Data Mining and Machine Learning

Arboles de Decision y Random Forest

Dr. Edgar Acuna
Department of Mathematics

Universidad de Puerto Rico- Mayaguez

academic.uprm.edu/eacuna

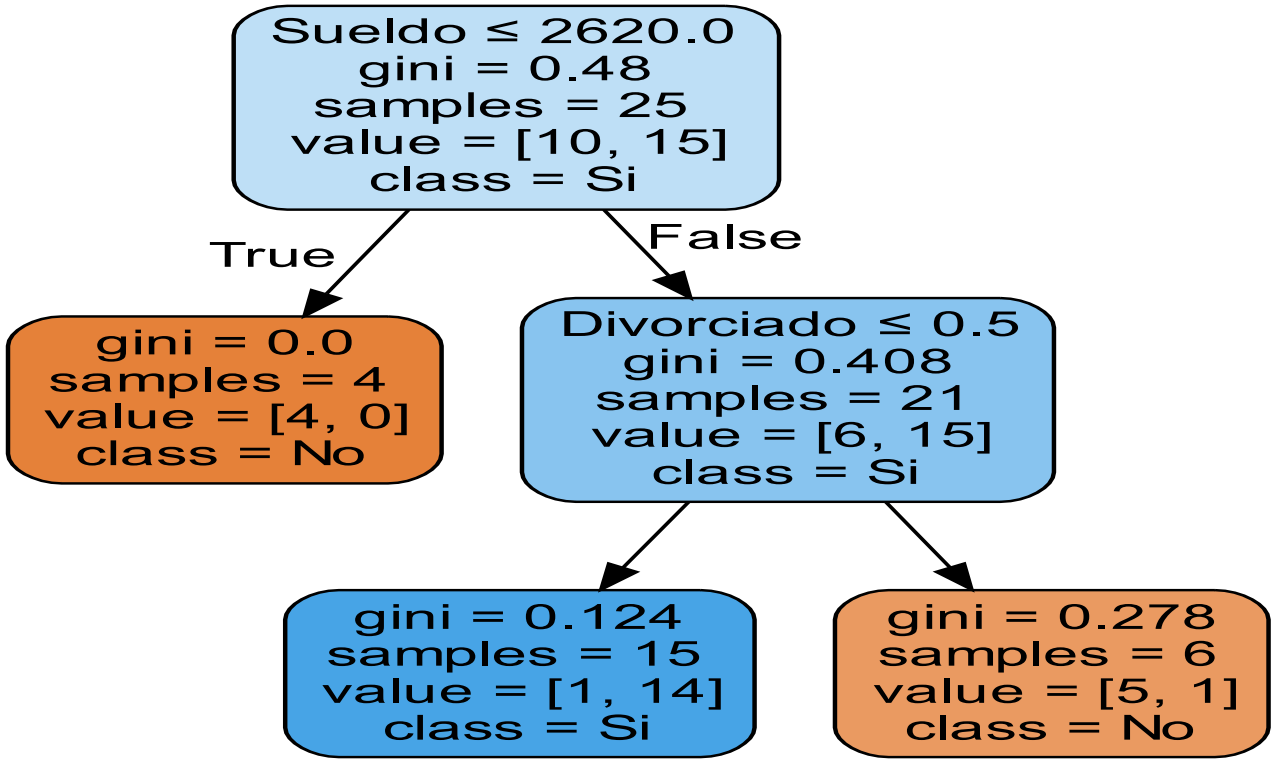
El uso de árboles de decisión tuvo su origen en las ciencias sociales con los trabajos de Sonquist y Morgan (1964) y Morgan y Messenger (1979) realizado en el Survey Research Center del Institute for Social Research de la Universidad de Michigan. El programa THAID (Theta Automatic Interaction Detection), de Sonquist, Baker y Morgan (1971), fue uno de los primeros métodos de ajuste de los datos basados en árboles de clasificación.

En estadística, Kass (1980) introdujo un algoritmo recursivo de clasificación no binario, llamado CHAID (Chi-square automatic interaction detection). Más tarde, Breiman, Friedman, Olshen y Stone (1984) introdujeron un nuevo algoritmo para la construcción de arboles y los aplicaron a problemas de regresión y clasificación. El método es conocido como CART (Classification and regression trees) por sus siglas en inglés. Casi al mismo tiempo el proceso de indución mediante árboles de decisión comenzó a ser usado por la comunidad de "Machine Learning" (Michalski, (1973), Quinlan (1983)). En el área de "Pattern Recognition" , Henrichon y Fu (1969) escribieron un paper en particionamiento noparametrico que guarda cierta relacion con arboles de decisión, pero usa hiperplanos que no son paralelos a los ejes coordenados.

Ejemplo: Prestamo para comprar carro

Sexo	Familia	CasPropia	AnosEmpleo	Sueldo	StatustMarital	Prestamo
Hombre	3	No	17	2500	Soltero	No
Mujer	5	Si	10	3000	Casado	Si
Mujer	4	No	15	2000	Viudo	No
Hombre	3	Si	16	2800	Soltero	Si
Hombre	6	Si	11	4000	Viudo	Si
Mujer	4	Si	26	3200	Soltero	Si
Mujer	2	Si	14	1800	Soltero	No
Hombre	5	Si	10	3750	Casado	Si
Hombre	6	No	18	2970	Divorciado	No
Hombre	4	Si	12	3350	Divorciado	No
Hombre	1	No	23	1950	Soltero	No
Mujer	2	Si	25	2740	Soltero	Si
Mujer	3	No	7	3100	Soltero	Si
Hombre	5	Si	5	3845	Divorciado	No
Hombre	3	No	13	3200	Casado	Si
Mujer	3	Si	9	2800	Soltero	No
Hombre	2	No	6	3200	Soltero	Si
Hombre	3	Si	7	3815	Viudo	Si
Mujer	2	Si	11	2980	Divorciado	No
Hombre	4	Si	15	2850	Viudo	Si
Mujer	1	No	6	3125	Divorciado	No
Hombre	1	No	8	3500	Soltero	Si
Hombre	4	No	22	4500	Divorciado	Si
Hombre	2	Si	10	3200	Casado	Si
Hombre	3	Si	9	3000	Casado	Si

Example: Predicting a bank decision to offer a loan to a customer for buying a car



Notice that the split on the categorical variable is not understandable. This is due to the fact that sciki-learn works only with numerical attributes

Algoritmos para arboles de Decisión

C4.5. Introducido por Quinlan (1993) dentro de la comunidad de "Machine Learning". Es decendiente del ID3 (Quinlan, 1986).

CHAID. Significa "Chi-square automatic interaction detection", fue introducido por Kass (1980) y es un derivado del THAID: "A sequential search program for the analysis of nominal scale dependent variables" (Morgan and Messenger, 1973). El criterio para particionar está basado en $\chi 2$.

NewId. (Boswell, 1990). Es descendiente del ID3 (Quinlan, 1986)

CART. Introducido por Breiman et al. (1984), propiamente es un algoritmo de árboles de decisión binario. Existe una versión similar llamada IndCART y que está disponible en el paquete IND distribuido por la NASA. RPART (PARTicionamiento Recursivo), una versión de CART esta disponible en R.

Arboles Bayesianos: Está basado en aplicación de métodos Bayesianos a arboles de decisión. Buntine (1992). Disponible en el paquete IND distribuido por la NASA.

CN2. Introducido por Clark and Niblett (1989).

Construcción de arboles de decisión

Un árbol de decisión particiona el espacio de variables predictoras en un conjunto de hiper-rectángulos y en cada uno de ellos ajusta un modelo sencillo, generalmente una constante. Es decir y=c, donde y es la variable de respuesta.

