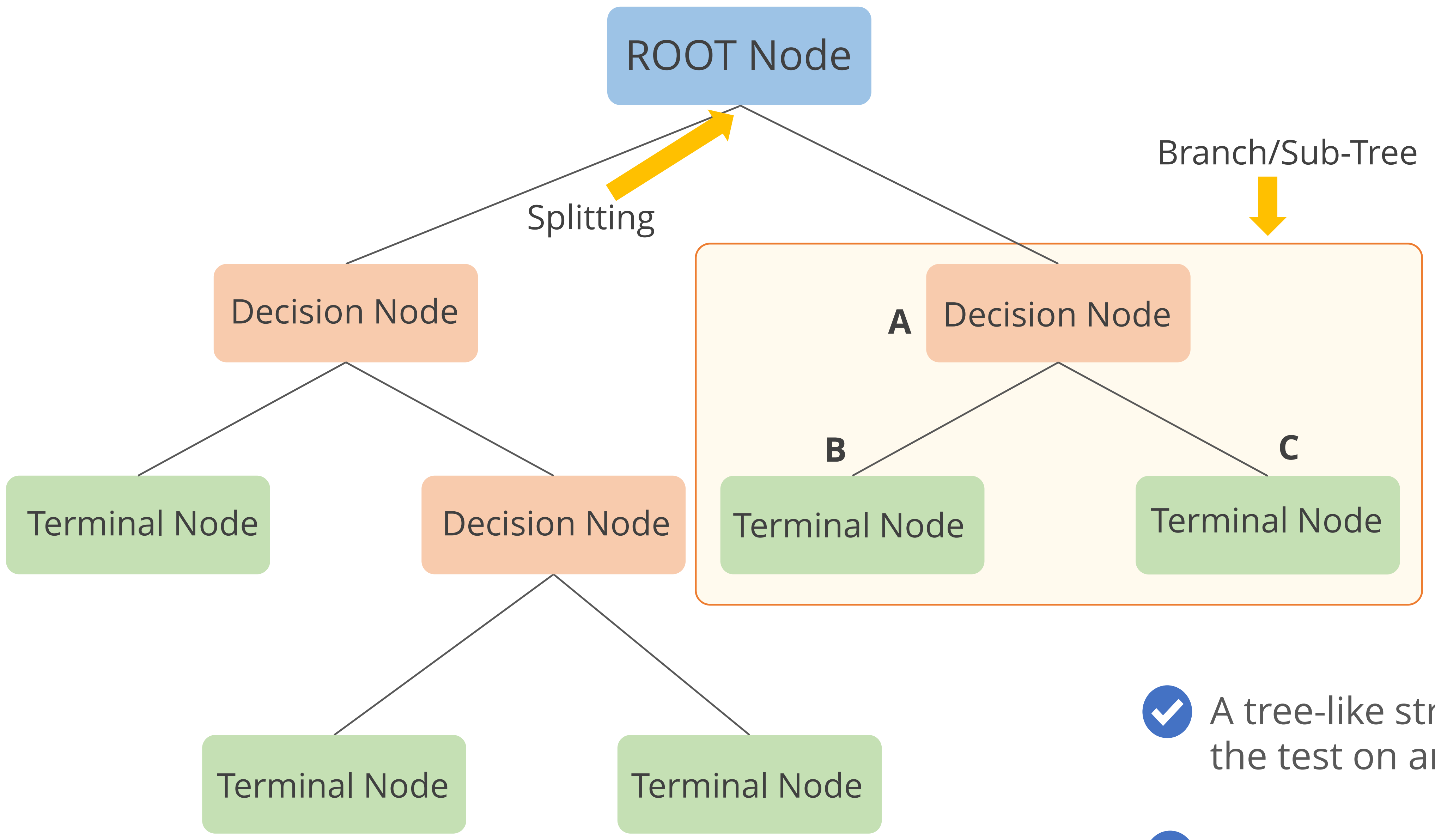
Random Forest

Naîve Bayes Classifier

Classification

Topic 3: Decision Tree Classifier

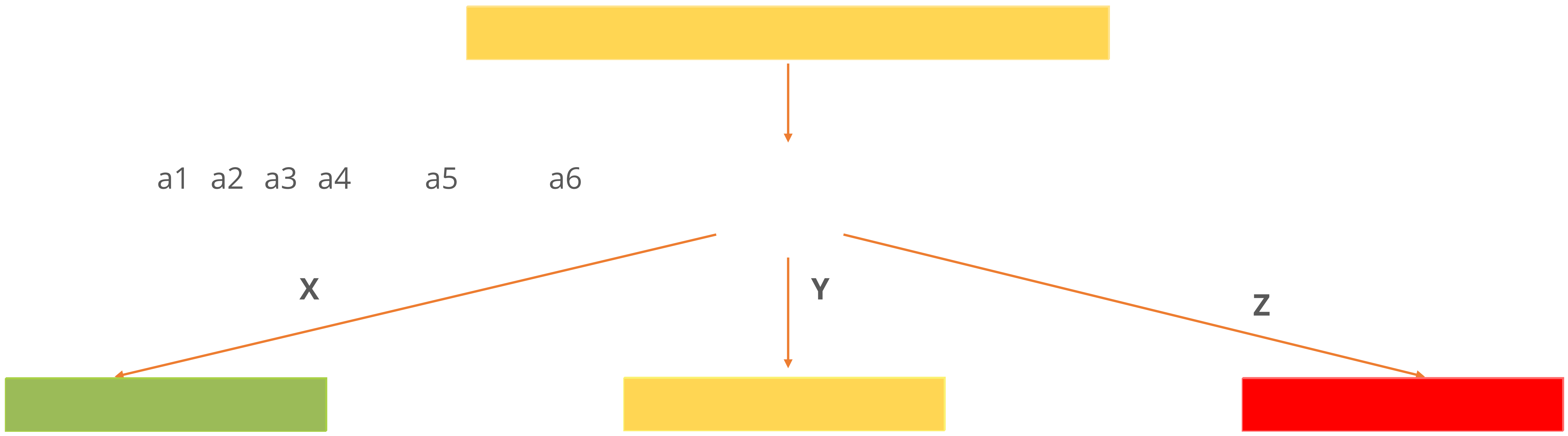
Decision Tree Classifier



- A tree-like structure in which the internal node represents the test on an attribute
- Each branch represents the outcome of the test, and each leaf node represents the class label
- A path from root to leaf represents classification rules

Decision Tree: Schematic Representation

The tree is splitted whenever an impure node is detected



If impure node is detected, select best attribute and continue

If impure node is detected, select best attribute and continue

If pure node is detected, it can be classified as leaf node

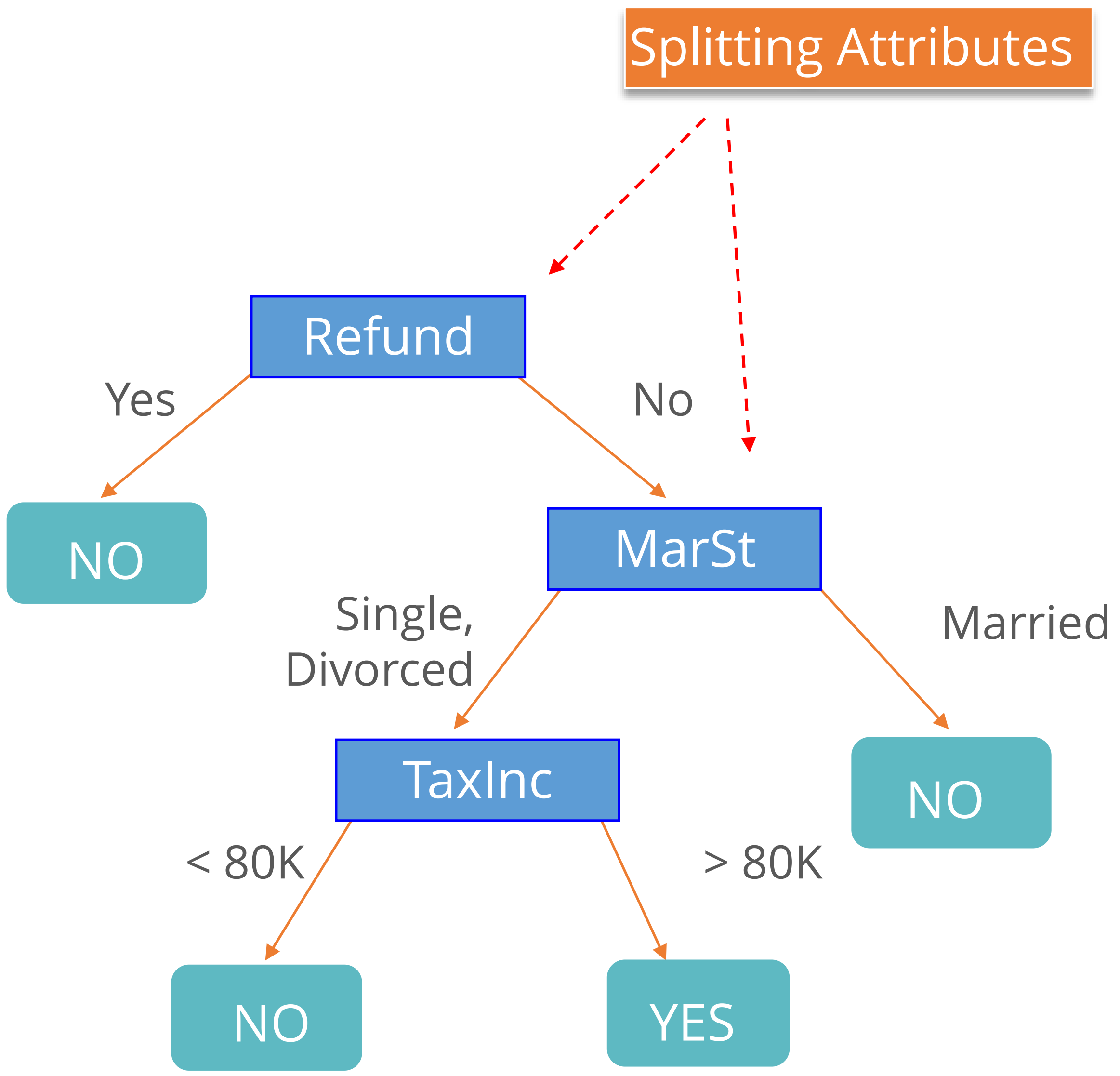
Decision Tree: Example 1

Below example illustrates the splitting attributes with respect to the adjacent training data

