



## T8-Árbol de decisión



**Tecnológico  
de Monterrey**

Aarón Pérez Ontiveros – A01422524  
Miguel Ángel Muñoz Vázquez - A01423629  
ITESM CAMPUS CUERNAVACA

## Parte 1

### Introducción

Un árbol de decisión representa una función que toma como entrada un vector de valores de atributos y regresa una decisión como salida única. Los valores de entrada y salida pueden ser discretos o continuos. Un árbol de decisión alcanza su decisión mediante la realización de una secuencia de pruebas. Cada nodo interno corresponde a la prueba de un valor de los atributos de entrada  $A_i$  y las ramas del nodo están etiquetadas con los posibles valores del atributo,  $A_i = V_{ik}$ . Cada nodo hoja representa el valor retornado por la función. Para la construcción de estos árboles se usa la medida de la entropía.

### Árbol de decisión

Se implementó en Python el algoritmo de árbol de decisión usando recursión para construir el modelo.

```
def build_model(data, attributes, my_number):
    global node_number
    class_frecuencias = {}
    # class_entropy = 0
    for row in data:
        class_name = row[1]
        if class_name not in class_frecuencias:
            class_frecuencias[class_name]=1
        else:
            class_frecuencias[class_name]+=1
    data_len = len(data)
    if len(class_frecuencias) <= 1:
        return Node(next(iter(class_frecuencias)),[])
    # for k,v in class_frecuencias.items():
    #     Pi = v / data_len
    #     class_entropy -= Pi * log2(Pi)
    best_attribute = float('inf'), ''
    for attribute in attributes:
        sum_f = 0
        value_frecuencias, value_to_class_count = {}, {}
        for row in data:
            value = row[0][attribute]
            if value not in value_frecuencias:
                value_frecuencias[value]=1
            else:
                value_frecuencias[value]+=1
            if value not in value_to_class_count:
                value_to_class_count[value]={}
            value_class = row[1]
            if value_class not in value_to_class_count[value]:
```

```

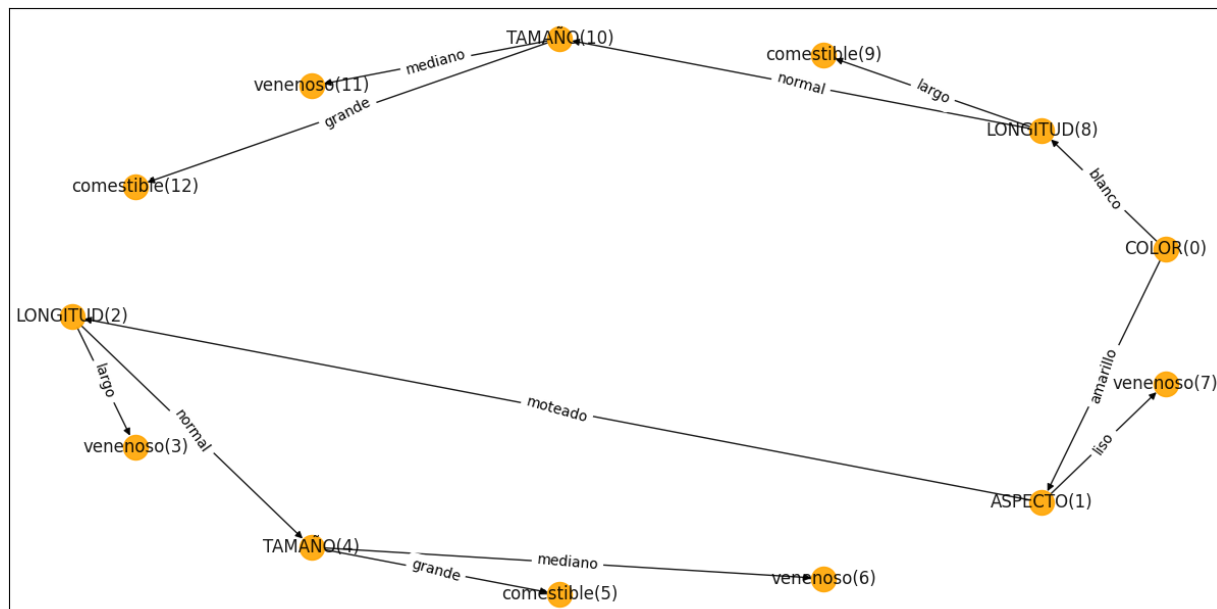
        value_to_class_count[value][value_class]=1
    else:
        value_to_class_count[value][value_class]+=1
    antigain = 0
    for k,f in value_frecuencias.items():
        value_entropy = 0
        for frequency in value_to_class_count[k].values():
            Pi = frequency / f
            value_entropy -= Pi * log2(Pi)
        #print('value',k,'frecuency',f,'value_entropy based in class',value_e
ntropy)

    antigain += f/data_len * value_entropy
    best_attribute = min((antigain,attribute),best_attribute)
    best_attribute_name = best_attribute[1]
    attributes_copy = deepcopy(attributes)
    attributes_copy.remove(best_attribute_name)
    dataSplitted = {}
    for row in data:
        bs_a_value = row[0][best_attribute_name]
        if bs_a_value not in dataSplitted:
            dataSplitted[bs_a_value]=[]
        row_copy = deepcopy(row)
        row_copy[0].pop(best_attribute_name)
        dataSplitted[bs_a_value].append(row_copy)

    node = Node(best_attribute_name,[])
    for value,data in dataSplitted.items():
        node_number+=1
        child_number = node_number
        child = build_model(data,deepcopy(attributes_copy),child_number)
        a = f'{node.attribute}({my_number})'
        b = f'{child.attribute}({child_number})'
        edges.append((a,b))
        edges_extended.append(((a,b),value))
        #print(edges)
        node.children.append((child,value))
    return node