La construcción de un árbol de decisión se basa en cuatro elementos:

a) Un conjunto de preguntas binarias Q de la forma { $x \in A$?} donde A es un subconjunto del espacio muestral de la variable x.

b) El método usado para particionar los nodos.

c) La estrategia requerida para parar el crecimiento del árbol.

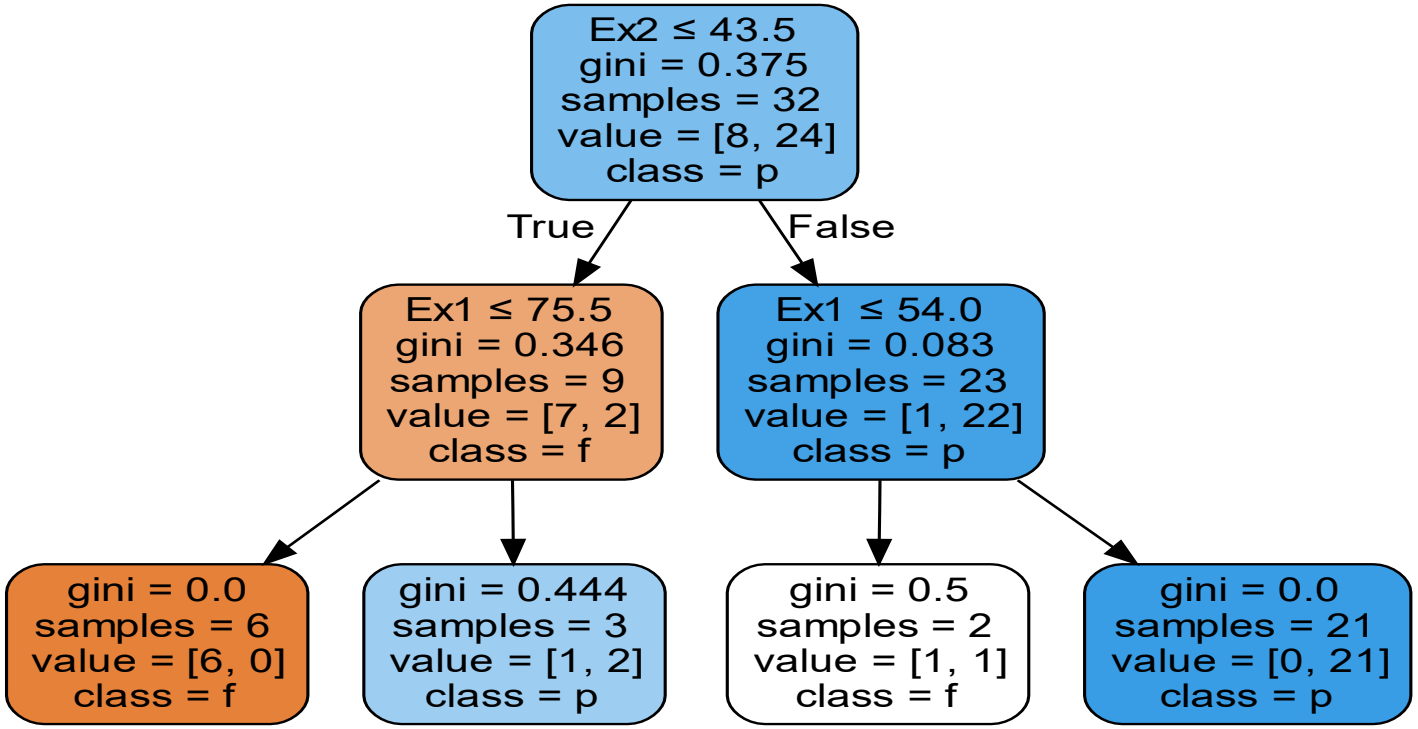
d) La asignación de cada nodo terminal a un valor de la variable de respuesta (regresión) o a una clase (clasificación).

Las diferencias principales entre los algoritmos para construir árboles se hallan en la regla para particionar los nodos, la estrategia para podar los árboles, y el tratamiento de valores perdidos (“missing values”).

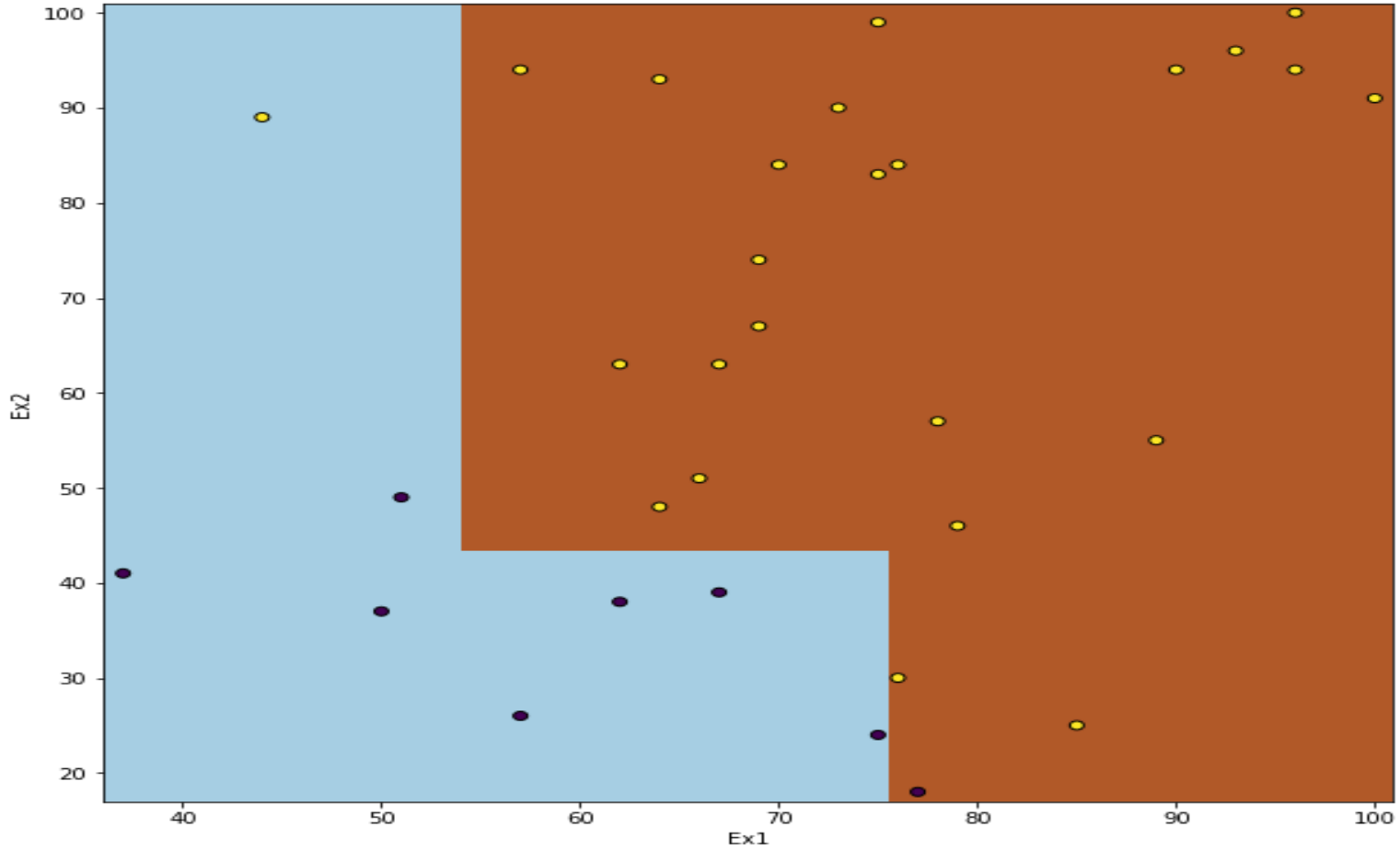
Ejemplo: Prediciendo la nota final en una clase basados en lox examines E1 and E2

```
df=pd.read_csv("http://academic.uprm.edu/eacuna/eje1dis.csv")
#extracting the matrix of predictors and the column corresponding to classes
y=df['Nota']
X=df.iloc[:,0:2]
#Applying the decisión tree algorithm
modeltree = tree.DecisionTreeClassifier(max_depth=2)
modeltree = modeltree.fit(X,y)
modeltree = tree.DecisionTreeClassifier(max_depth=2)
modeltree = modeltree.fit(X,y)
#Finding the predictions
pred=modeltree.predict(X)
print pred
['p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'f' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'p' 'f' 'f' 'f' 'f' 'f' 'f' 'p' ‘f’]
# There are two instances with a wrong prediction
```