categorical categorical continuous class

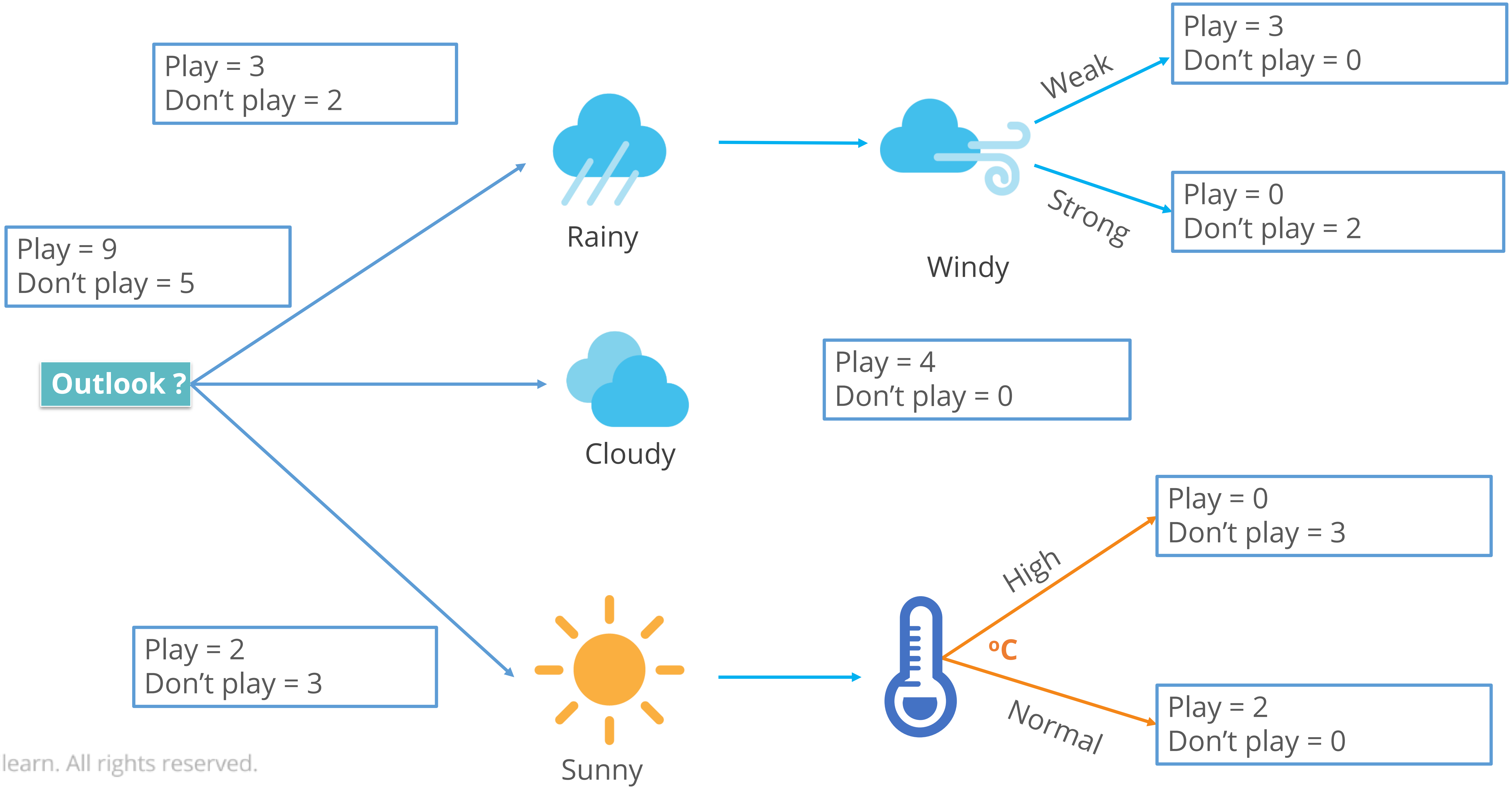
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

Training Data



Decision Tree: Example 2

Forming a decision tree to check if the match will be played or not based on climatic conditions



Decision Tree Formation

Entropy

- Entropy measures the *impurity* of a collection of examples.
- It depends on the distribution of the random variable.
- Entropy, in general, measures the amount of information in a random variable:

$$H(X) = -\sum_{i=1}^{c} p_i \log_2 p_i = \sum_{i=1}^{c} p_i \log_2 1/p_i$$

$X = \{i, \ldots, c\}$

for classification in *c* classes

Information Gain

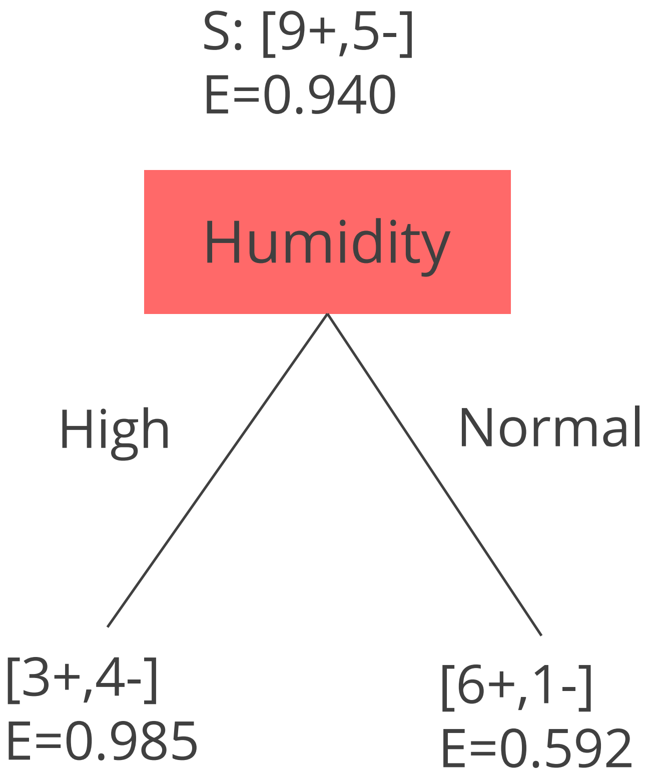
- *Information gain* is the *expected* reduction in entropy caused by partitioning the examples on an attribute.
- Higher the information gain, the more effective the attribute in classifying training data.
- Expected reduction in entropy, given *A*

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

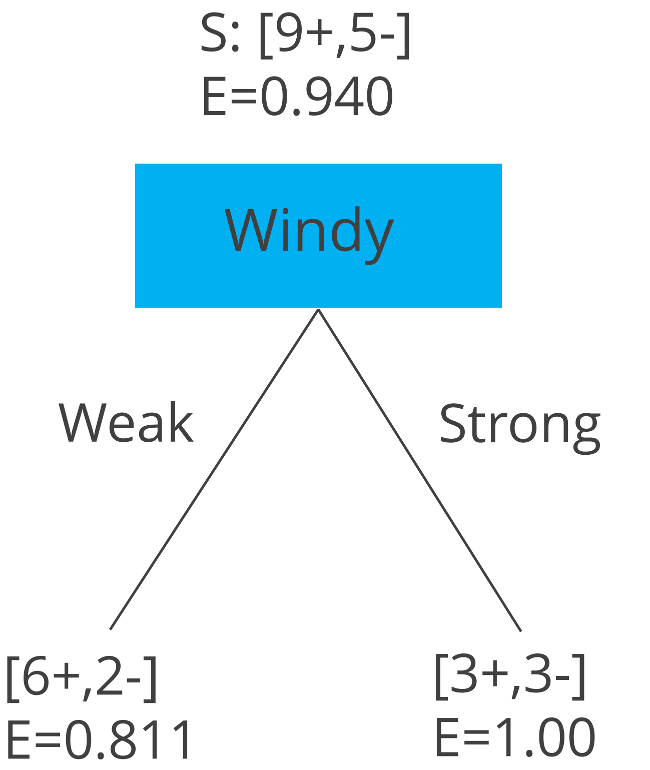
$Values(A)$ = possible values for A

Which Attribute Is the Best Classifier?

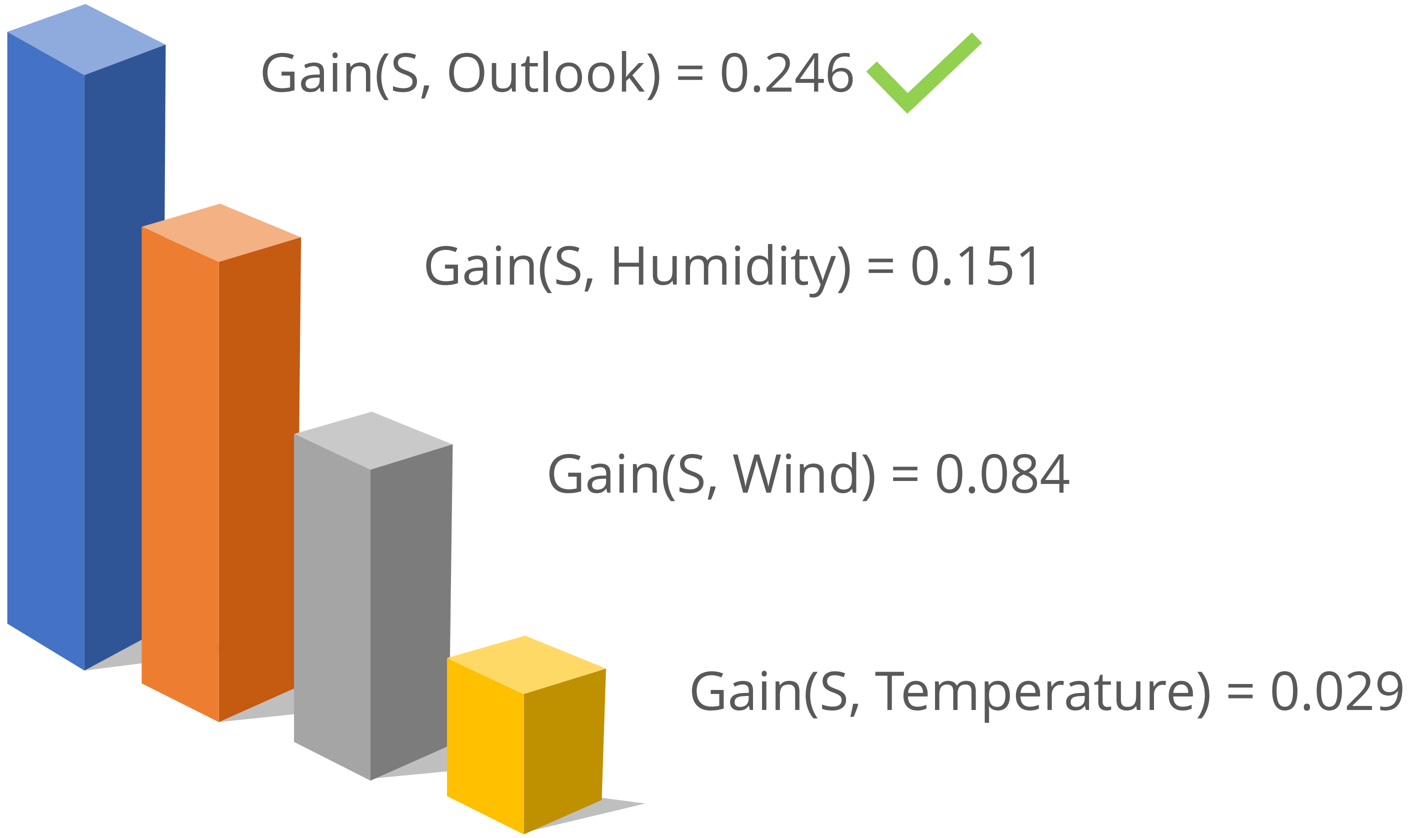
The attribute with the highest information gain is selected as the splitting attribute



Gain (S, Humidity)
= .940 – (7/14).985 – (7/14).592
=.151



Gain (S, Wind)
= .940 – (8/14).811 – (6/14)1.0
=.048



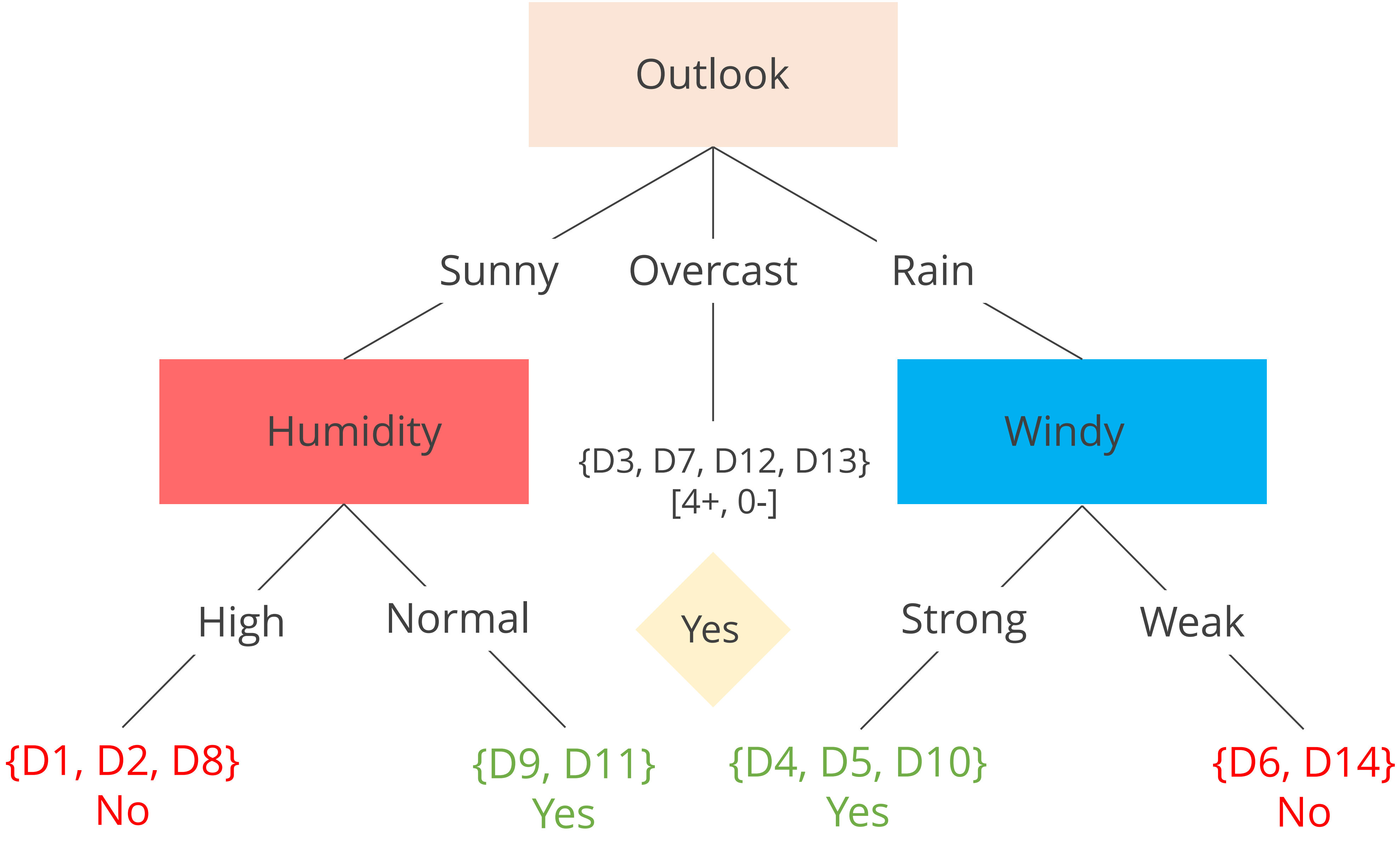
Which Attribute Is the Best Classifier?

