دوره دیتا ساینس کاربردی

Machine Learning

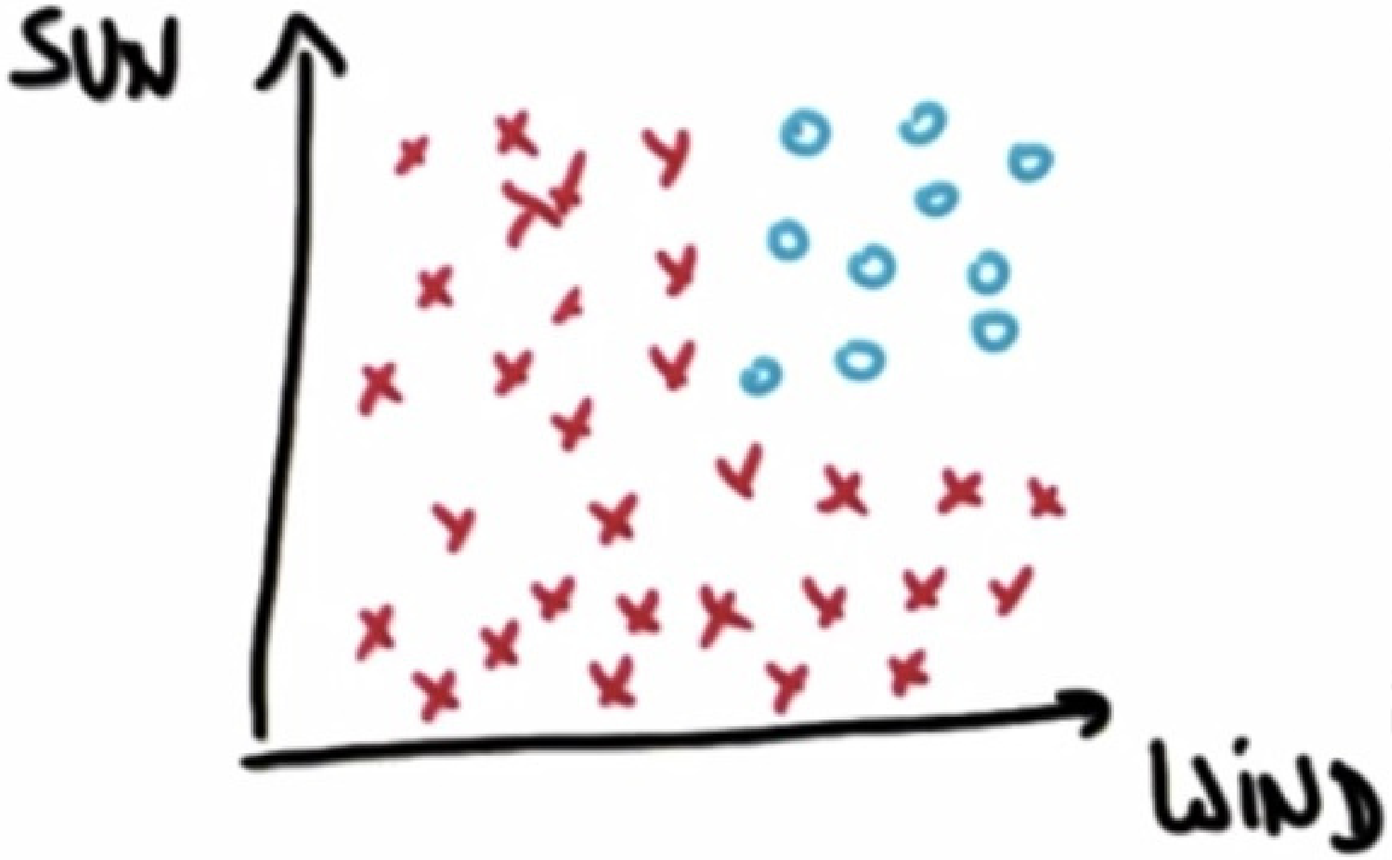
Decision Trees &Random Forest

dataroadmap

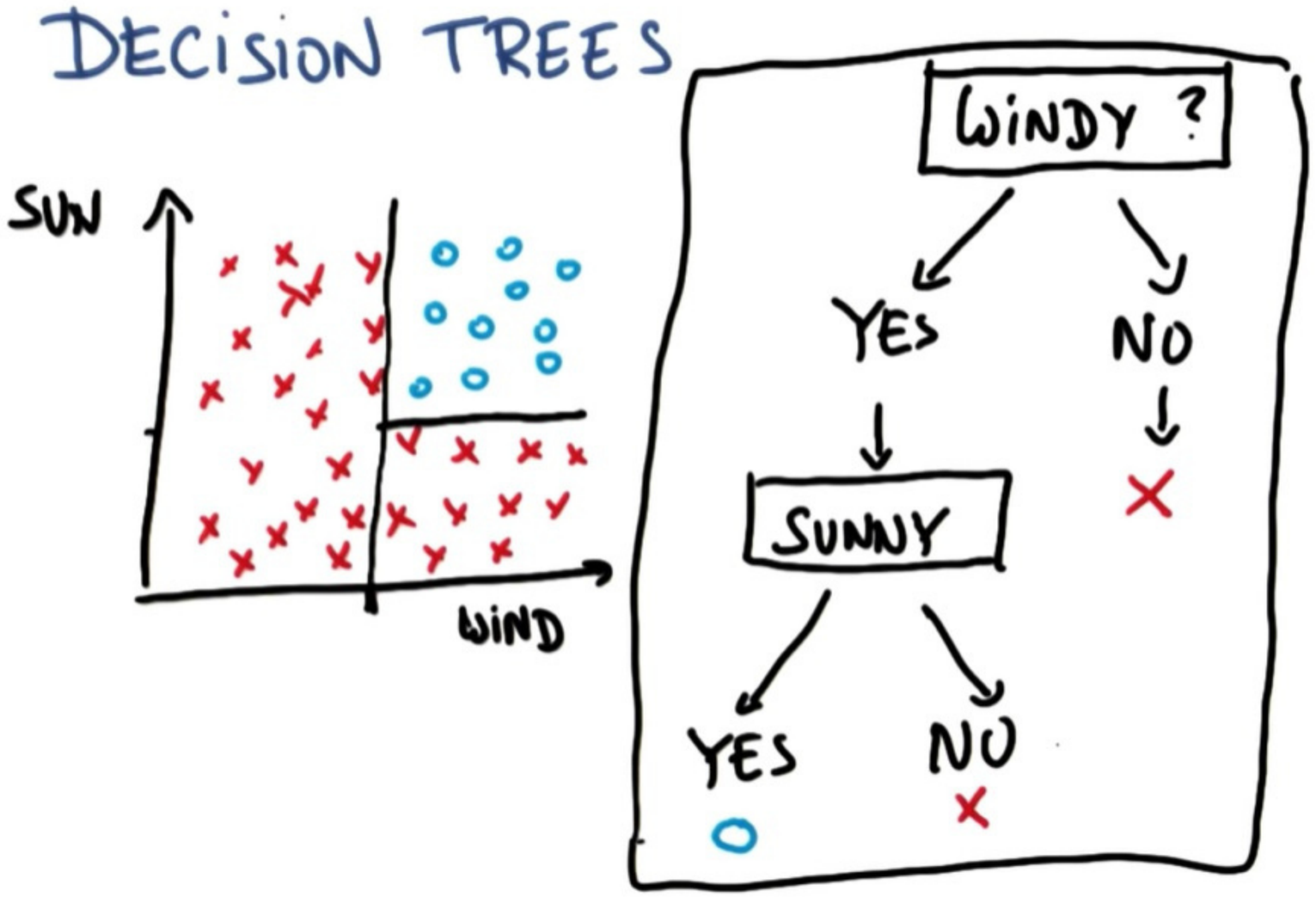
مدرس: مونا حاتمی

جلسه هشتم

Decision trees- Supervised Learning