```

Ésta es la función recursiva que calcula el mejor atributo y llama recursivamente a sus hijos para cada valor posible del atributo escogido. En el caso de que ya todos los posibles valores pertenezcan a una sola clase se regresa el nodo como hoja.



Como se puede observar, el árbol generado tiene como raíz el atributo COLOR.

## Conclusión

El árbol de decisión construido resultó tener 13 nodos en lugar de los 31 nodos que tendría si el árbol se construyera considerando absolutamente todas las posibilidades, logrando así una optimización del 58.06% con respecto a la solución exhaustiva.

Este árbol resultante puede ser utilizado para clasificar cualquier otra instancia en complejidad  $O(1)$ , debido a que en el peor caso se tienen que bajar 4 niveles para retornar la clase que corresponda.

## Bibliografía

Russell, S. and Norvig, P., 2016. *Artificial Intelligence: A Modern Approach*. 3rd ed. Edinburgh Gate: Pearson Education Limited.

[https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

## Parte 2

### Introducción

Existen diferentes tipos de algoritmos útiles para la clasificación de datos. Cada uno de ellos tiene una implementación distinta, pero el objetivo siempre es aproximar la clasificación de las instancias de pruebas a las instancias que se usaron para el modelo de entrenamiento. Las pruebas que se muestran a continuación son con el conjunto de datos de Weka, *Iris.arff*.

Los algoritmos usados en esta comparación de algoritmos son.

**Naïve Bayes:** Es un clasificador probabilístico fundamentado en el teorema de Bayes y el aprendizaje es supervisado.

**J48:** Es un clasificador estadístico que genera un árbol de decisión.

**Random Forest:** También llamado *random decision forests* es un método de clasificación que consiste en construir varios árboles de decisión durante el tiempo de entrenamiento y generando de salida la clase que es la moda de las clases. Los *random forests* arreglan el problema de *overfitting* que presentan los árboles de decisión. Generalmente los *random forests* son mejores que los árboles de decisión.

### Iris con 30 % de datos para entrenamiento

- Naïve Bayes 30 %

Classifier

Choose

NaiveBayes

Test options

☐ Use training set
 ☐ Supplied test set
 

Set...

☐ Cross-validation
 Folds 

10

☒ Percentage split
 % 

30

More options...