The attributes within outlook are further splitted with respect to their gains

- Working on *Outlook=Sunny* node:
  $Gain(S_{Sunny}, Humidity) = 0.970 - 3/5 \times 0.0 - 2/5 \times 0.0 = 0.970$
  $Gain(S_{Sunny}, Wind) = 0.970 - 2/5 \times 1.0 - 3.5 \times 0.918 = 0\ .019$
  $Gain(S_{Sunny}, Temp.) = 0.970 - 2/5 \times 0.0 - 2/5 \times 1.0 - 1/5 \times 0.0 = 0.570$
- *Humidity* provides the best prediction for the target
- For each possible value of *Humidity*, you can add a successor to the tree.

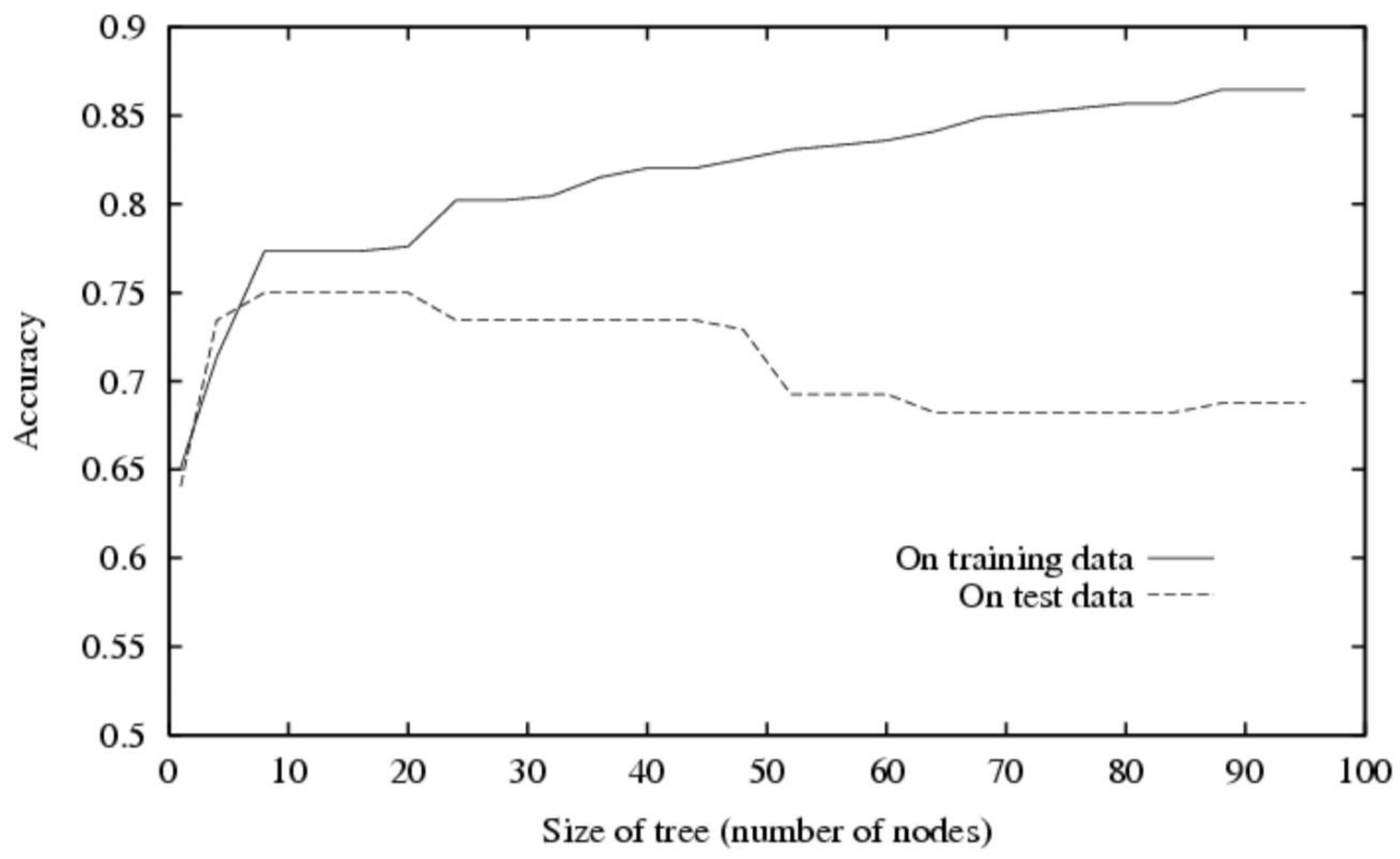
Which Attribute Is the Best Classifier?

Finally, you arrive at leaf nodes with a strong decisions

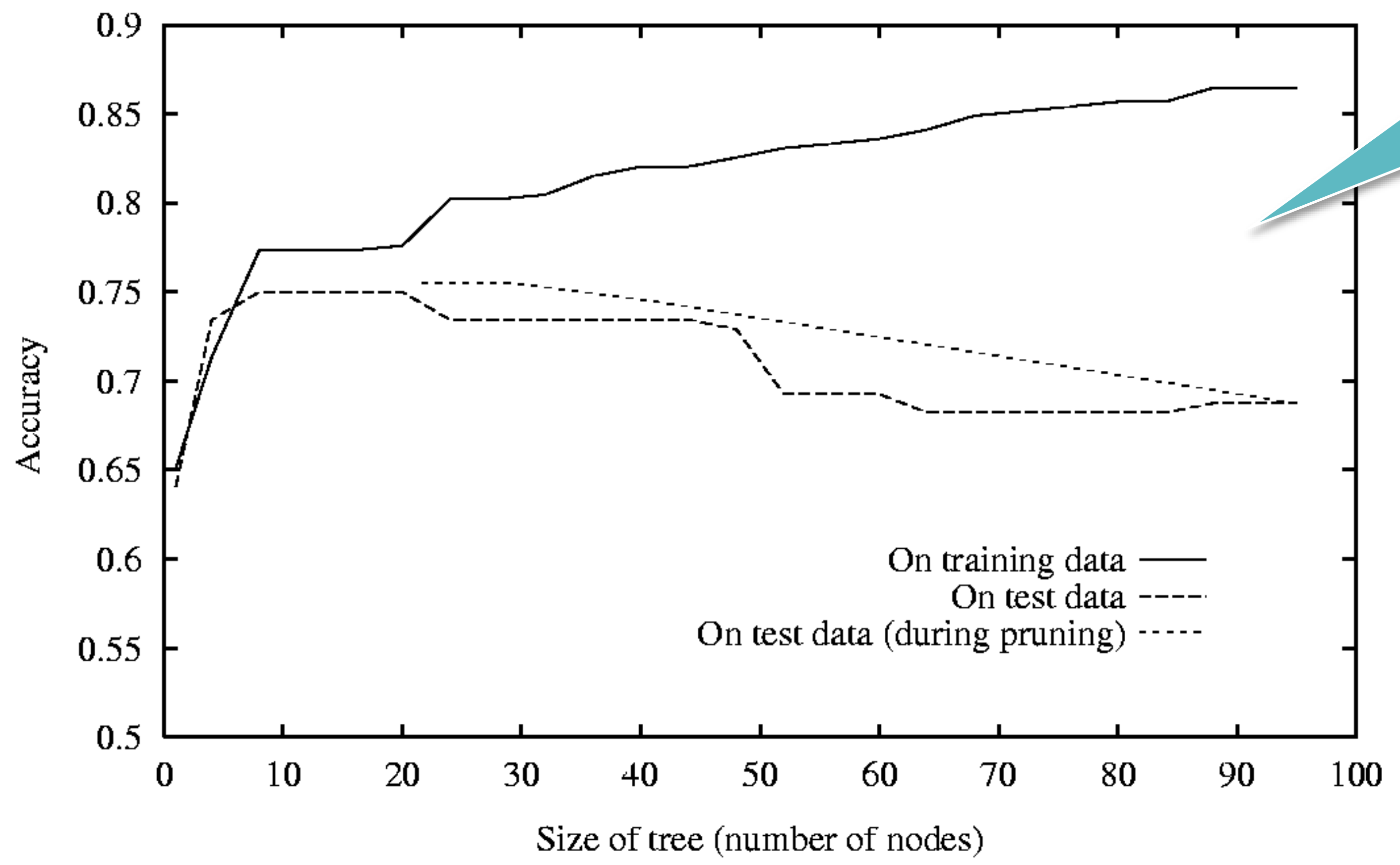


Overfitting of Decision Trees

Overfitting occurs when the learning algorithm continues to develop hypotheses that reduce training set error at the cost of an increased test set error.



Avoiding Overfitting of Decision Trees



Post Pruning

Classification

Topic 4: Random Forest Classifier

Bagging and Bootstrapping

Bagging

A technique for reducing the variance of an estimated prediction function

For classification, a committee of trees each cast a vote for the predicted class