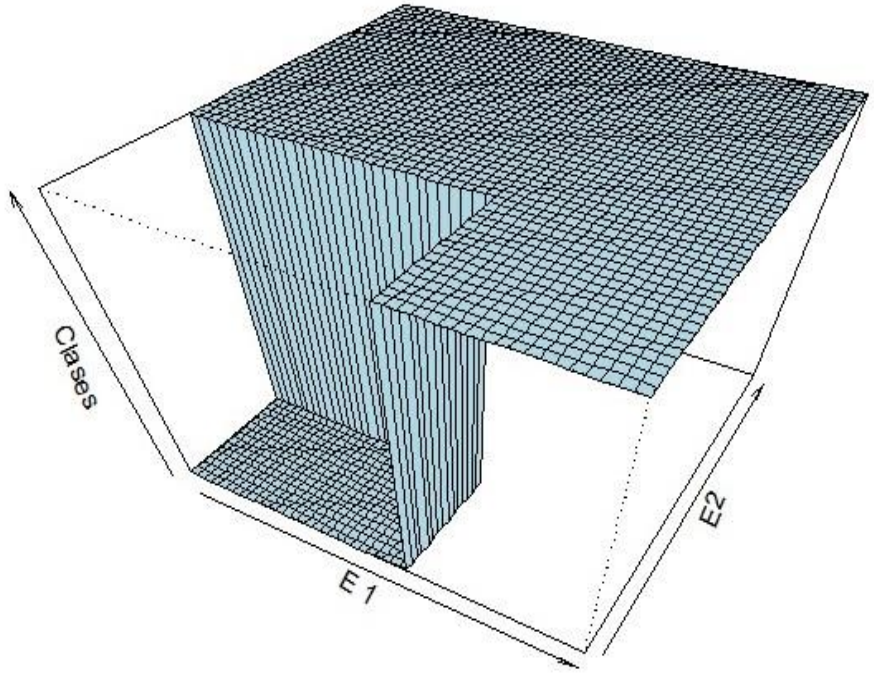
Example: Predicting the final grade in a class based on the scores in E1 y E2



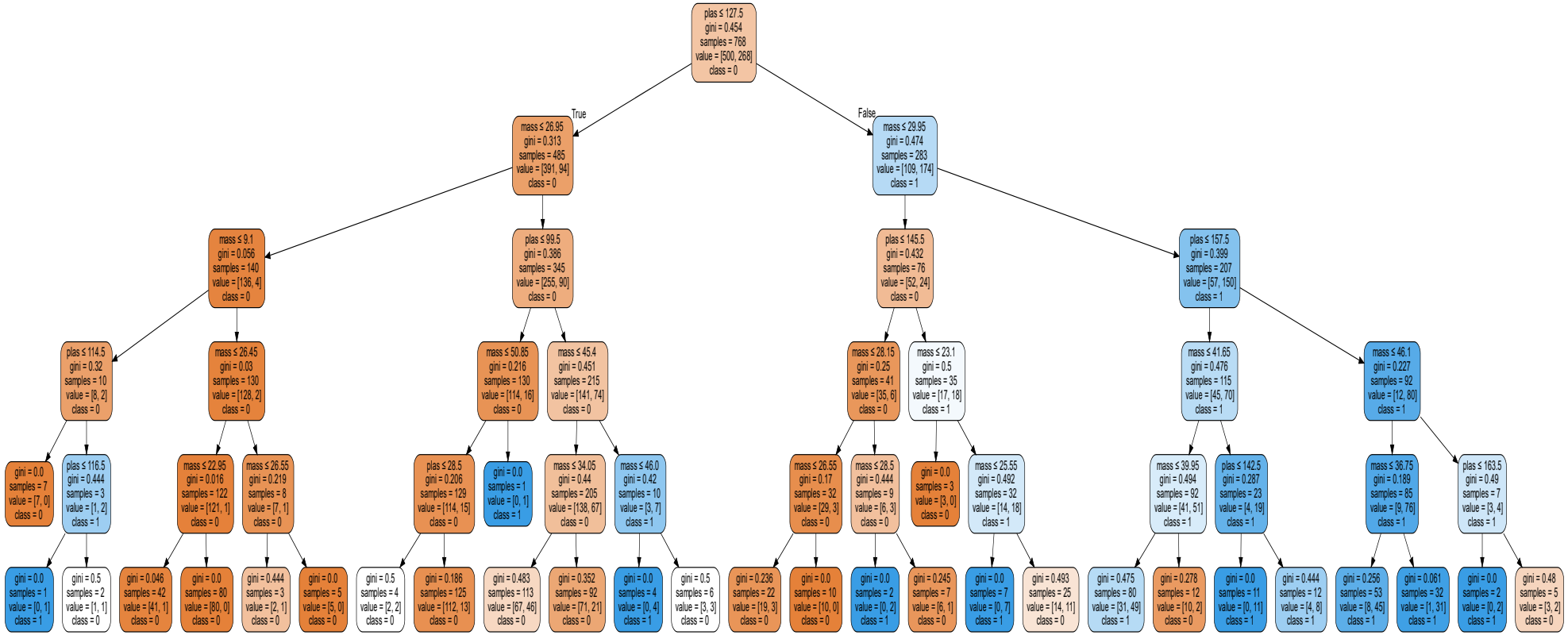
Particionamiento en el espacio muestral de las predictoras