Decision trees- Supervised Learning



Decision trees

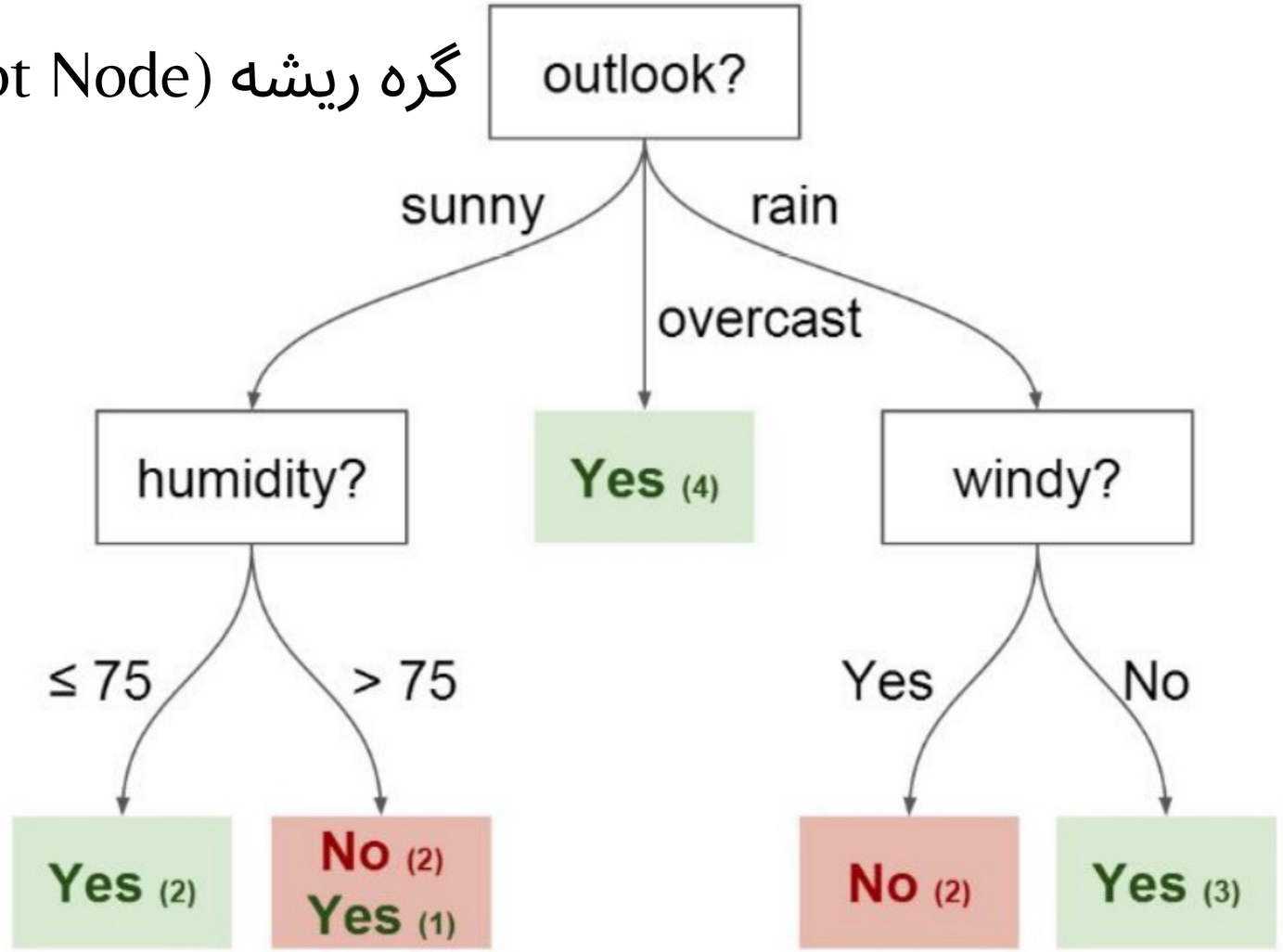
Temperature	Outlook	Humidity	Windy	Played?
Mild	Sunny	80	No	Yes
Hot	Sunny	75	Yes	**No**
Hot	Overcast	77	No	Yes
Cool	Rain	70	No	Yes
Cool	Overcast	72	Yes	Yes
Mild	Sunny	77	No	**No**
Cool	Sunny	70	No	Yes
Mild	Rain	69	No	Yes
Mild	Sunny	65	Yes	Yes
Mild	Overcast	77	Yes	Yes

Decision trees

گره (Node)

برگ(Leaf)

گره ریشه (Root Node)