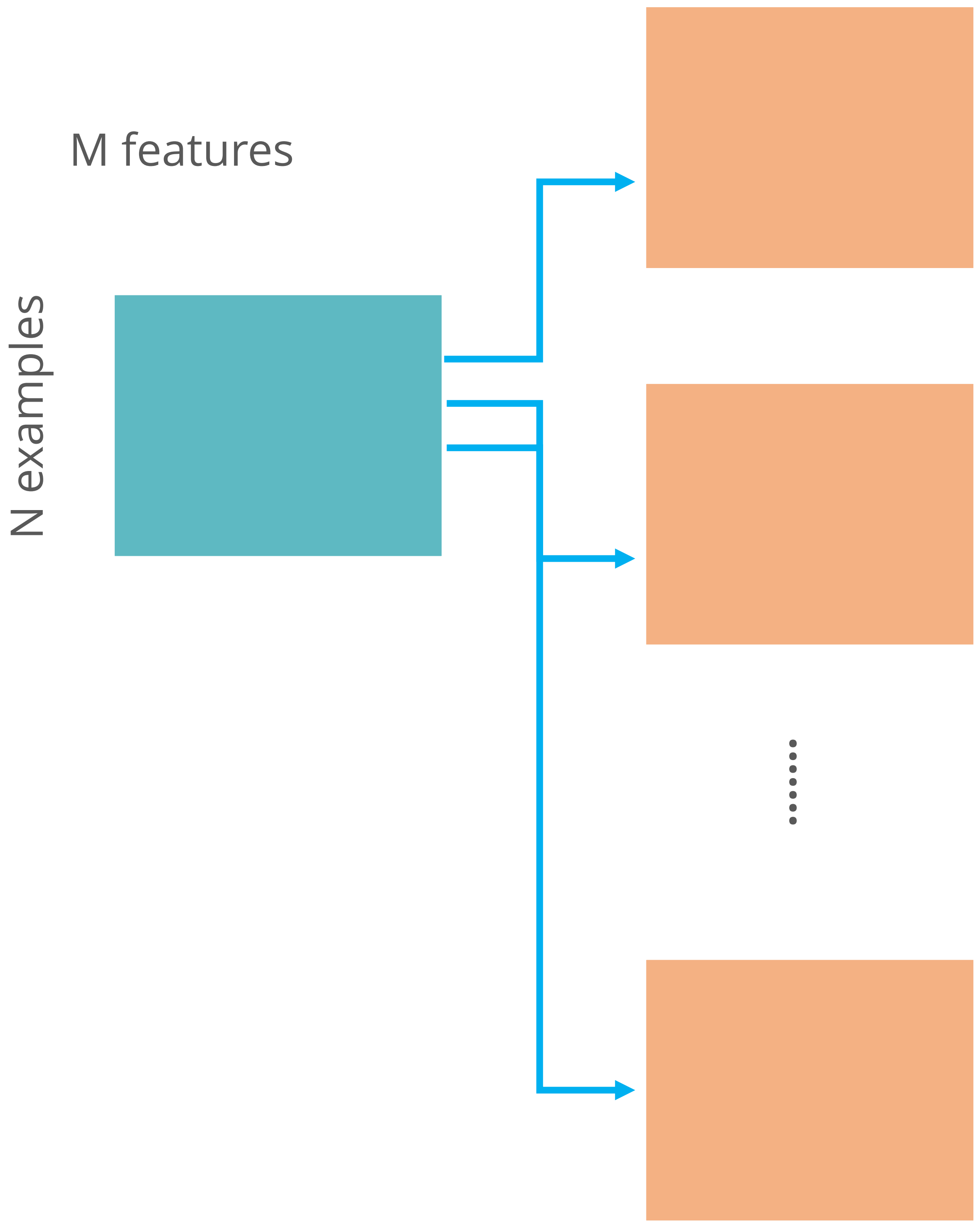
Bootstrapping

Randomly draws datasets with replacement from the training data

Each sample is of the same size as the original training set

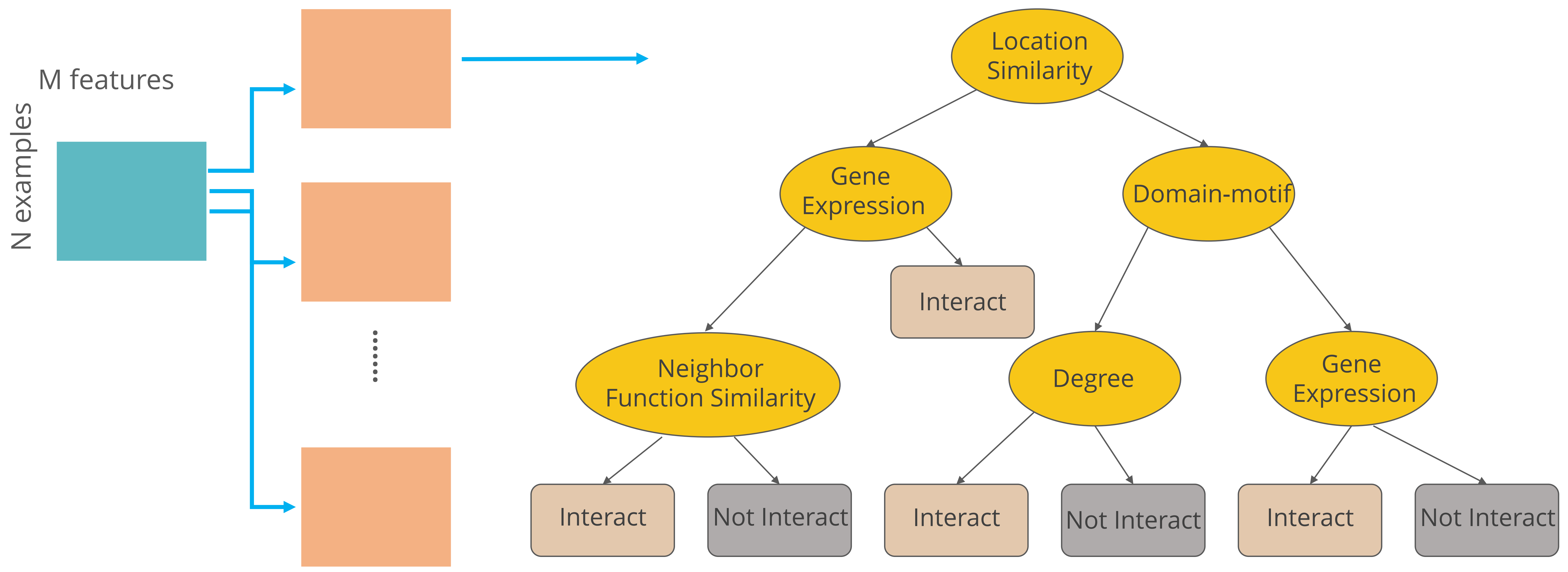
Bagging and Bootstrapping

Create bootstrap samples from the training data

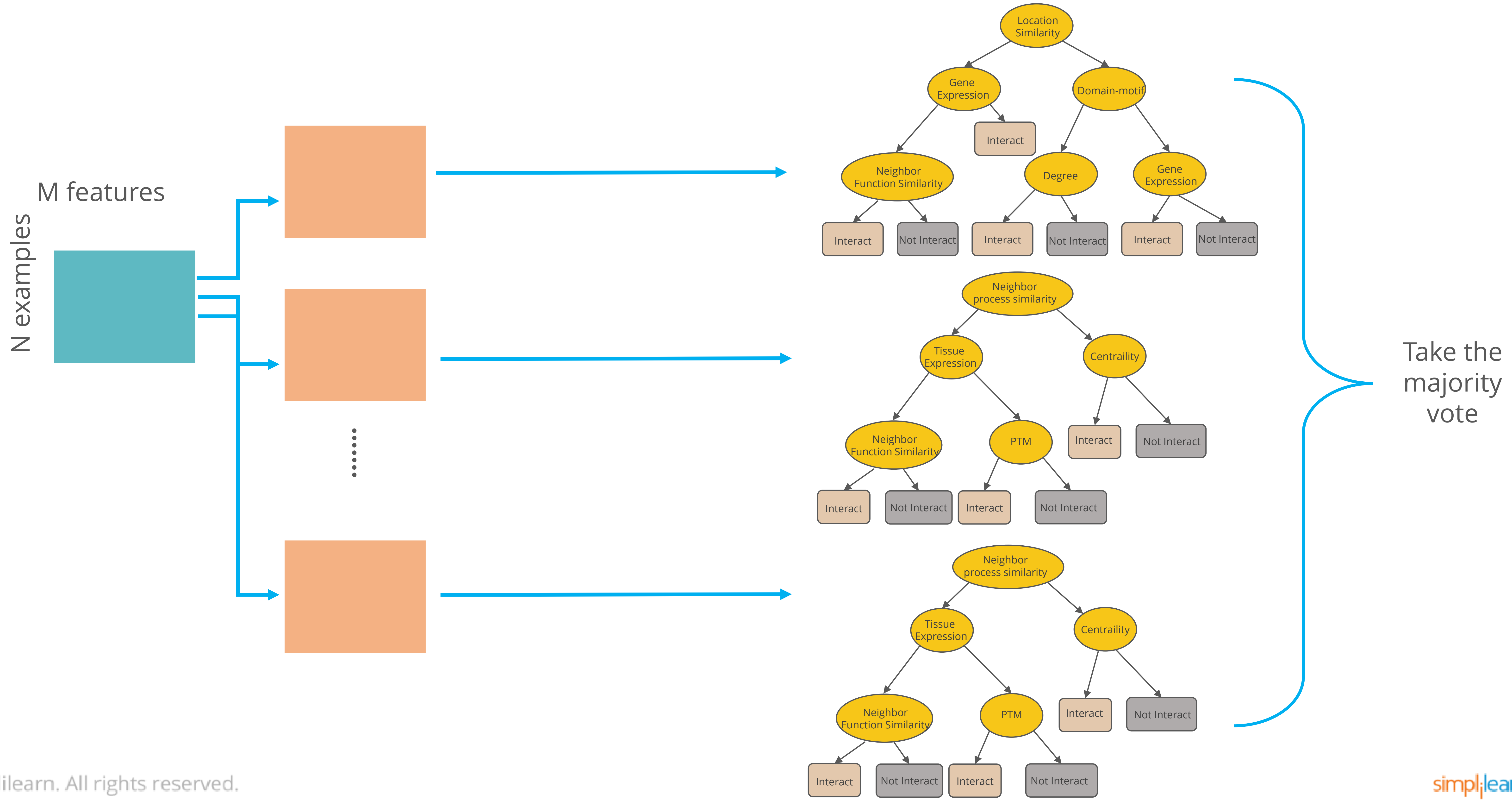


Decision Tree Classifier

Each sample contributes to a decision tree classifier



Random Forest Classifier



Classification

Topic 5: Performance Measures

Confusion Matrix

Focus on the predictive capability of a model

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Accuracy Metric

Ratio of true positives and true negatives to the sum of true positives, true negatives, false negatives, and false positives

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Limitation of Accuracy

Consider a 2-class problem

Number of Class 0 examples = 9990

Number of Class 1 examples = 10

If the model predicts every example to be class 0, accuracy is 9990/10000 = 99.9 %

Hence, accuracy is misleading because the model does not detect any class 1 example

Cost Matrix

Cost matrix takes weights into account

	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	Class=Yes	Class=No
	Class=Yes	C(Yes|Yes)	C(No|Yes)
	Class=No	C(Yes|No)	C(No|No)

Cost of classifying class **j** example as class **i**

$$\text{Weighted Accuracy} = \frac{w_1 a + w_4 d}{w_1 a + w_2 b + w_3 c + w_4 d}$$

Computing Cost of Classification

Cost Matrix	PREDICTED CLASS		
ACTUAL CLASS	C(i|j)	+	-
	+	-1	100
	-	1	0

Model M_1	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	150	40
	-	60	250

Model M_2	PREDICTED CLASS		
ACTUAL CLASS		+	-
	+	250	45
	-	5	200

Accuracy = 80%

Cost = 3910

Accuracy = 90%

Cost = 4255

Cost vs. Accuracy

Count	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

Cost	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	p	q
	Class=No	q	p

Accuracy is proportional to cost if
1. C(Yes|No)=C(No|Yes) = q
2. C(Yes|Yes)=C(No|No) = p

$N = a + b + c + d$

$\text{Accuracy} = (a + d)/N$

$$\begin{aligned}\text{Cost} &= p\,(a + d) + q\,(b + c) \\ &= p\,(a + d) + q\,(N - a - d) \\ &= q\,N - (q - p)(a + d) \\ &= N\,[q - (q-p) \times \text{Accuracy}]\end{aligned}$$

Assisted Practice

Random Forest Classifier

Duration: 15 mins.

Problem Statement: Predict the survival of a horse based on various observed medical conditions. Load the data from "horses.csv" and observe whether it contains missing values. The dataset contains many categorical features; replace them with label encoding. Replace the missing values by the most frequent value in each column. Fit a decision tree classifier and random forest classifier, and observe the accuracy.

Objective: Learn to fit a decision tree, and compare its accuracy with random forest classifier.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Unassisted Practice

Random Forest Classifier

Duration: 15 mins.

Problem Statement: PeerLoanKart is an NBFC (Non-banking Financial Company) that facilitates peer-to-peer loan. It connects people who need money (borrowers) with people who have money (investors). As an investor, you would want to invest in people who showed a profile of having a high probability of paying you back.
You "as an ML expert" create a model that will help predict whether a borrower will pay the loan or not.