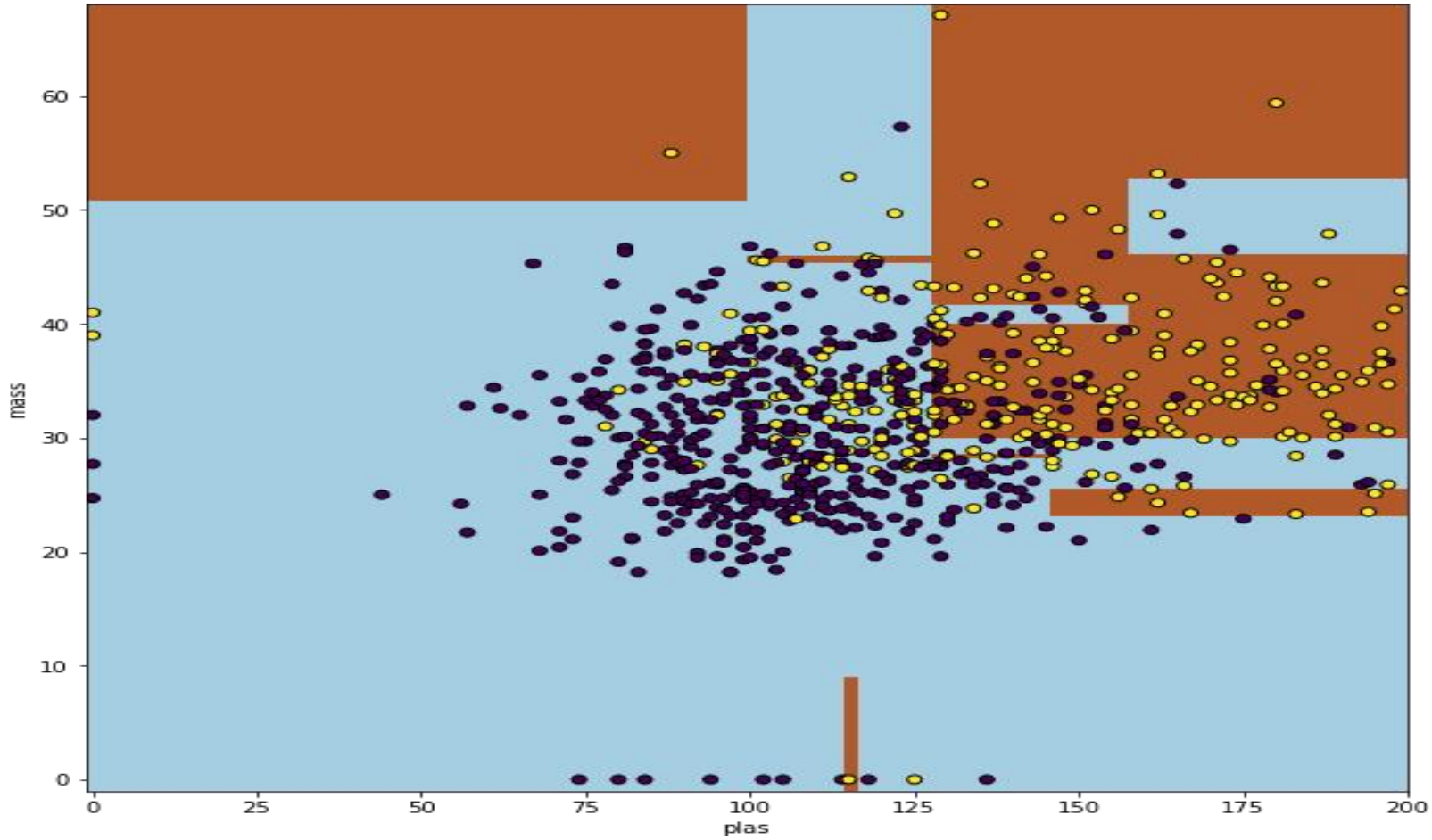
Superficie de respuesta del arbol de decision



Example: Decision Tree for Diabetes using only predictors plas and mass



Partitioning of the sample space corresponding to the previous tree



I-El conjunto de preguntas binarias Q

- Supongamos que el vector de variables predictoras es de la forma $\mathbf{x}=(x_1,\ldots..x_p)$, donde algunas de las variables x_i son discretas y otras son continuas. Entonces, el conjunto Q de preguntas binarias en los nodos debe tener las siguientes características
  - a) cada división de los nodos depende del valor de una sola variable predictora
  - b) Si la variable x_k es continua entonces Q incluye todas las preguntas de la forma $\{\text{Es } x_k \leq c\}$, donde c es cualquier número real. Usualmente c es el punto medio entre dos valores consecutivos de un atributo.

c) Si la variable x_k es categórica tomando valores en $\{b_1,b_2,\ldots\ldots b_m\}$ entonces Q incluye todas las preguntas de la forma $\{ x_k \in A?\}$ donde A es un subconjunto cualquiera de $\{b_1,b_2,\ldots\ldots b_m\}$. En total se pueden considerar $2^{m-1}-1$

- Por ejemplo si x2, x3 y x4 son variables predictoras continuas y x1 es categórica con valores 0, 1 y 2, entonces Q incluye preguntas de la siguiente forma:

Es $x3 \leq 4.5$?

Es $x4 \leq -1.4$?

Es $x1 = 0$ ó 1?

- También se puede usar divisiones en mas de dos nodos, pero no se recomienda porque el conjunto de datos se dividiría muy rápido dejando muy pocos datos para las subsiguientes divisiones.

II-Procedimiento para particionar los nodos

La idea fundamental es que los nodos hijos sean más puros que los nodos padres. La partición de un nodo t del árbol T se hace de acuerdo a un criterio que es diseñado para producir nodos hijos que produzcan una suma de cuadrados de errores menor que la del nodo padre en el caso de regresión o que separen mejor las clases que el nodo padre en el caso de clasificación.

En árboles de clasificación sean $p(s)=\{\#\ i\leq N: Xi\in s\}/N$ la proporción de observaciones en el nodo s, y

$p(j/s)=\{\#\ i\leq N: Xi \in s \text{ y } Yi=j\}/\{\#\ i\leq N: Xi \in s\}$

la proporción de observaciones en el nodo s que pertenecen a la clase j (j=1,….,J), donde J es el número de clases.

El indice de la impureza del nodo s se define como $i(s)=\varphi(p(1/s),p(2/s),\ldots p(J/s))$ donde φ es una función de impureza, la cual debe satisfacer ciertas propiedades.

Entonces la regla para particionar el nodo t es como sigue:

Formar el nodo hijo derecho t_R y el nodo hijo izquierdo t_L tal que la disminución de la impureza dada por

$\Delta i(t)= i(t)-\{p(t_L)i(t_L)+p(t_R)i(t_R)\}$

sea máxima.

Medidas de Impureza

Para árboles de clasificación se pueden usar las siguientes medidas de impureza.

a) El ***Coeficiente de Gini***. *Para el nodo t* ***se*** *define por*

$$i_G(t) = \sum_j \sum_k p(j/t)\,p(k/t) = \sum_{j=1}^{J} p(j/t)(1-p(j/t))$$

Si hay dos clases (J=2) presentes entonces

$i_G(t)=2p(1-p)$,

donde p es la proporción de observaciones en la primera clase.

El coeficiente de Gini se puede interpretar como uno en donde se clasifica cada observación de un nodo a una clase j con probabilidad p(j/t) en lugar de clasificar todas las observaciones a la clase con mayor número de observaciones.

b) La ***Entropia*** **Cruzada o Devianza o Impureza de Información** definida por