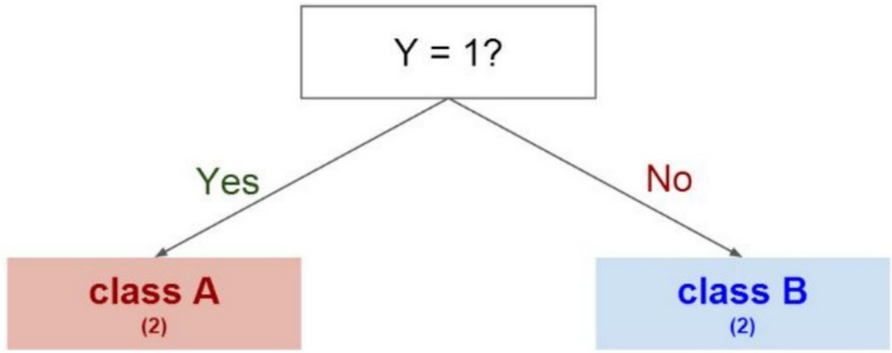
Decision trees

Split

X	Y	Z	Class
1	1	1	A
1	1	0	A
0	0	1	B
1	0	0	B

first split on

X		Y		Z	
X = 0	X = 1	Y = 0	Y = 1	Z = 0	Z = 1
B	A A B	B B	A A	A B	A B



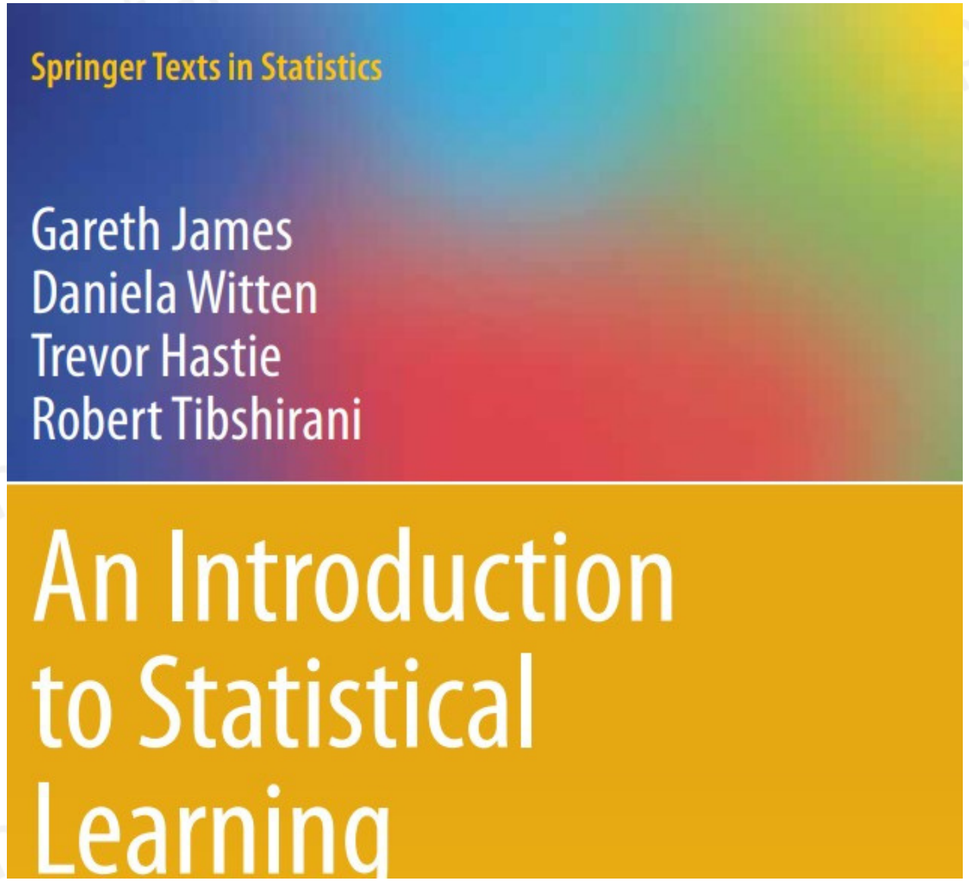
Decision trees- Equations

$$Entropy:$$

$$H(S) = -\sum_{i} p_i(S)\, log_2\, p_i(S)$$

$$Information\ Gain:$$

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



Decision trees

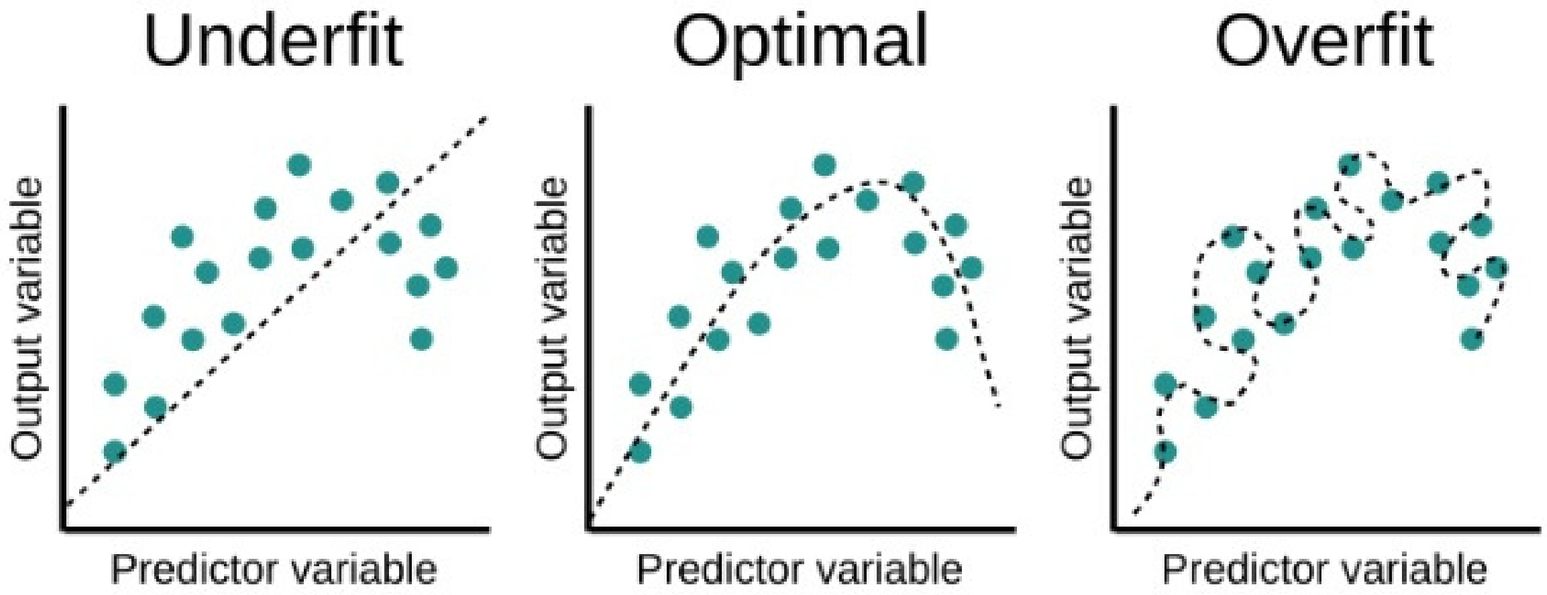
مزایا:

فهم آسان

معایب:

بیش از حد مناسب بودن(overfitting)

Overfitting



Read the Data

```
import pandas as pd
```

The Data ¶

```
df = pd.read_csv('preprocessed_dataset.csv')
```

```
df.head()
```

[3]:

	Unnamed: 0	PayloadMass	Flights	GridFins	Reused	Legs	Block	ReusedCou
0	0	6104.959412	1	0	0	0	1.0	
1	1	525.000000	1	0	0	0	1.0	
2	2	677.000000	1	0	0	0	1.0	

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 90 entries, 0 to 89
Data columns (total 89 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Unnamed: 0                              90 non-null     int64
 1   PayloadMass                             90 non-null     float64
 2   Flights                                 90 non-null     int64
```

```
X=df.drop('Class',axis=1)
y=df['Class']
```

Train set- Test set in sklearn library

Train Test Split

In [6]:
```
from sklearn.model_selection import train_test_split
```

In [7]:
```
X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y, test_size=0.3,
                                                    random_state=101)
```

Train & Prediction

```
from sklearn.tree import DecisionTreeClassifier
```

In [9]:
```
tree = DecisionTreeClassifier()
```

In [10]:
```
tree.fit(X_train,y_train)
```

Out[10]:
```
DecisionTreeClassifier()
```

In [11]:
```
predictions = tree.predict(X_test)
```

Prediction vs y_test

```
In [12]: predictions
```

```
Out[12]: array([0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
                1, 1, 0, 1, 1], dtype=int64)
```