(Nom) class

Start

Stop

Result list (right-click for options)

10.47.35 - trees.J48

10.54.04 - bayes.NaiveBayes

10.57.11 - bayes.NaiveBayes

Classifier output

Naive Bayes Classifier

Attribute	Class		
	Iris-setosa (0.33)	Iris-versicolor (0.33)	Iris-virginica (0.33)
=====			
sepal.length			
mean	4.9913	5.9379	6.5795
std. dev.	0.355	0.5042	0.6353
weight sum	50	50	50
precision	0.1059	0.1059	0.1059
-----			
sepal.width			
mean	3.4015	2.7687	2.9629
std. dev.	0.3925	0.3038	0.3088
weight sum	50	50	50
precision	0.1091	0.1091	0.1091
-----			
petal.length			
mean	1.4694	4.2452	5.5516
std. dev.	0.1782	0.4712	0.5529
weight sum	50	50	50
precision	0.1405	0.1405	0.1405
-----			
petal.width			
mean	0.2743	1.3097	2.0343
std. dev.	0.1096	0.1915	0.2646
weight sum	50	50	50
precision	0.1143	0.1143	0.1143
-----			

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances	97	92.381 %
Incorrectly Classified Instances	8	7.619 %
Kappa statistic	0.886	
Mean absolute error	0.0578	
Root mean squared error	0.2072	
Relative absolute error	12.8992 %	
Root relative squared error	43.2862 %	
-----		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PBC Area	Class
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-setosa
	0.969	0.096	0.816	0.969	0.886	0.836	0.971	0.907	Iris-versicolor
	0.821	0.015	0.970	0.821	0.889	0.838	0.974	0.967	Iris-virginica
Weighted Avg.	0.924	0.035	0.933	0.924	0.924	0.890	0.981	0.960	

=== Confusion Matrix ===

a	b	c	<-- Classified as
34	0	0	a = Iris-setosa
0	31	1	b = Iris-versicolor
0	7	32	c = Iris-virginica

- J48 30%

**Classifier**

Choose **J48 - C 0.25 - M 2**

**Test options**

☐ Use training set  
☐ Supplied test set   
☐ Cross-validation Folds 10  
☒ Percentage split % 30

(Nom) class

**Result list (right-click for options)**

- 10:47:35 - trees.J48
- 10:54:04 - bayes.NaiveBayes
- 10:57:11 - bayes.NaiveBayes
- 10:59:17 - trees.J48

**Classifier output**

```

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (48.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves : 5
Size of the tree : 9

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances 100 95.2381 %
Incorrectly Classified Instances 5 4.7619 %
Kappa statistic 0.9287
Mean absolute error 0.0497
Root mean squared error 0.1823
Relative absolute error 11.1045 %
Root relative squared error 38.0667 %
Total Number of Instances 105

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-setosa
1.000 0.068 0.865 1.000 0.928 0.898 0.966 0.965 Iris-versicolor
0.872 0.000 1.000 0.872 0.932 0.900 0.936 0.919 Iris-virginica
Weighted Avg. 0.952 0.021 0.959 0.952 0.952 0.932 0.966 0.925

=== Confusion Matrix ===

a b c <-- classified as
34 0 0 | a = Iris-setosa
0 32 0 | b = Iris-versicolor
0 5 34 | c = Iris-virginica

```

- **Random Forest 30 %**

**Classifier**

Choose **RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1**

**Test options**

☐ Use training set  
☐ Supplied test set   
☐ Cross-validation Folds 10  
☒ Percentage split % 30

(Nom) class

**Result list (right-click for options)**

- 10:47:35 - trees.J48
- 10:54:04 - bayes.NaiveBayes
- 10:57:11 - bayes.NaiveBayes
- 10:59:17 - trees.J48
- 11:00:23 - trees.J48
- 11:01:15 - bayes.NaiveBayes
- 11:05:42 - trees.RandomForest

**Classifier output**

```

sepalwidth
petalwidth
petalwidth
class

Test mode: split 30.0% train, remainder test

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.06 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

=== Summary ===

Correctly Classified Instances 100 95.2381 %
Incorrectly Classified Instances 5 4.7619 %
Kappa statistic 0.9287
Mean absolute error 0.053
Root mean squared error 0.1768
Relative absolute error 11.8228 %
Root relative squared error 36.9065 %
Total Number of Instances 105

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-setosa
1.000 0.068 0.865 1.000 0.928 0.898 0.966 0.934 Iris-versicolor
0.872 0.000 1.000 0.872 0.932 0.900 0.968 0.968 Iris-virginica
Weighted Avg. 0.952 0.021 0.959 0.952 0.952 0.932 0.964 0.968

=== Confusion Matrix ===

a b c <-- classified as
34 0 0 | a = Iris-setosa
0 32 0 | b = Iris-versicolor
0 5 34 | c = Iris-virginica

```

Con el 30% de los datos de entrenamiento y con los tres diferentes algoritmos podemos observar que Los algoritmos J48 y Random Forest obtienen el mismo resultado, con 105 instancias para pruebas logran acertar con 100 instancias. Naïve Bayes acertó solo con 97.