$i_E(t)= -\Sigma j\ p(j/t)\log[p(j/t)]$

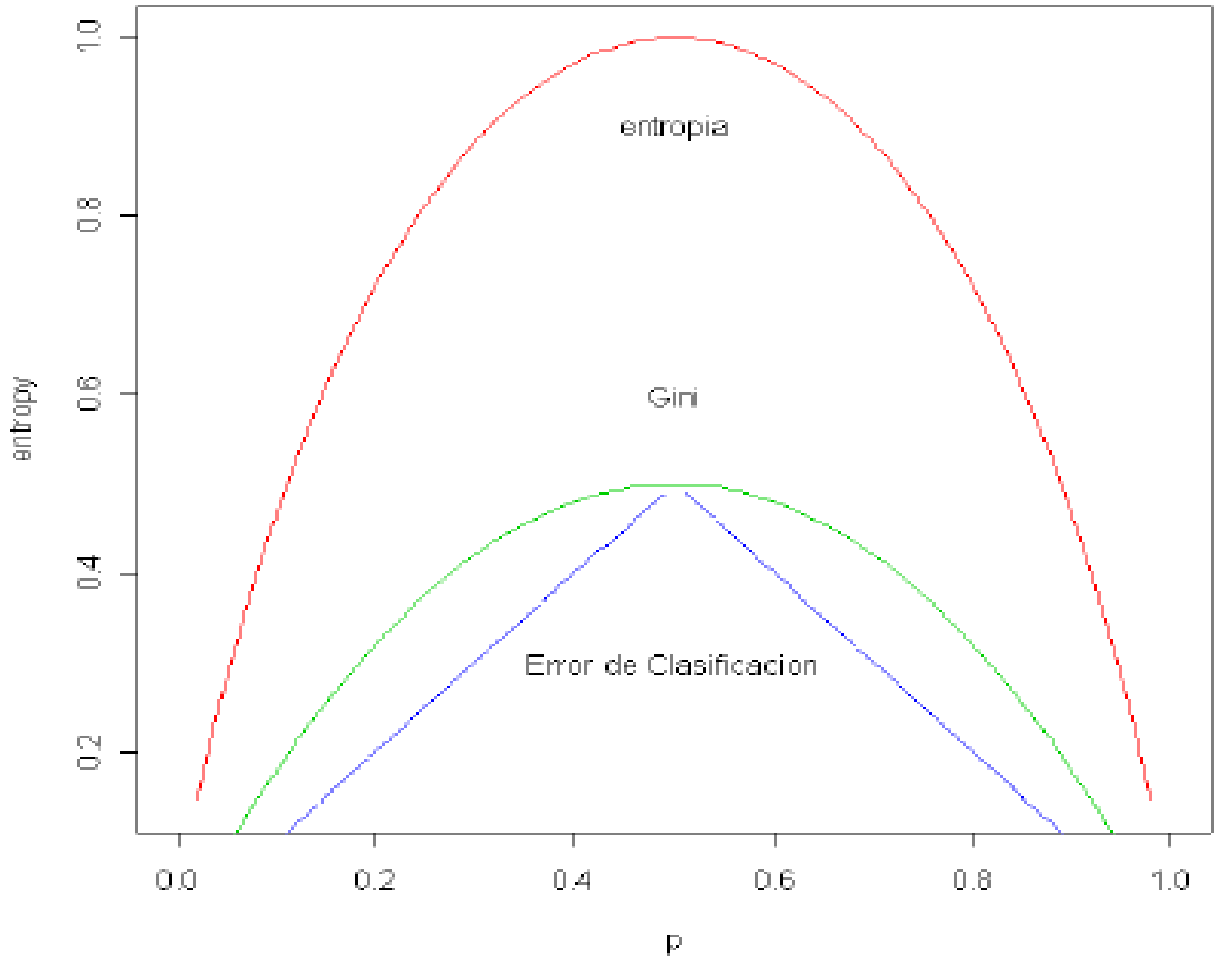
El logaritmo es tomado en base 2. Cuando se aplica entropía a distribuciones continuas se aplica logaritmo natural. Para dos clases se tiene

$i_E(t)= -p*\log(p) -(1-p)*\log(1-p)$

En regresion, La Devianza es equivalente a la suma de cuadrados de los errores y está dada por el negativo del doble del logaritmo de la función de verosimilitud.

Rpart usa por default la impureza Gini, pero tiene la opción de usar la impureza de información. C4.5 usa esta ultima.

- c) La ***tasa de Mala clasificación***, definida por $i_{MC}(t)=1-\max_j p(j/t)$ Para dos clases sería: $i_{MC}(t)=1-\max(p,1-p)$



Example 1: Computing the impurity

In the loan example if we consider the first partition in the two children nodes:

t_R:Status Marital= Soltero , there are 4 subjects not receiving loan and 6 receiving loan.

t_L: Status Marital= Casado, Viudo o Divorciado, there are 6 subjects not receiving loan and 9 receiving loan.

The impurity of t_R would be:

Entropy -4/10*log2(4/10)-6/10*log2(6/10)=.9709

Gini coefficient: 2(4/10)(6/10)=48/100=.48

Missclassification error rate:1-max(4/10,6/10)=4/10=.4

The impurity of t_L would be:

Entropy: -6/15*log2(6/15)-9/15*log2(9/15)=.9709

Gini coefficient: 2(6/15)(9/15)=108/225=.48

Missclassification error rate:1-max(6/15,9/15)=6/15=.4

Example 1: Computing the impurity (cont)

The impurity of the root node, where15 subjects received loan and 10 did not received it, is:

Entropy= -(10/25)*log2(10/25)-(15/25)*log2(15/25)=.9709

Gini=2*10/25*15/25=.48

Missclassification error rate:1-max(10/25,15/25)=.4

Therefore, the decrease of impurity with each function will be:

Δi(t)= .9709-(10/25*.9709+15/25*.9709)=0(Entropy)

Δi(t)= .48-(10/25*.48+15/25*.48)=0(Gini)

Δi(t)= .4-(10/25*.4+15/25*.4)=0 (ME rate)

However, the partition in the children nodes t_R: Sueldo < 2620 (4: No and 0: Yes) and t_L: Sueldo>= 2620 (6: No and 15: Yes), will produce the greatest decrease of impurity. Here, we will show the decrease using the entropy measure.

Δi(t)= .9709-(4/25*0+21/25*.8631)=.2459

In fact, this is the best partition one can get.

Example 2: Computing the impurity

In the example to predict the final grade, suppose that our first node is obtained by splitting at E2=47. Then the children nodes would be:

t_R:E2<47, where 3 students pass and 7 fail and.
t_L:E2>=47, where 21 students pass and 1 fail.

Then,
The impurity of t_R with each measure will be :
Entropy -3/10*log2(3/10)-7/10*log2(7/10)=.8812
Gini coefficient: 2(3/10)(7/10)=.42
Missclassification error rate:1-max(3/10,7/10)=3/10=.3

The impurity of t_L with each measure will be : :
Entropy: -21/22*log2(21/22)-1/22*log2(1/22)=.2667
Gini coefficient i: 2(21/22)(1/22)=.086
Missclasification error rate:1-max(21/22,1/22)=1/22=.045

Example 2: Computing the impurity(cont)

The impurity of the root node, where 8 students failed the class and 24 students passed according to the three measures is given by

Entropy= -(8/32)*log2(8/32)-(24/32)*log2(24/32)=.8112

Gini=2*8/32*24/32=.375

Missclassification error rate:1-max(8/32,24/32)=8/32=.25

Therefore, the decrease of impurity with each function will be

Δi(t)= .8112-(10/32*.8812+22/32*.2667)=.3524(Entropia)

Δi(t)= .375-(10/32*.42+22/32*.086)=.184 (Gini)

Δi(t)= .25-(10/32*.3+22/32*.045)=.125 (ME)

However la partition at the split E2=43.5 will give the greatest decrease. Here we will show only the decrease using the entropy as impurity measure

Δi(t)= .8112-(9/32*.764+23/32*.2580)=.4108.

In fact, this is the best partition.

Impurity measures for trees in regression

For decision trees applied to regression problems, the following two measures can be applied

i) The variance

$$i(t) = \sum_{j \in t} \frac{(y_j - \bar{y}_t)^2}{n_t}$$

Where $\bar{y}_t$ is the mean of the response variable y at the node t.

ii) The median absolute deviation

$$i(t) = \sum_{j} \frac{|y_{j-\bar{Y}_t}|}{n_t}$$

III-Criteria to stop the growth of a tree

The growth of a tree ends when it is imposible to continue, That is,

1) There is only one instance in each of the children nodes
2) All the instances in each children node belong to the same class.
3) The maximum number of levels ("depth") of the tree set by the user has been reached.

The largest tree usually creates overfitting, that is it follows a lot of the trend of the data. The last partitions of the tree cause the overfitting. Therefore pruning the tree is required.