Objective: Increase profits up to 20% as NPA will be reduced due to loan disbursal for only creditworthy borrowers

Note: This practice is not graded. It is only intended for you to apply the knowledge you gained to solve real-world problems.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Import Libraries

Code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split
```

Get the Data

Code

```
loans = pd.read_csv('loan_borowwer_data.csv')
loans.describe()
```

```
loans.describe()
```

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.util	inq.last.6mths	delinq.2yrs
count	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9578.000000	9.578000e+03	9578.000000	9578.000000	9578.00000
mean	0.804970	0.122640	319.089413	10.932117	12.606679	710.846314	4560.767197	1.691396e+04	46.799236	1.577469	0.16370
std	0.396245	0.026847	207.071301	0.614813	6.883970	37.970537	2496.930377	3.375619e+04	29.014417	2.200245	0.54621
min	0.000000	0.060000	15.670000	7.547502	0.000000	612.000000	178.958333	0.000000e+00	0.000000	0.000000	0.00000
25%	1.000000	0.103900	163.770000	10.558414	7.212500	682.000000	2820.000000	3.187000e+03	22.600000	0.000000	0.00000
50%	1.000000	0.122100	268.950000	10.928884	12.665000	707.000000	4139.958333	8.596000e+03	46.300000	1.000000	0.00000
75%	1.000000	0.140700	432.762500	11.291293	17.950000	737.000000	5730.000000	1.824950e+04	70.900000	2.000000	0.00000
max	1.000000	0.216400	940.140000	14.528354	29.960000	827.000000	17639.958330	1.207359e+06	119.000000	33.000000	13.00000

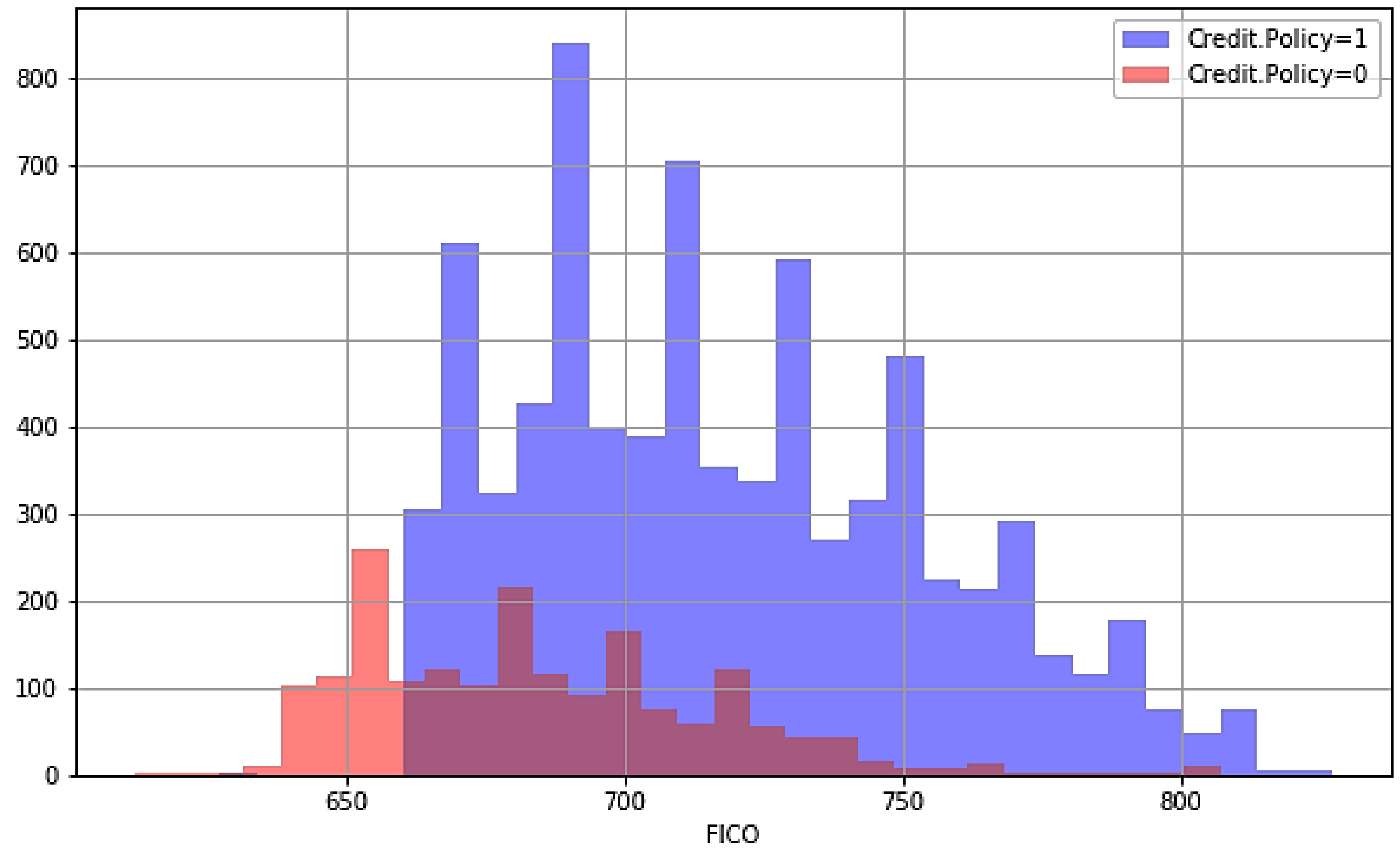
Exploratory Data Analysis

Create a histogram of two FICO distributions on top of each other, one for each credit.policy outcome.

Code

```
plt.figure(figsize=(10,6))
loans[loans['credit.policy']==1]['fico'].hist(alpha=0.5,color='blue',
bins=30,label='Credit.Policy=1')
loans[loans['credit.policy']==0]['fico'].hist(alpha=0.5,color='red',
bins=30,label='Credit.Policy=0')
plt.legend()
plt.xlabel('FICO')
```

Exploratory Data Analysis



Exploratory Data Analysis

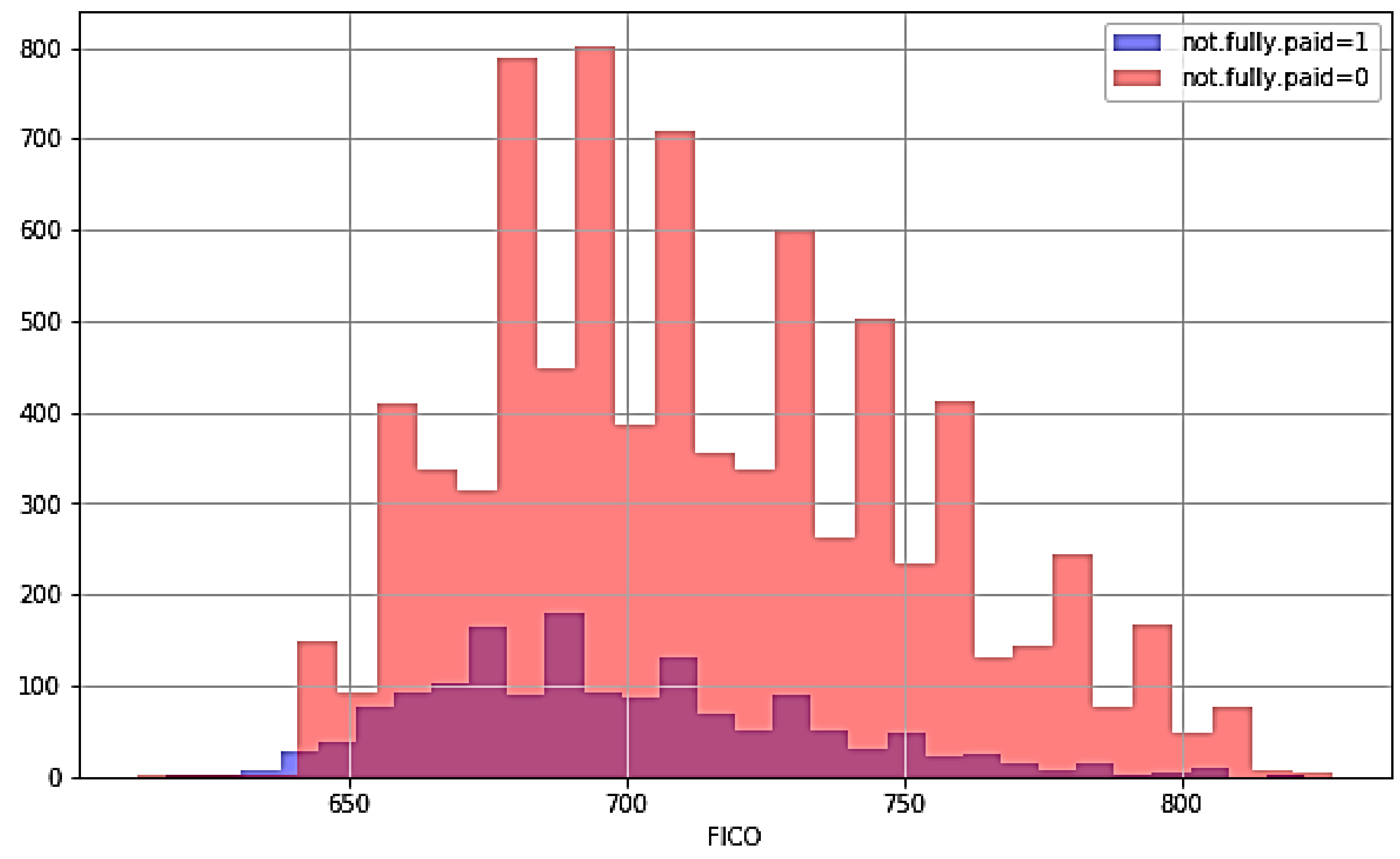
Create a similar figure; select the not.fully.paid column

Code

```
plt.figure(figsize=(10,6))
loans[loans['not.fully.paid']==1]['fico'].hist(alpha=0.5,color='blue',
bins=30,label='not.fully.paid=1')
loans[loans['not.fully.paid']==0]['fico'].hist(alpha=0.5,color='red',
bins=30,label='not.fully.paid=0')
plt.legend()
plt.xlabel('FICO')
```

Exploratory Data Analysis

```
Text(0.5,0,'FICO')
```



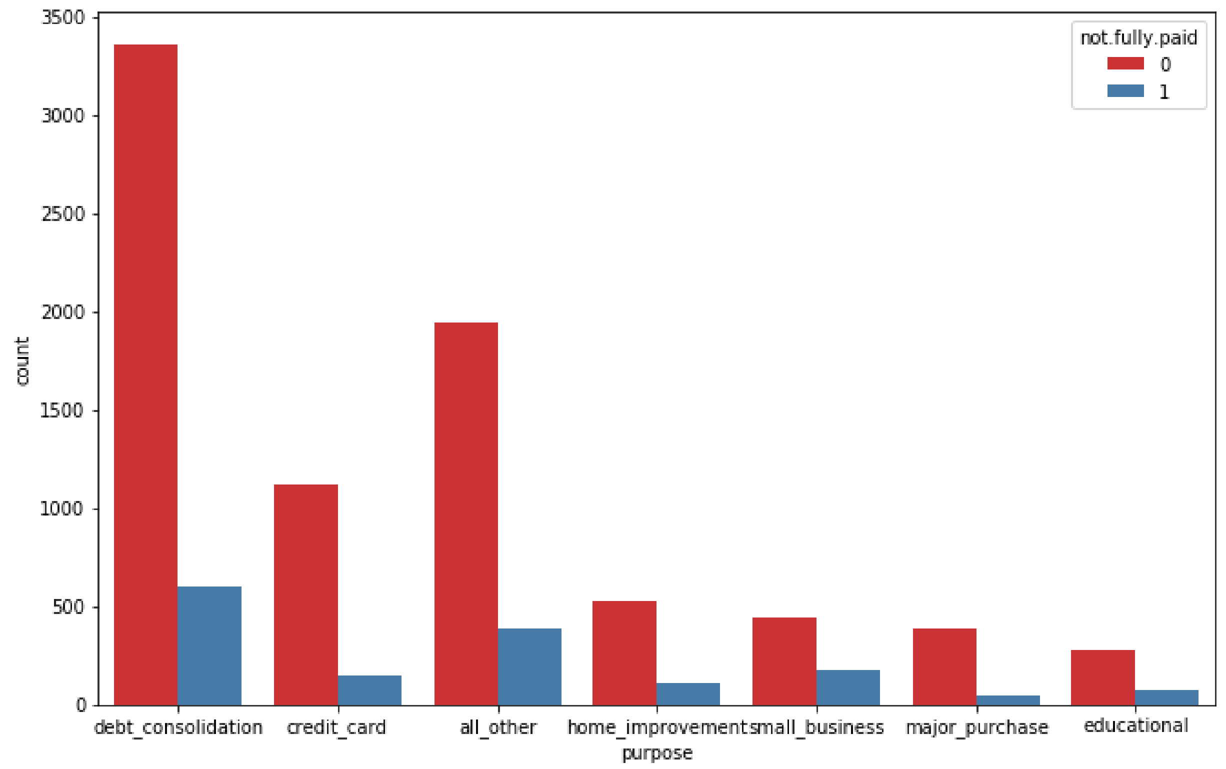
Exploratory Data Analysis

Create a countplot using seaborn showing the counts of loans by purpose, with the hue defined by not.fully.paid.

Code