## Iris con 66 % de datos para entrenamiento

- Naïve Bayes 66%

**Classifier**

Choose **NaiveBayes**

**Test options**

☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds 10  
☒ Percentage split % 66  
 More options...

(Nom) class

Start Stop

**Classifier output**

```

petallength
  mean      1.4694      4.2452      5.5516
  std. dev.  0.1782      0.4712      0.5529
  weight sum      50         50         50
  precision   0.1405      0.1405      0.1405

petalwidth
  mean      0.2743      1.3097      2.0343
  std. dev.  0.1096      0.1915      0.2646
  weight sum      50         50         50
  precision   0.1143      0.1143      0.1143

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

=== Summary ===

Correctly Classified Instances      48      94.1176 %
Incorrectly Classified Instances      3      5.8824 %
Kappa statistic      0.9113
Mean absolute error      0.0447
Root mean squared error      0.1722
Relative absolute error      10.0365 %
Root relative squared error      36.4196 %
Total Number of Instances      51

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
      1.000    0.000    1.000     1.000    1.000     1.000    1.000    1.000    Iris-setosa
      0.947    0.063    0.900     0.947    0.923     0.876    0.988    0.980    Iris-versicolor
      0.882    0.029    0.938     0.882    0.909     0.867    0.988    0.980    Iris-virginica
Weighted Avg.   0.941    0.033    0.942     0.941    0.941     0.909    0.992    0.986

=== Confusion Matrix ===

 a  b  c  <-- classified as
15  0  0 | a = Iris-setosa
 0 18  1 | b = Iris-versicolor
 0  2 15 | c = Iris-virginica
  
```

**Result list (right-click for options)**

12:52:39 - bayes NaiveBayes

- J48 66%



**Classifier**

Choose **J48 -C 0.25 -M 2**

**Test options**

☐ Use training set  
☐ Supplied test set   
☐ Cross-validation Folds 10  
☒ Percentage split % 66

(Nom) class

**Result list (right-click for options)**

- 12:52:39 - bayes.NaiveBayes
- 12:53:12 - trees.J48

**Classifier output**

```

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (46.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| | petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves : 5
Size of the tree : 9

Time taken to build model: 0.01 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances 49 96.0784 %
Incorrectly Classified Instances 2 3.9216 %
Kappa statistic 0.9408
Mean absolute error 0.0396
Root mean squared error 0.1579
Relative absolute error 8.8979 %
Root relative squared error 33.4091 %
Total Number of Instances 51

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area FRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-setosa
1.000 0.063 0.905 1.000 0.950 0.921 0.969 0.905 Iris-versicolor
0.882 0.000 1.000 0.882 0.938 0.913 0.967 0.938 Iris-virginica
Weighted Avg. 0.961 0.023 0.965 0.961 0.961 0.942 0.977 0.944

=== Confusion Matrix ===

a b c <-- classified as
15 0 0 | a = Iris-setosa
0 19 0 | b = Iris-versicolor
0 2 15 | c = Iris-virginica

```

- Random Forest 66 %

**Classifier**

Choose **RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1**

**Test options**

☐ Use training set  
☐ Supplied test set   
☐ Cross-validation Folds 10  
☒ Percentage split % 66

(Nom) class

**Result list (right-click for options)**

- 10:47:35 - trees.J48
- 10:54:04 - bayes.NaiveBayes
- 10:57:11 - bayes.NaiveBayes
- 10:59:17 - trees.J48
- 11:00:23 - trees.J48
- 11:01:15 - bayes.NaiveBayes
- 11:05:42 - trees.RandomForest
- 11:06:21 - trees.RandomForest