III-Criteria to stop the growth of a tree (cont)

The DecisionTreeClassifer function available at scikit-learn has several option to control the growth of the tree.

min_samples_split : ***(default=2)*****.** The minimum number of samples required to split an internal node:

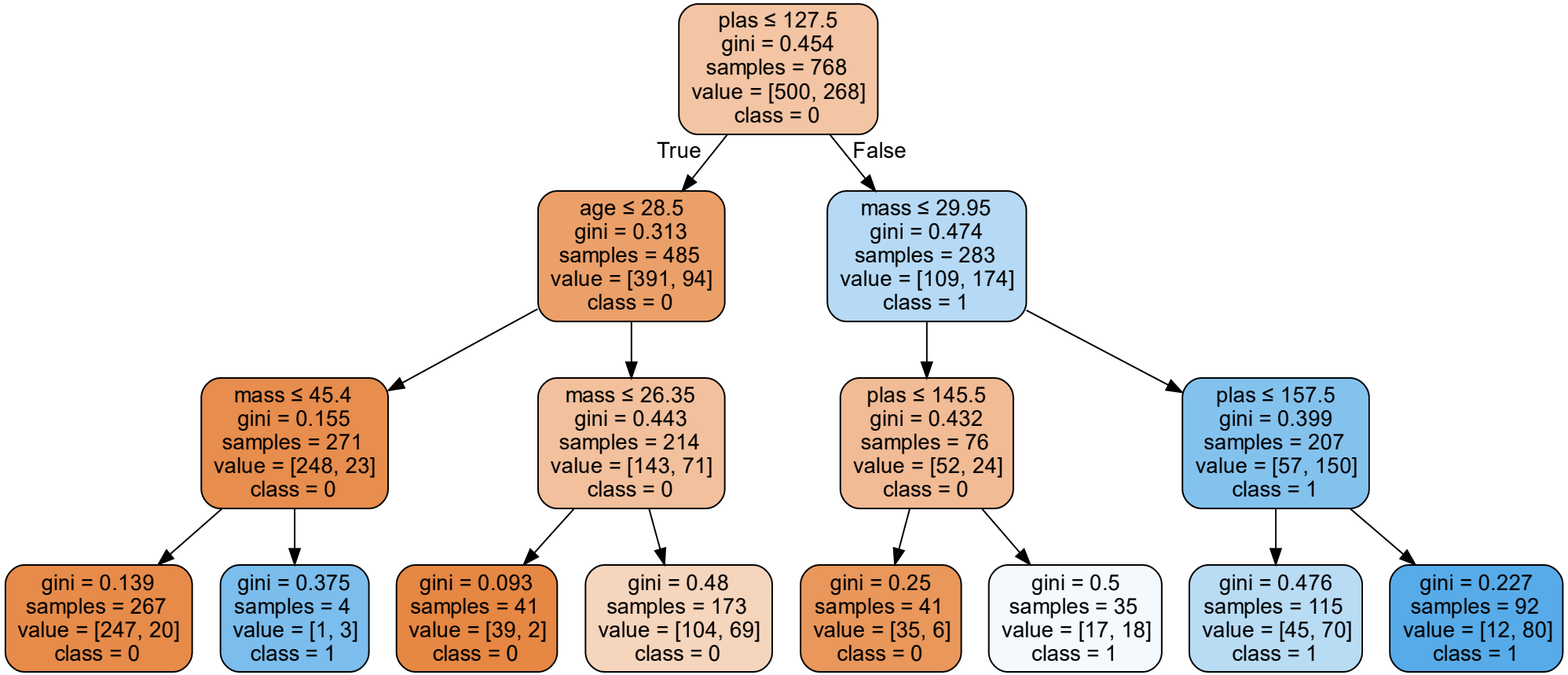
min_samples_leaf : ***int, float, optional (default=1)*****.**The minimum number of samples required to be at a terminal node.

min_impurity_decrease : ***float, optional (default=0.)*** A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

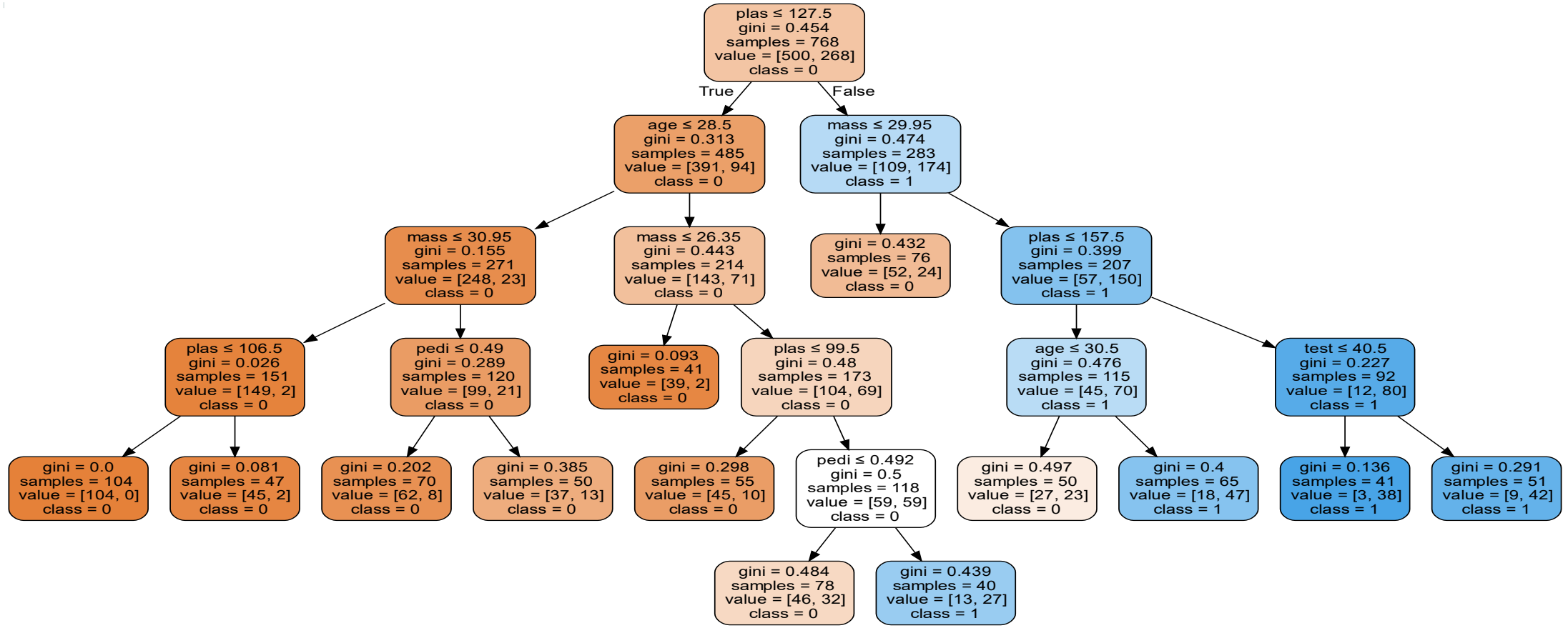
max_leaf_nodes : ***int or None, optional (default=None)*** Grow a tree with max_leaf_nodes in best-first fashion.

max_depth : ***int or None, optional (default=None).***The maximum depth of the tree.