```
plt.figure(figsize=(11,7))
sns.countplot(x='purpose',hue='not.fully.paid',data=loans,palette='Set1')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xe81fd30>
```



simplilearn

Setting Up the Data

Create a list of elements, containing the string “purpose.” Call this list cat_feats.

Code

```
cat_feats = ['purpose’]
```

Setting Up the Data

Now use pd.get_dummies (loans,columns=cat_feats,drop_first=True) to create a fixed larger data frame that has new feature columns with dummy variables. Set this data frame as final_data.

Code

```
final_data = pd.get_dummies(loans,columns=cat_feats,drop_first=True)
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9578 entries, 0 to 9577
Data columns (total 19 columns):
credit.policy                 9578 non-null int64
int.rate                      9578 non-null float64
installment                   9578 non-null float64
log.annual.inc                9578 non-null float64
dti                           9578 non-null float64
fico                          9578 non-null int64
days.with.cr.line             9578 non-null float64
revol.bal                     9578 non-null int64
revol.util                    9578 non-null float64
inq.last.6mths                9578 non-null int64
delinq.2yrs                   9578 non-null int64
pub.rec                       9578 non-null int64
not.fully.paid                9578 non-null int64
purpose_credit_card           9578 non-null uint8
purpose_debt_consolidation    9578 non-null uint8
purpose_educational           9578 non-null uint8
purpose_home_improvement      9578 non-null uint8
purpose_major_purchase        9578 non-null uint8
purpose_small_business        9578 non-null uint8
dtypes: float64(6), int64(7), uint8(6)
memory usage: 1.0 MB
```

Train-Test Split

Code

```
X = final_data.drop('not.fully.paid',axis=1)
y = final_data['not.fully.paid']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=101)
```

Training Decision Tree Model

Code

```
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
```

```
Out[24]: DecisionTreeClassifier(class_weight=None, criterion='gini', 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=None,
                     splitter='best')
```

Evaluating Decision Tree

Create predictions from the test set, and create a classification report and a confusion matrix.

Code

```
predictions = dtree.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,predictions))
```

```
             precision    recall  f1-score   support

          0       0.85      0.82      0.84      2431
          1       0.19      0.23      0.21       443

avg / total       0.75      0.73      0.74      2874
```

Confusion Matrix

Code

```
print(confusion_matrix(y_test,predictions))
```

```
[[1993  438]
 [ 340  103]]
```

Training Random Forest Model

Code

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=600)
rfc.fit(X_train,y_train)
```

```
Out[37]: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                    max_depth=None, max_features='auto', 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, n_estimators=600, n_jobs=1,
                    oob_score=False, random_state=None, verbose=0,
                    warm_start=False)
```

Evaluating Random Forest Model

Code

```
predictions = rfc.predict(X_test)
from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,predictions))
```

	precision	recall	f1-score	support
0	0.85	1.00	0.92	2431
1	0.62	0.02	0.04	443
avg / total	0.81	0.85	0.78	2874

Printing the Confusion Matrix

Code

```
print(confusion_matrix(y_test,predictions))
```

```
[[2425    6]
 [ 433   10]]
```

Classification

Topic 7: Naïve Baye’s Classifier

Naïve Baye's Classifier and Baye's Theorem

Classification technique based on Baye's theorem

Baye's Theorem

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

Where,

- P(A) – Class Prior Probability
- P(B|A) – Likelihood
- P(A|B) – Posterior Probability
- P(A) - Predictor Prior Probability

Note: Naive Baye's classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naïve Baye's Classifier: Example

As the first step toward prediction using naïve bayes, you will have to estimate frequency of each and every attribute

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

Frequency Table		Play	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	3	2

Frequency Table		Play	
		Yes	No
Humidity	High	3	4
	Normal	6	1

Frequency Table		Play	
		Yes	No
Wind	Strong	6	2
	Weak	3	3

Building Likelihood Tables

Calculating likelihood of each attribute

Likelihood Table		Play		
		Yes	No	
Outlook	Sunny	3/9	2/5	5/14
	Overcast	4/9	0/5	4/14
	Rainy	3/9	2/5	5/14
		10/14	4/14	

$P(B|A) = P(Sunny|Yes) = 3/9 = 0.33$

$P(B) = P(Sunny) = 5/14 = 0.36$

$P(A) = P(Yes) = 10/14 = 0.71$

Similarly likelihood of "No" given Sunny is:

$P(A|B) = P(No|Sunny) = P(Sunny|No)* P(No) / P(Sunny) = (0.4 \times 0.36) /0.36 = 0.40$

Building Likelihood Tables

Likelihood table for Humidity

Likelihood Table		Play		
		Yes	No	
Humidity	High	3/9	4/5	7/14
	Normal	6/9	1/5	7/14
		9/14	5/14	

Likelihood table for Wind

Likelihood Table		Play		
		Yes	No	
Wind	Weak	6/9	2/5	8/14
	Strong	3/9	3/5	6/14
		9/14	5/14	

$P(Yes \mid High)$ = 0.33 x 0.6 / 0.5 = 0.42

$P(No \mid High)$ = 0.8 x 0.36 / 0.5 = 0.58

$P(Yes \mid Weak)$ = 0.67 x 0.64 / 0.57 = 0.75

$P(No \mid Weak)$ = 0.4 x 0.36 / 0.57 = 0.25

Getting the Output

Outlook	=	Rain
Humidity	=	High
Wind	=	Weak
Play	=	?

Likelihood of **"Yes"** = P(Outlook = Rain|**Yes**)*P(Humidity= High|**Yes**)* P(Wind= Weak|**Yes**)*P(**Yes**) = 2/9 * 3/9 * 6/9 * 9/14 = 0.0199

Likelihood of **"No"** = P(Outlook = Rain|**No**)*P(Humidity= High|**No**)* P(Wind= Weak|**No**)*P(**No**) = 2/5 * 4/5 * 2/5 * 5/14 = 0.0166

Getting the Output

Normalizing the values

P(**Yes**) = 0.0199 / (0.0199+ 0.0166) = 0.55

P(**No**) = 0.0166 / (0.0199+ 0.0166) = 0.45

The model predicts
that there is a 55%
chance
that there will be
game tomorrow

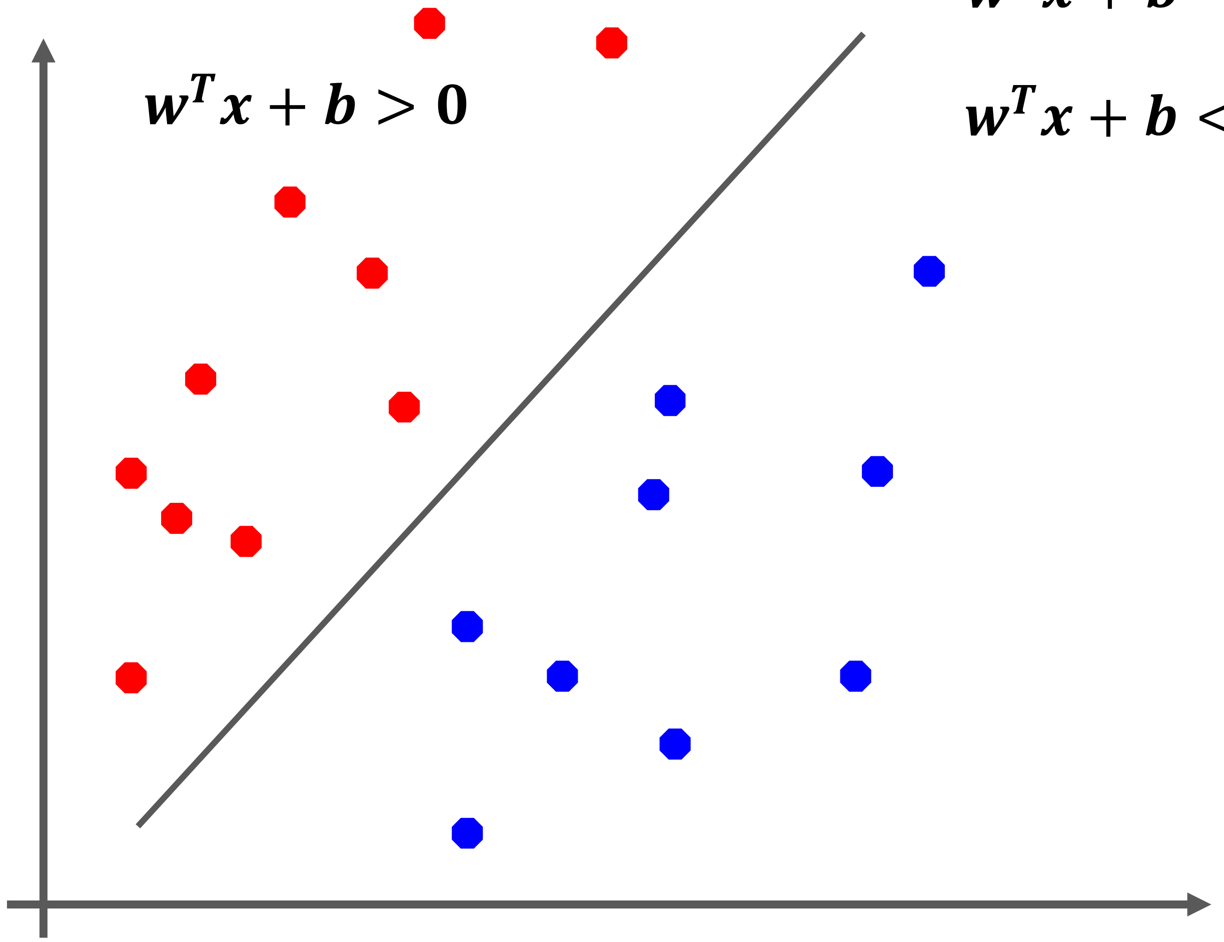
Classification

Topic 8: Support Vector Machines

simplilearn

Linear Separators

Consider a binary separation which can be viewed as the task of separating classes in feature space.