**Classifier output**

```

sepalwidth
petallength
petalwidth
class

Test mode: split 66.0% train, remainder test

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.02 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances 49 96.0784 %
Incorrectly Classified Instances 2 3.9216 %
Kappa statistic 0.9408
Mean absolute error 0.0349
Root mean squared error 0.1432
Relative absolute error 7.8349 %
Root relative squared error 30.2995 %
Total Number of Instances 51

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure MCC ROC Area FRC Area Class
1.000 0.000 1.000 1.000 1.000 1.000 1.000 1.000 Iris-setosa
1.000 0.063 0.905 1.000 0.950 0.921 0.994 0.990 Iris-versicolor
0.882 0.000 1.000 0.882 0.938 0.913 0.994 0.987 Iris-virginica
Weighted Avg. 0.961 0.023 0.965 0.961 0.961 0.942 0.996 0.992

=== Confusion Matrix ===

a b c <-- classified as
15 0 0 | a = Iris-setosa
0 19 0 | b = Iris-versicolor
0 2 15 | c = Iris-virginica

```

Con el 66% de los datos de entramiento y con los tres diferentes algoritmos podemos observar que Los algoritmos J48 y Random Forest obtienen el mismo resultado. Del conjunto de datos restan 51

instancias y las pruebas arrojan que 49 instancias fueron clasificadas de forma correcta y 2 de forma incorrecta. Naive Bayes acertó solo con 48.

## Iris con 80 % de datos para entrenamiento

- Naïve Bayes 80 %

Classifier

Choose NaiveBayes

Test options

☐ Use training set  
☐ Supplied test set   
☐ Cross-validation Folds 10  
☒ Percentage split % 80

(Nom) class

Result list (right-click for options)

10:47:35 - trees.J48  
10:54:04 - bayes.NaiveBayes  
10:57:11 - bayes.NaiveBayes  
10:59:17 - trees.J48  
11:00:23 - trees.J48  
11:01:15 - bayes.NaiveBayes

Classifier output

Attribute	Class		
	Iris-setosa (0.33)	Iris-versicolor (0.33)	Iris-virginica (0.33)
=====			
sepal.length			
mean	4.9913	5.9379	6.5795
std. dev.	0.355	0.5042	0.6353
weight sum	50	50	50
precision	0.1059	0.1059	0.1059
sepal.width			
mean	3.4015	2.7687	2.9629
std. dev.	0.3925	0.3038	0.3088
weight sum	50	50	50
precision	0.1091	0.1091	0.1091
petal.length			
mean	1.4694	4.2452	5.5516
std. dev.	0.1782	0.4712	0.5529
weight sum	50	50	50
precision	0.1405	0.1405	0.1405
petal.width			
mean	0.2743	1.3097	2.0343
std. dev.	0.1096	0.1915	0.2646
weight sum	50	50	50
precision	0.1143	0.1143	0.1143

Time taken to build model: 0 seconds  
  
=== Evaluation on test split ===  
  
Time taken to test model on test split: 0 seconds  
  
=== Summary ===  
  

Correctly Classified Instances	29	96.6667 %
Incorrectly Classified Instances	1	3.3333 %
Kappa statistic	0.9497	
Mean absolute error	0.0304	
Root mean squared error	0.1226	
Relative absolute error	6.8425 %	
Root relative squared error	25.9804 %	
Total Number of Instances	30	

=== Detailed Accuracy By Class ===  
  

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	1.000	0.000	1.000	1.000	1.000	1.000	1.000	1.000	Iris-setosa
	1.000	0.050	0.909	1.000	0.952	0.929	0.995	0.991	Iris-versicolor
	0.889	0.000	1.000	0.889	0.941	0.921	0.995	0.989	Iris-virginica
Weighted Avg.	0.967	0.017	0.970	0.967	0.966	0.953	0.997	0.994	

=== Confusion Matrix ===  
  

```

a b c <-- classified as
11 0 0 | a = Iris-setosa
0 10 0 | b = Iris-versicolor
0 1 8 | c = Iris-virginica

```

- J48 80 %

**Classifier**

Choose **J48 -C 0.25-M 2**

**Test options**

☐ Use training set  
☐ Supplied test set   
☐ Cross-validation Folds 10  
☒ Percentage split % 80