```
In [13]: y_test
```

```
Out[13]: 50    0
         6     1
         51    0
         54    1
         53    1
```

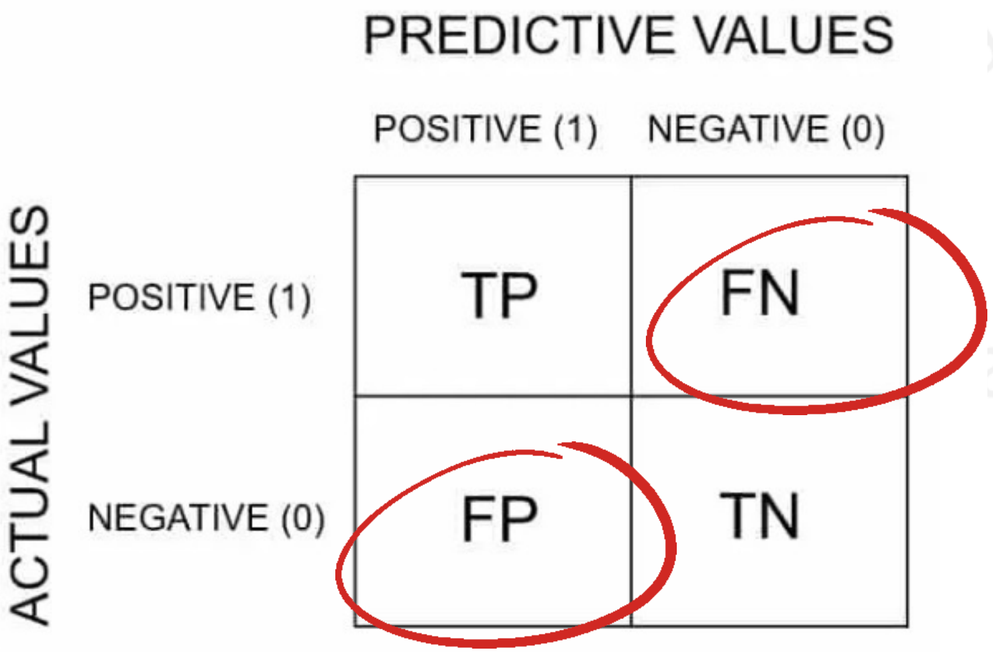
Evaluation

In [14]:
```
from sklearn.metrics import confusion_matrix
```

In [15]:
```
confusion_matrix(y_test,predictions)
```

Out[15]:
```
array([[ 9,  2],
       [ 1, 15]], dtype=int64)
```

Confusion Matrix Error I & Error II




```
([[ 9,  2],
  [ 1, 15]], dtype=int64)
```

Evaluation

In [16]:
```
from sklearn.metrics import accuracy_score
```

In [17]:
```
accuracy_score(y_test,predictions, normalize=False)
```

Out[17]: 24

In [18]:
```
accuracy_score(y_test,predictions, normalize=True)
```

Out[18]: 0.8888888888888888

Confusion Matrix Error I & Error II

Accuracy = (TP+TN)/(TP+FP+FN+TN) صحت

Precision = TP/(TP+FP) دقت

Recall = TP/(TP+FN) بازخوانی

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Classification Report

In [19]:
```
from sklearn.metrics import classification_report
```

In [20]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.90      0.82      0.86        11
           1       0.88      0.94      0.91        16

    accuracy                           0.89        27
   macro avg       0.89      0.88      0.88        27
weighted avg       0.89      0.89      0.89        27
```

Hyperparameters

```
In [22]: tree_1 = DecisionTreeClassifier()

In [23]: Init signature:
         DecisionTreeClassifier(
             *,
             criterion='gini',
             splitter='best',
             max_depth=None,
             min_samples_split=2,
             min_samples_leaf=1,
             min_weight_fraction_leaf=0.0,
             max_features=None,

In [24]:

Out[24]:
```

Grid Search-Decision Tree- Parameters

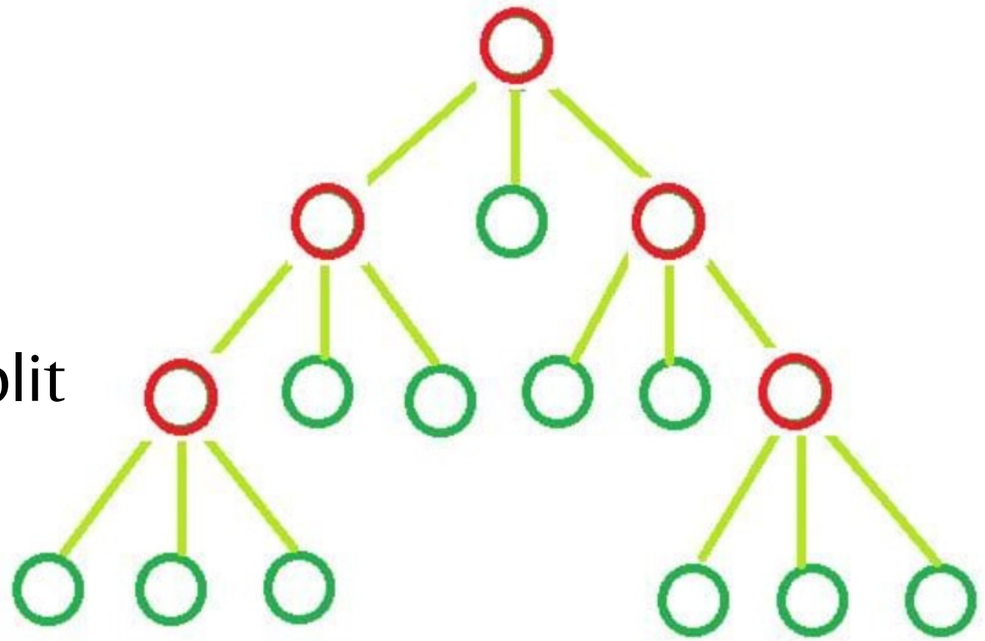
[21]:
```
# Allows us to test parameters of classification algorithms and find the best one
from sklearn.model_selection import GridSearchCV
```