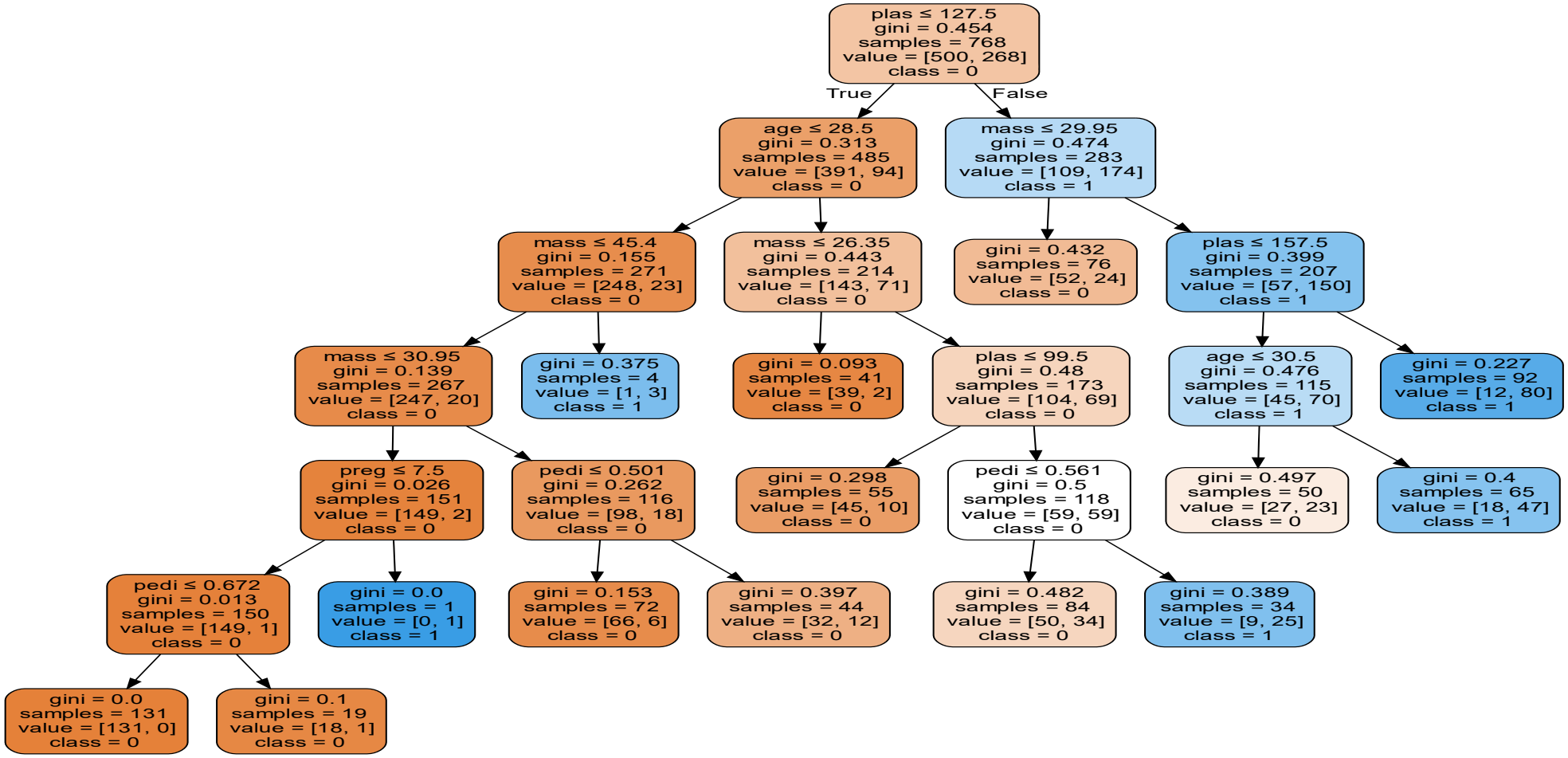
Tree for Diabetes max_depth=3



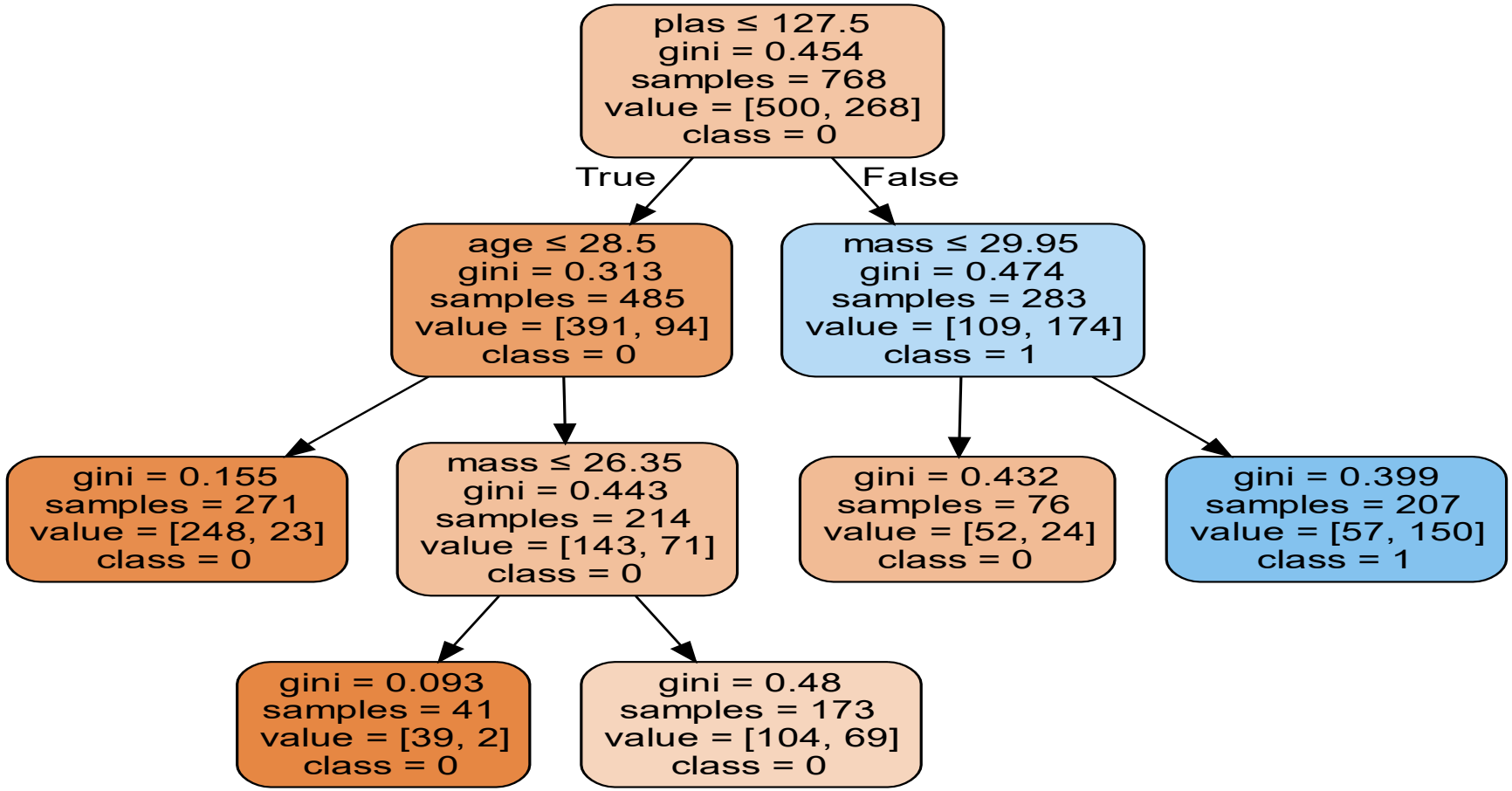
Tree for Diabetes min_samples_leaf=40



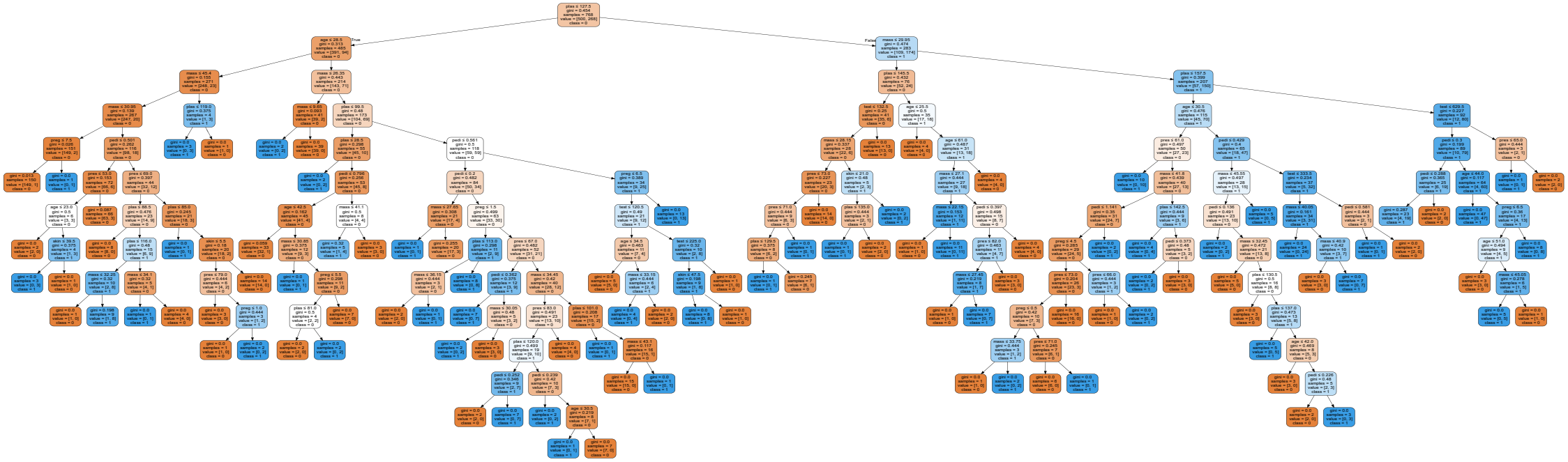
Tree for Diabetes min_samples_split=100



Tree for Diabetes max_leaf_nodes=5



Tree for Diabetes with min_impurity_decrease=.001



Pruning a tree

- To grow a tree too large can cause overfitting. This means that the model follows more the noise than the signal.

For any tree T and any $\alpha \geq 0$ (α is called the complexity parameter), a measure of merit of the tree T (or cost-complexity measure) is given by:

$R_\alpha(T) = Resub(T) + \alpha|T|$

where Resub(T) is the estimated by resubstitution of the misclassification error rate using T, and |T| is the number of terminal nodes in T. When $\alpha=0$, the largest possible tree is obtained and when $\alpha=\propto$ a tree with a single node is produced. Thus, as α increases the tree is pruned more and more.

The optimal tree T_α is the smallest tree that minimizes $R_\alpha(T)$.

DecisionTreesClassifier from scikit learn does not have a pruning option

Error estimation

Breiman, et al (1984) recommended the use of 10-fold cross validation to estimate the miss classification error. Other methods are recommended only for small samples.

```
modeltree = tree.DecisionTreeClassifier(max_depth=3)
modeltree = modeltree.fit(X,y)
from sklearn.model_selection import cross_val_score
scores = cross_val_score(modeltree, X, y, cv=10)
#Hallando la precision media y un intervalo de confianza
print("CV Accuracy: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std() * 2))
CV Accuracy: 0.732 (+/- 0.078)
modeltree4 = tree.DecisionTreeClassifier(min_impurity_decrease=.001)
modeltree4 = modeltree4.fit(X,y)
scores = cross_val_score(modeltree4, X, y, cv=10)
#Hallando la precision media y un intervalo de confianza
print("CV Accuracy: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std() * 2))
CV Accuracy: 0.719 (+/- 0.128)
```

Handling of missing values in decision trees

In the CART algorithm the procedure to handle the missing values is as follows:

The best partition of a node based on a predictor, say x_k is found only with the available data. If when a observation either from the training or the test sample need to be classified and there is not a corresponding value of the variable x_k then a surrogate partition is used. If the observed value of the variable in this partition is also missing then a second surrogate partition is used and so on.

This is somehow similar when in a linear model a missing value of a predictor variable is replaced by the predicted value with the regression with other predictor that is most correlated wit it.

Other decision trees algorithm have a more complicated treatment of missing values.

Advantages and disadvantages of tree classification

Advantages:

- It can be applied to any type of predictors: continuous and categorical.
- The results are very easy to understand and to be interpreted.
- There is not problem to work with missing values
- It performs automatically feature selection
- It is invariant to transformations of the predictor variables.