(Nom) class

**Result list (right-click for options)**

- 10:47:35 - trees.J48
- 10:54:04 - bayes.NaiveBayes
- 10:57:11 - bayes.NaiveBayes
- 10:59:17 - trees.J48
- 11:00:23 - trees.J48**

**Classifier output**

```

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
| petalwidth <= 1.7
| | petallength <= 4.9: Iris-versicolor (46.0/1.0)
| | petallength > 4.9
| | | petalwidth <= 1.5: Iris-virginica (3.0)
| | | petalwidth > 1.5: Iris-versicolor (3.0/1.0)
| petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves :    5

Size of the tree :    9

Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances      30           100 %
Incorrectly Classified Instances    0             0 %
Kappa statistic                     1
Mean absolute error                 0.0105
Root mean squared error            0.0166
Relative absolute error             2.3665 %
Root relative squared error        3.5274 %
Total Number of Instances         30

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
          1.000    0.000    1.000     1.000    1.000     1.000    1.000    1.000    Iris-setosa
          1.000    0.000    1.000     1.000    1.000     1.000    1.000    1.000    Iris-versicolor
          1.000    0.000    1.000     1.000    1.000     1.000    1.000    1.000    Iris-virginica
Weighted Avg.   1.000    0.000    1.000     1.000    1.000     1.000    1.000    1.000

=== Confusion Matrix ===
  a  b  c  <-- classified as
11 0 0 | a = Iris-setosa
 0 10 0 | b = Iris-versicolor
 0 0 9 | c = Iris-virginica

```

- Random Forest 80 %

**Classifier**

Choose **RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1**

**Test options**

☐ Use training set  
☐ Supplied test set   
☐ Cross-validation Folds 10  
☒ Percentage split % 80

(Nom) class

**Result list (right-click for options)**

- 10:47:35 - trees.J48
- 10:54:04 - bayes.NaiveBayes
- 10:57:11 - bayes.NaiveBayes
- 10:59:17 - trees.J48
- 11:00:23 - trees.J48
- 11:01:15 - bayes.NaiveBayes
- 11:05:42 - trees.RandomForest
- 11:06:21 - trees.RandomForest
- 11:06:57 - trees.RandomForest**

**Classifier output**

```

sepalwidth
petallength
petalwidth
class

Test mode: split 80.0% train, remainder test

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-check-capabilities

Time taken to build model: 0.03 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances      29           96.6667 %
Incorrectly Classified Instances    1             3.3333 %
Kappa statistic                    0.9497
Mean absolute error                0.0304
Root mean squared error            0.1132
Relative absolute error             6.8444 %
Root relative squared error        24.001 %
Total Number of Instances         30

=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
          1.000    0.000    1.000     1.000    1.000     1.000    1.000    1.000    Iris-setosa
          1.000    0.050    0.909     1.000    0.952     0.929    1.000    1.000    Iris-versicolor
          0.889    0.000    1.000     0.889    0.941     0.921    1.000    1.000    Iris-virginica
Weighted Avg.   0.967    0.017    0.970     0.967    0.966     0.953    1.000    1.000

=== Confusion Matrix ===
  a  b  c  <-- classified as
11 0 0 | a = Iris-setosa
 0 10 0 | b = Iris-versicolor
 0 1 8 | c = Iris-virginica

```

Con el 80% de los datos de entrenamiento y con los tres diferentes algoritmos podemos observar una diferencia con respecto a los anteriores. La cantidad de instancias para pruebas son solamente 30. La mejor precisión fue con el algoritmo J48, ya que pudo clasificar de forma correcta todas las instancias de prueba. Después, con los algoritmos Random Forest y Naive Bayes se tuvieron 29 clasificaciones correctas y una incorrecta.

## Conclusión

En general, los tres algoritmos tuvieron un buen grado de clasificación en todos los casos. Pero podemos destacar que el algoritmo con mejores resultados es el J48. El algoritmo J48, a diferencia de los otros dos, siempre se mantuvo con el mejor porcentaje de clasificación. Probablemente el J48 sea mejor debido a que no son muchos atributos y porque el dominio de cada atributo no es tan grande.

## Bibliografía

Russell, S. and Norvig, P., 2016. *Artificial Intelligence: A Modern Approach*. 3rd ed. Edinburgh Gate: Pearson Education Limited.

[https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)