[22]:
```
tree_1 = DecisionTreeClassifier()
```

[23]:
```
parameters = {'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5]}
```

min_samples_split

min_samples_leaf



Grid Serach- Best Hyperparameters

```
print("tuned hpyerparameters :(best parameters) ",tree_cv.best_params_)
```

```
tuned hpyerparameters :(best parameters)  {'min_samples_leaf': 1, 'min_samples_split': 5}
```

Tune the Best Hyperparameters

In [26]:
```
tree_1 = DecisionTreeClassifier(min_samples_leaf= 1, min_samples_split= 5)
```

In [27]:
```
tree_1.fit(X_train,y_train)
```

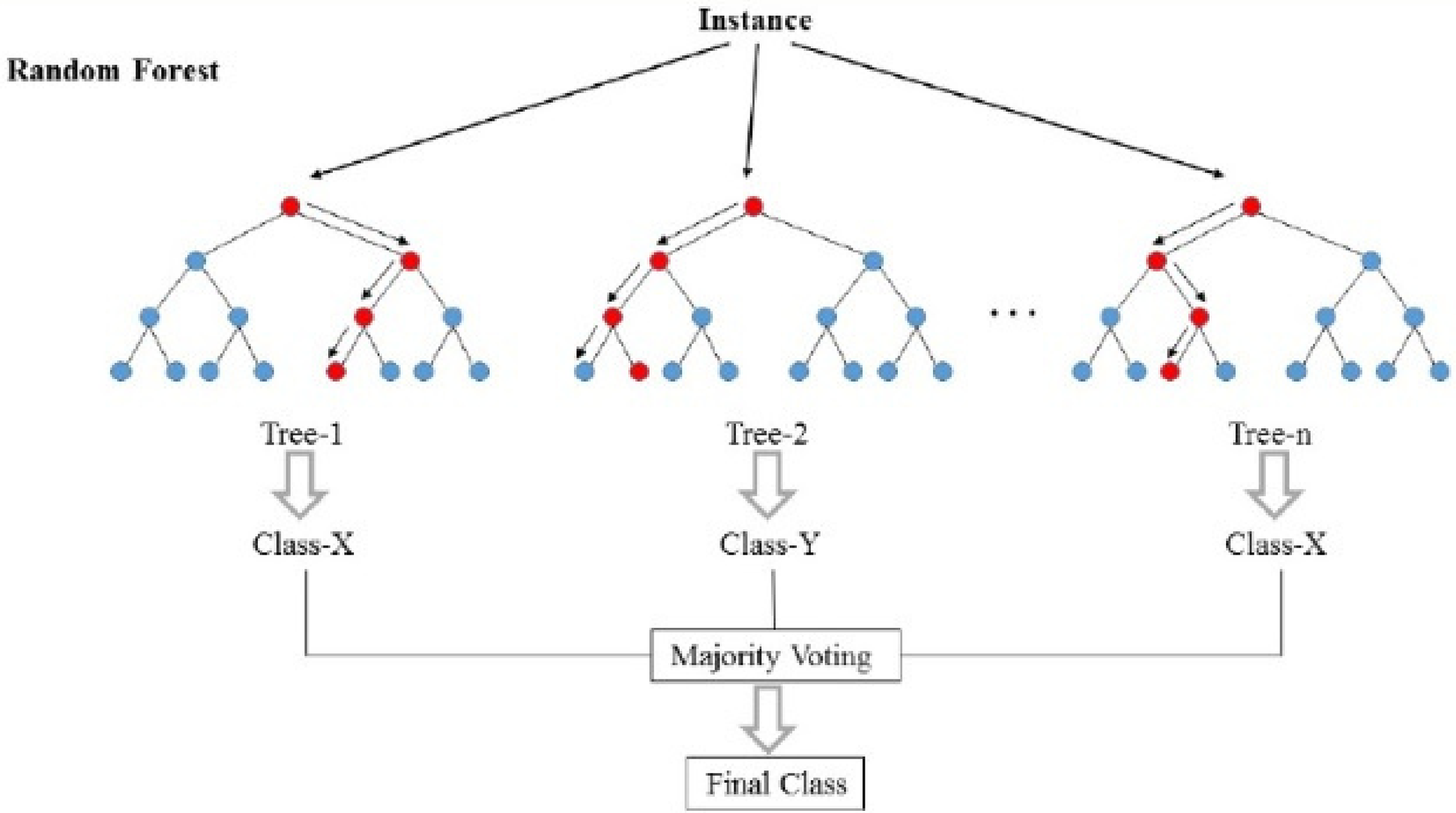
Out[27]:
```
DecisionTreeClassifier(min_samples_split=5)
```