$\boldsymbol{w^T x + b > 0}$

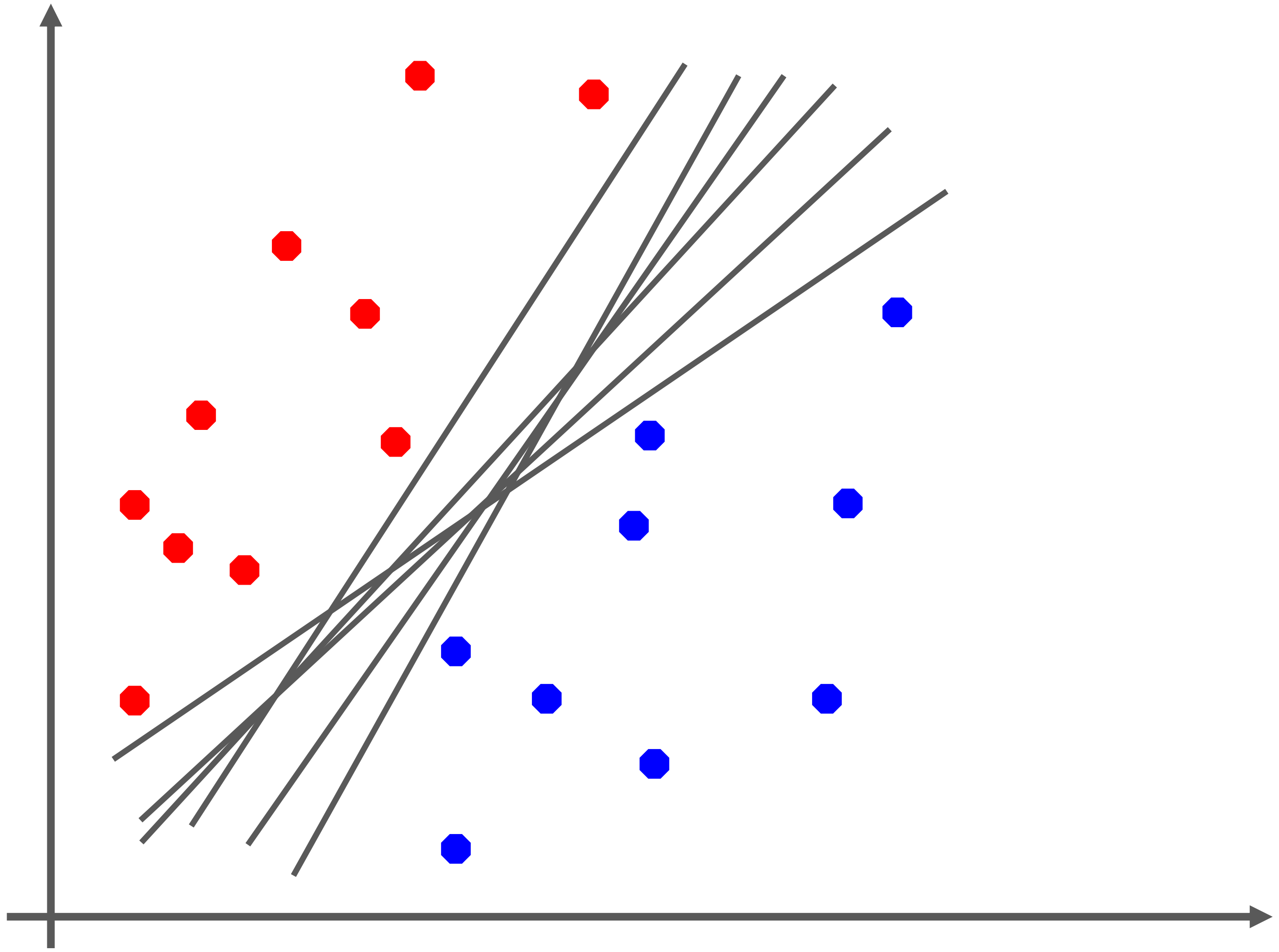
$\boldsymbol{w^T x + b = 0}$

$\boldsymbol{w^T x + b < 0}$

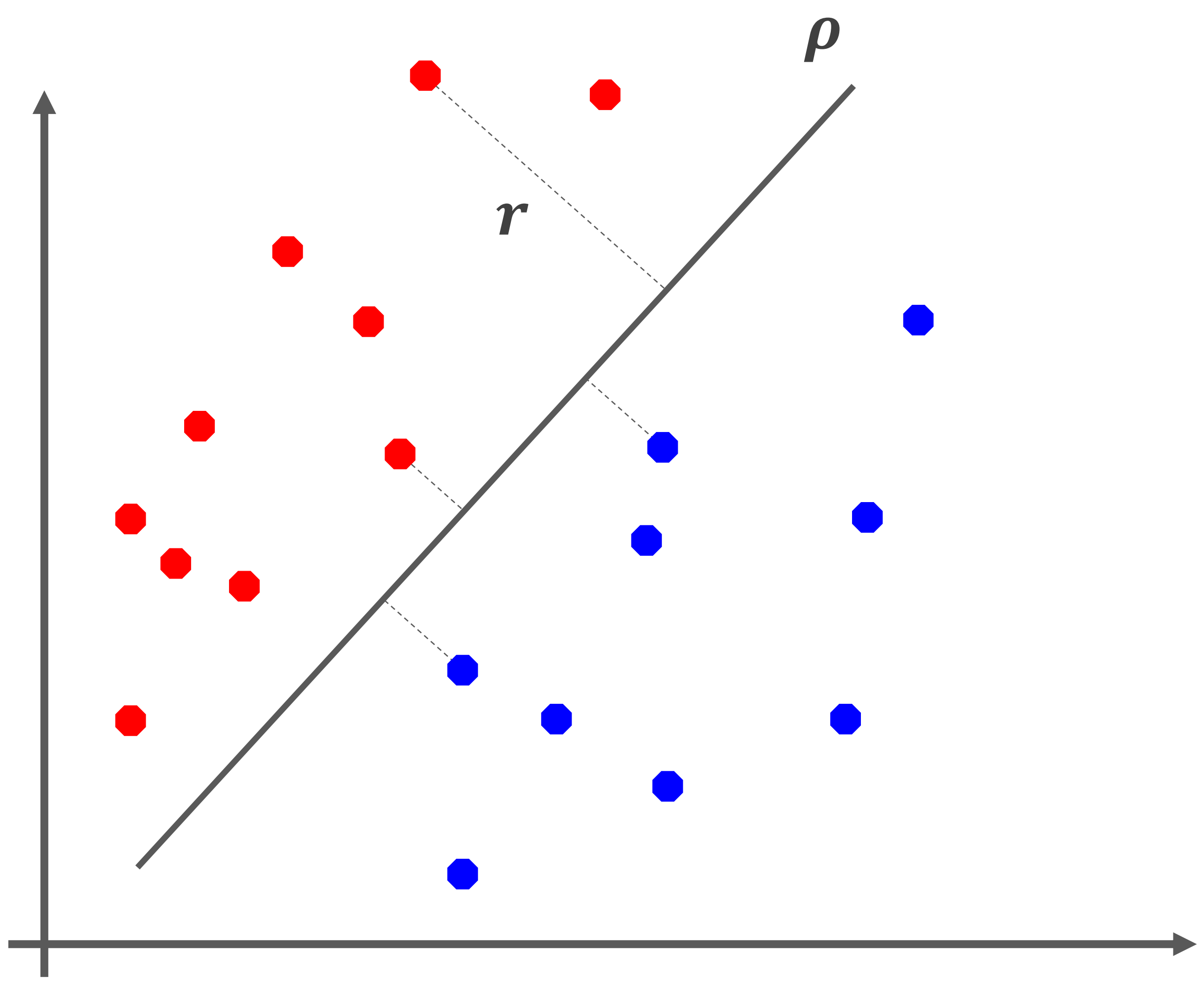
$$\boldsymbol{f(x) = sign(w^T x + b)}$$

Optimal Separation

It’s difficult to evaluate the optimal separator.



Concept of Classification Margin



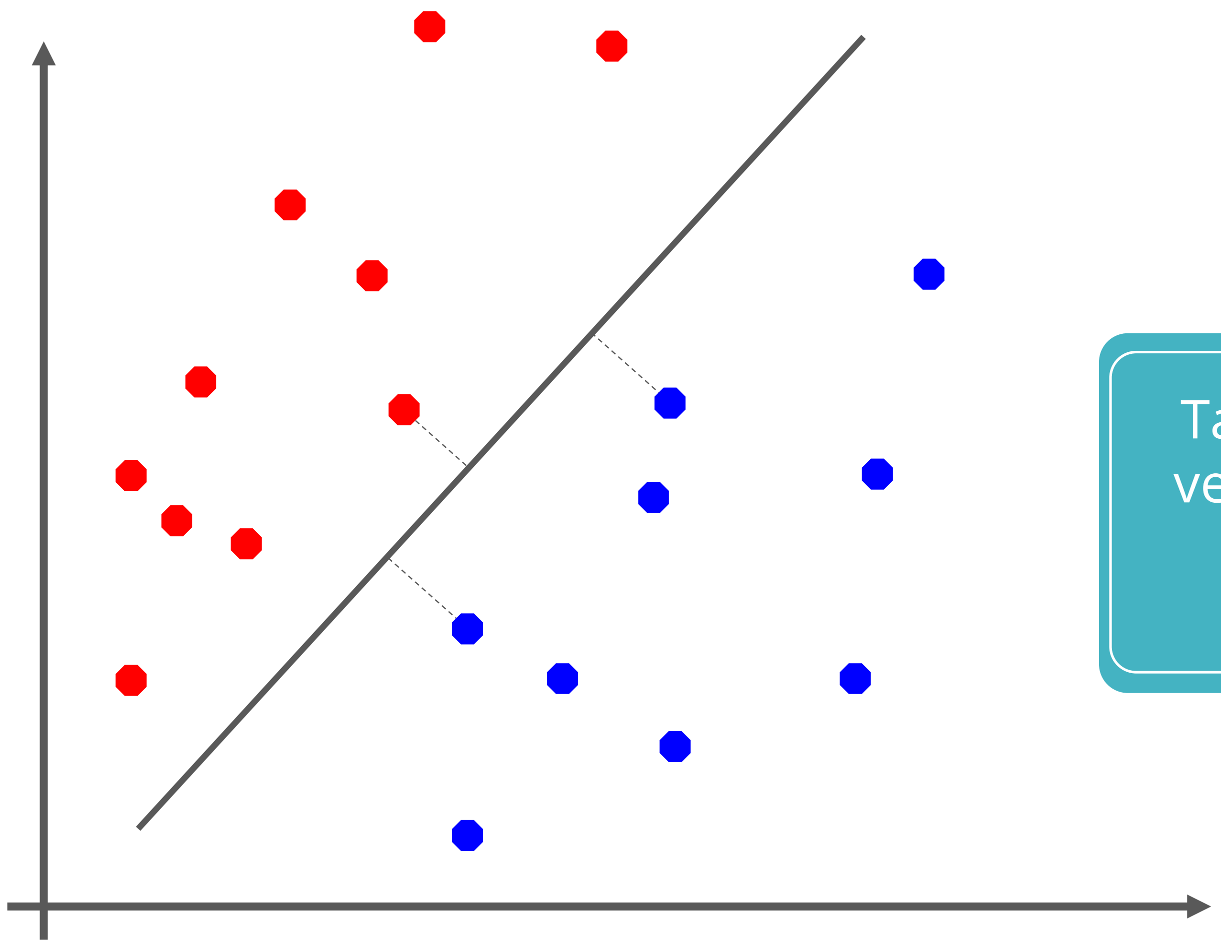
- Distance from example x_i to the separator is $r = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$

- Closest to the hyperplane are *support vectors*.

- *Margin ρ* of the separator is the distance between support vectors.

Maximizing Classification Margin

Helps generalize the predictions and perform better on the test data by not overfitting the model to the training data



Takes care only of the support vectors, ignoring other training examples

Linear SVM: Mathematically

Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{array}{ll} \mathbf{w}^T\mathbf{x}_i + b \leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^T\mathbf{x}_i + b \geq \rho/2 & \text{if } y_i = 1 \end{array} \quad \Leftrightarrow \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq \rho/2$$

For every support vector $\mathbf{x}_s$, the above inequality is an equality. After rescaling **w** and *b* by $\rho/2$ in the equality, you obtain the distance between each $\mathbf{x}_s$. The hyperplane is:

$$r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

Then the margin can be expressed through (rescaled) **w** and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVM: Mathematically

Now, you can formulate the quadratic optimization problem:

Find **w** and *b* such that $\rho = \frac{2}{\|\mathbf{w}\|}$ is maximized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

You can reformulate the problem as:

Find **w** and *b* such that

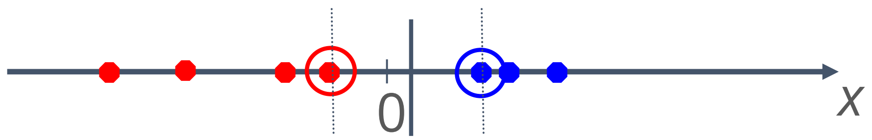
$\Phi(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Nonlinear SVMs

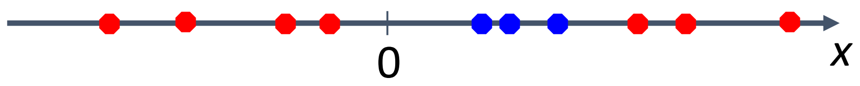
Scenario 1

- Datasets that are linearly separable with some noise:



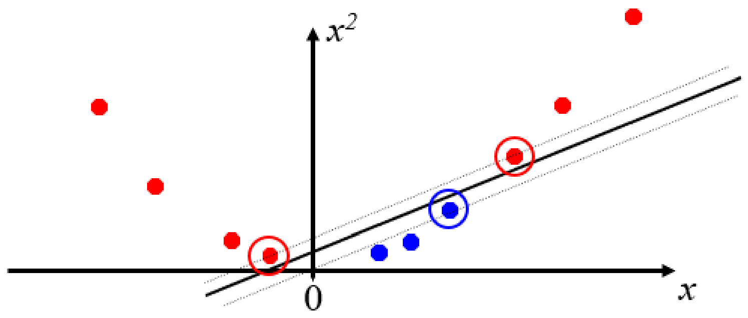
Scenario 2

- When the dataset is hard:



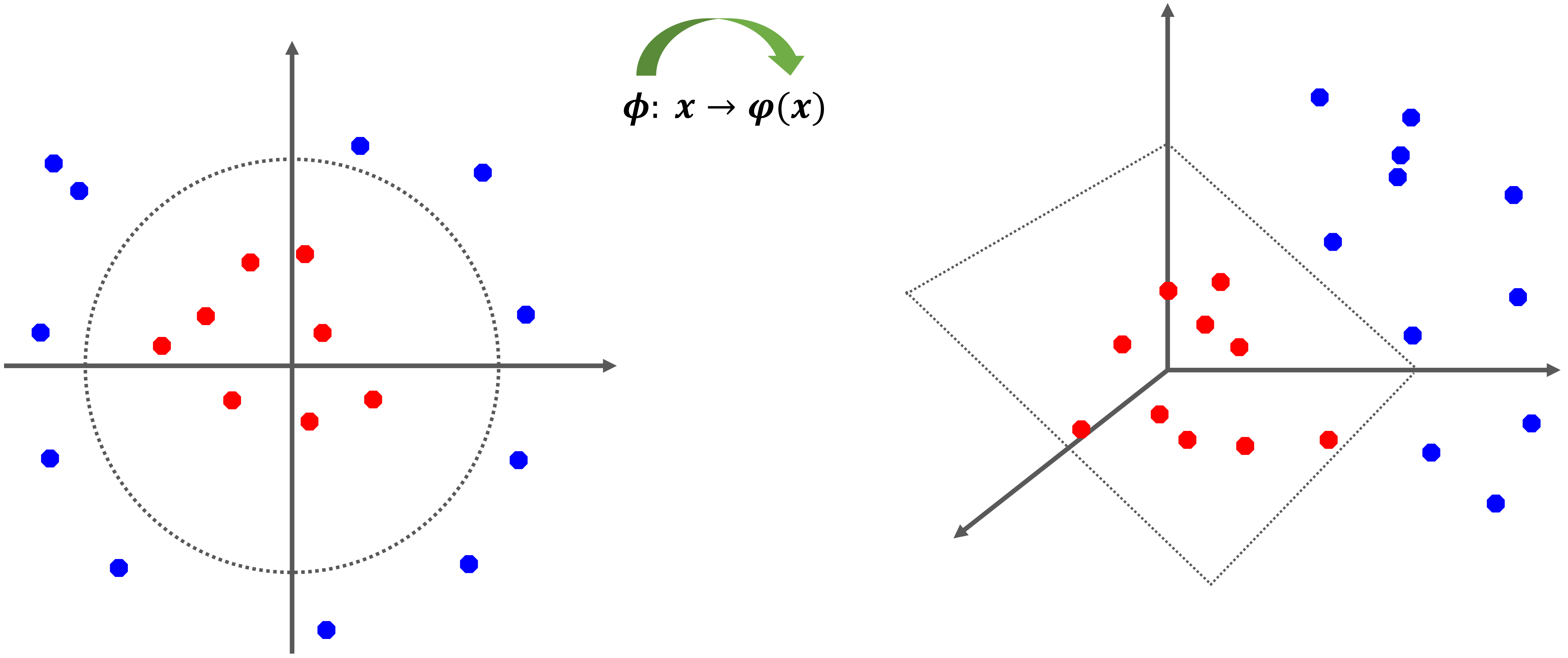
Scenario 3

- Mapping data to a higher dimensional space



Nonlinear SVMs: Feature Spaces

The original feature space can always be mapped to some higher-dimensional feature space where the training set is separable.



$$\phi: x \rightarrow \varphi(x)$$

The Kernel Trick

- The linear classifier relies on inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$
- If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \phi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\phi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{x}_j)$$

- A **kernel function** is a function that is equivalent to an inner product in a feature space.
- Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \;\; x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2$,

  Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \boldsymbol{\phi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{x}_j)$:

  $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

  $= [1 \;\; x_{i1}^2 \;\; \sqrt{2}\, x_{i1}x_{i2} \;\; x_{i2}^2 \;\; \sqrt{2}x_{i1} \;\; \sqrt{2}x_{i2}]^{\mathbf{T}} [1 \;\; x_{j1}^2 \;\; \sqrt{2}\, x_{j1}x_{j2} \;\; x_{j2}^2 \;\; \sqrt{2}x_{j1} \;\; \sqrt{2}x_{j2}] =$

  $= \boldsymbol{\phi}(\mathbf{x}_i)^{\mathbf{T}}\boldsymbol{\phi}(\mathbf{x}_j)$, where $\boldsymbol{\phi}(\mathbf{x}) = [1 \;\; x_1^2 \;\; \sqrt{2}\, x_1x_2 \;\; x_2^2 \;\; \sqrt{2}x_1 \;\; \sqrt{2}x_2]$

- Thus, a kernel function implicitly maps data to a high-dimensional space (without the need to compute each $\boldsymbol{\phi}(\mathbf{x})$ explicitly).

Assisted Practice

Support Vector Machines

Duration: 15 mins.

Problem Statement: Motion Studios is the largest radio production house in Europe. Its total revenue is $ 1B+. The company has launched a new reality show "The Star RJ." The show is about finding a new radio jockey who will be the star presenter on upcoming shows.
In the first round, participants have to upload their voice clip online. The clip will be evaluated by experts for selection to the next round. There is a separate team in the first round for evaluation of male and female voice.
Response to the show is unprecedented, and company is flooded with voice clips.
You “as an ML” expert have to classify the voice as either male or female so that the first level of filtration is quicker.

Objective: Optimize selection process.

Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Unassisted Practice

Support Vector Machines

Duration: 15 mins.

Problem Statement: Load the data from "college.csv" that has attributes collected about private and public colleges for a particular year. Predict the private/public status of the colleges from other attributes.
Use LabelEncoder to encode the target variable to numerical form. Split the data such that 20% of the data is set aside for testing. Fit a linear svm from scikit learn and observe the accuracy. [Hint: Use Linear SVC]
Preprocess the data using StandardScalar and fit the same model again. Observe the change in accuracy.
Use scikit learn's gridsearch to select the best hyperparameter for a nonlinear SVM. Identify the model with best score and its parameters. [Hint: Refer to model_selection module of Scikit learn]

Objective: Employ SVM from scikit learn for binary classification and measure the impact of preprocessing data and hyper parameter search using grid search.

Note: This practice is not graded. It is only intended for you to apply the knowledge you have gained to solve real-world problems.
Access: Click on the Labs tab on the left side panel of the LMS. Copy or note the username and password that are generated. Click on the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

Import the Dataset

Code

```
import pandas as pd
df = pd.read_csv("College.csv")
df.columns
```

```
Out[5]: Index(['Private', 'Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc',
               'F.Undergrad', 'P.Undergrad', 'Outstate', 'Room.Board', 'Books',
               'Personal', 'PhD', 'Terminal', 'S.F.Ratio', 'perc.alumni', 'Expend',
               'Grad.Rate'],
              dtype='object')
```

Label Encoding

Code

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
X, y = df.iloc[:, 1:].values, df.iloc[:, 0].values
# male -> 1
# female -> 0
target_encoder = LabelEncoder()
y = target_encoder.fit_transform(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
print(X_train.shape)
```

```
(621, 17)
```


simplilearn

Fit the Linear SVC Classifier

Code

```
from sklearn.svm import LinearSVC,SVC
classifier = LinearSVC()

classifier.fit(X_train,y_train)
y_predict = classifier.predict(X_test)
classifier.score(X_test,y_test)
```

```
Out[8]: 0.8333333333333337
```

Obtain Performance Matrix

Code

```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_predict,y_test))
```

```
[[39 26]
 [ 0 91]]
```

Fit the SVC Classifier

Code

```
classifier = SVC()
classifier.fit(X_train,y_train)
classifier.score(X_test,y_test)
```

```
Out[10]: 0.75
```

Preprocess the Data

Code

```
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X, y = df.iloc[:, 1:].values, df.iloc[:, 0].values
X = scaler.fit_transform(X)
target_encoder = LabelEncoder()
y = target_encoder.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=1)
print(X_train.shape)
```

```
(621, 17)
```

Refitting the SVC Model

Code

```
classifier = SVC()
classifier.fit(X_train,y_train)
classifier.score(X_test,y_test)
```

```
Out[14]: 0.9423076923076923 9
```

Fitting Grid Search

Code

```
import numpy as np
from sklearn.model_selection import StratifiedShuffleSplit
C_range = np.logspace(-2, 10, 13)
gamma_range = np.logspace(-9, 3, 13)
param_grid = dict( gamma=gamma_range,C=C_range)
grid = GridSearchCV(SVC(), param_grid=param_grid)
grid.fit(X_train, y_train)
```

```
Out[28]: GridSearchCV(cv=None, error_score='raise',
                estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False),
                fit_params=None, iid=True, n_jobs=1,
                param_grid={'gamma': array([  1.00000e-09,   1.00000e-08,   1.00000e-07,   1.00000e-06,
                  1.00000e-05,   1.00000e-04,   1.00000e-03,   1.00000e-02,
                  1.00000e-01,   1.00000e+00,   1.00000e+01,   1.00000e+02,
                  1.00000e+03]), 'C': array([  1.00000e-02,   1.00000e-01,   1.00000e+00,   1.00000e+01,
                  1.00000e+02,   1.00000e+03,   1.00000e+04,   1.00000e+05,
                  1.00000e+06,   1.00000e+07,   1.00000e+08,   1.00000e+09,
                  1.00000e+10])},
                pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
                scoring=None, verbose=0)
```

Getting the Best Hyperparameter

Code

```
print("The best parameters are %s with a score of %0.2f"
      % (grid.best_params_, grid.best_score_))
```

```
The best parameters are {'C': 10000000.0, 'gamma': 9.9999999999999995e-07} with a score of 0.94
```

Key Takeaways

Now, you are now able to:

- Understand classification as part of supervised learning
- Demonstrate different classification techniques in Python
- Evaluate classification models

Knowledge Check

Knowledge Check 1	**Let us train T1, a decision tree, with the data given below. Which feature will you split at the root?**

a. x1

b. x2

c. x3

d. y

x1	x2	x3	y
1	1	1	+1
0	1	0	-1
1	0	1	-1
0	0	1	+1

Knowledge Check

1

Let us train T1, a decision tree, with the data given below. Which feature will you split at the root?

a. **x1**

b. **x2**

c. **x3**

d. y

x1	x2	x3	y
1	1	1	+1
0	1	0	-1
1	0	1	-1
0	0	1	+1

The correct answer is **c. x3**

x3 will split because it has the lowest classification error. At row 3, x3=1, y=-1; there is only one error compared to other features.

Knowledge Check 2	**If you are training a decision tree, and you are at a node in which all of its data has the same y value, you should:**

a. Find the best feature to split

b. Create a leaf that predicts the y value of all the data

c. Terminate recursions on all branches and return the current tree

d. Go back to the parent node and select a different feature to split so that the y values are not all the same at this node

Knowledge Check 2	If you are training a decision tree, and you are at a node in which all of its data has the same y value, you should

a. Find the best feature to split

b. Create a leaf that predicts the y value of all the data

c. Terminate recursions on all branches and return the current tree

d. Go back to the parent node and select a different feature to split so that the y values are not all the same at this node

The correct answer is **b. Create a leaf that predicts the y value of all the data**

You should create a leaf that predicts the y value of all the data.

Lesson-End Project

Duration: 20 mins.

Problem Statement: Load the kinematics dataset as measured on mobile sensors from the file "run_or_walk.csv." List the columns in the dataset. Let the target variable "y" be the activity, and assign all the columns after it to "x."
Using Scikit-learn, fit a Gaussian Naive Bayes model and observe the accuracy. Generate a classification report using Scikit-learn. Repeat the model once using only the acceleration values as predictors and then using only the gyro values as predictors. Comment on the difference in accuracy between both the models.

Objective: Practice classification based on Naive Bayes algorithm. Identify the predictors that can be influential.

Access: Click the Labs tab in the left side panel of the LMS. Copy or note the username and password that are generated. Click the Launch Lab button. On the page that appears, enter the username and password in the respective fields, and click Login.

simplilearn

Thank You