In [28]:
```
predictions_1 = tree_1.predict(X_test)
```

Result

```
In [29]: confusion_matrix(y_test,predictions_1)

Out[29]: array([[ 9,  2],
                [ 1, 15]], dtype=int64)

In [30]: accuracy_score(y_test,predictions_1, normalize=False)

Out[30]: 24

In [31]: accuracy_score(y_test,predictions_1, normalize=True)

Out[31]: 0.888888888888888
```

Random Forest



Train & Prediction

In [32]:
```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train, y_train)
```

Out[32]:
```
RandomForestClassifier()
```

In [33]:
```
rfc_pred = rfc.predict(X_test)
```

Evaluation

In [34]:
```
print(confusion_matrix(y_test,rfc_pred))
```

```
[[ 8  3]
 [ 0 16]]
```

In [35]:
```
accuracy_score(y_test,predictions, normalize=False)
```

Out[35]: 24

In [36]:
```
accuracy_score(y_test,predictions, normalize=True)
```

Out[36]: 0.8888888888888888

Classification Report

In [37]:
```
print(classification_report(y_test,rfc_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.73      0.84        11
           1       0.84      1.00      0.91        16

    accuracy                           0.89        27
   macro avg       0.92      0.86      0.88        27
weighted avg       0.91      0.89      0.88        27
```

Grid Search

In [38]:
```
rfc_1 = RandomForestClassifier()
```

In [39]:
```
parameters = {'min_samples_leaf': [1, 2, 4],
     'min_samples_split': [2, 5, 10],  'n_estimators': [10,20,30] }
```

In [40]:
```
rfc_cv = GridSearchCV(rfc_1, parameters)
rfc_cv.fit(X_train, y_train)
```

Out[40]:
```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [10, 20, 30]})
```

Tune the Best Hyperparameters

```
print("tuned hpyerparameters :(best parameters) ",rfc_cv.best_params_)
```

```
tuned hpyerparameters :(best parameters)  {'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 10}
```

```
rfc_1 = RandomForestClassifier( n_estimators= 10, min_samples_leaf= 1, min_samples_split= 2)
```

Train & Predict

In [43]:
```
rfc_1.fit(X_train,y_train)
```

Out[43]:
```
RandomForestClassifier(n_estimators=10)
```

In [44]:
```
predictions_1 = rfc_1.predict(X_test)
```

In [45]:
```
confusion_matrix(y_test,predictions_1)
```

Out[45]:
```
array([[11,  0],
       [ 1, 15]], dtype=int64)
```

Result

In [46]:
```
accuracy_score(y_test,predictions_1, normalize=False)
```

Out[46]: 26

In [47]:
```
accuracy_score(y_test,predictions_1, normalize=True)
```

Out[47]: 0.962962962962962

Assignment:

تمرین:

کدهای ارائه شده در درس را در نوتبوک جدیدی انجام داده و در صورت نیاز از نوتبوک هفته هشتم استفاده کنید.

جلسه یازدهم دوره منتورینگ دیتاساینس در کانال یوتیوب را انجام دهید.

رزومه خود را آپدیت کرده و مهارتهایی که تا به امروز فراگرفته اید را اضافه کنید.