- It is robust to the presence of outliers.
- Is a non-prametric classifier, this means that it does not require assumptions.
- Fast computation.

Disadvantages:

- The feature selection process is biased towards the feature with more different values.
- It is not easy to establish the optimal tree.
- The prediction surface is not smooth, since it is a set of hyperplanes.
- It requires a large amount of data to assure that there is a significative number of instances in each terminal node.
- There is not a mathematical model.
- It does no take into account the interactions that can exists between the predictor variables.

Random Forest

Edgar Acuña

Department of Mathematical Sciences

UPR-Mayagüez

April 2019

Random Forest

Edgar Acuna

Departamento de Ciencias Matematicas

UPR-Mayaguez

Octubre 2018

Definicion

- **Random forest** es una combinacion de muchos arboles de decision y cuya clase predicha es obtenida por votacion (moda) de las clases predichas por los arboles indviduales.
- Fue introducido por L. Breiman en el 2001.
- El metodo combina “bagging” con la seleccion al azar de atributos predictores.

Algoritmo-1

1. Considerar que la muestra de entrenamiento tiene N instancias y que el numero de variables predictoras es M.

2. Construir n (usualmente 500) muestras de entrenamiento para el arbol, escogiendo muestras con reemplazo y del mismo tamano de la muestra orignal (muestras bootstrap). Las instancias que no son elegidas forman una muestra de prueba y son usadas para estimar la tasa de error del arbol mediante la prediccion de sus clases (estimacion out-of-bag del error de clasificacion).

3. Construir un arbol de decision para cada muestra bootstrap pero usando en cada particion solo m atributos elegidos al azar para hallar la mejor particion. Usualmente m=sqrt(M).

Algoritmo-2

4. A diferencia de un clasificador por arboles donde se hace poda, aqui se deja crecer el arbol hasta el final y no se hace poda.

Para predecir una nueva instancia se la hace recorrer el arbol y se le asigna la etiqueta del nodo terminal correspondiente. Este procedimiento es repetido sobre todos los arboles y al final se asigna la clase predicha por votacion.

Otros aspectos de RF

- Toma en cuenta el hecho que el conjunto de datos tiene clases desbalanceadas (Churn problem).
- Calcula proximidades entre pares de datos. Esta se define como la proporcion de las veces que dos instancias caen en el mismo nodo terminal en distintos arboles.
- Estas proximidades pueden ser usadas para hacer clustering, detectar outliers, para imputar valores perdidos (a lo knn-impute) o tambien para visualizar la data (se combina con MDS, multidimensional scaling).
- Tambien sirve para detectar experimentalmente si hay interacciones entre variables predictoras.

Estimando la importancia de cada predictora

1. Construir S random Forest. Para k=1…S, repetir pasos del 2 al 4.
2. Para cada arbol t del k-esimo RF, considerar el estimado OOB_t del error de mala clasificacion.
3. Para cada predictor Xj permutar al azar los valores de X_j para generar una nueva muestra y obtener un nuevo error $OOB_{t(j)}$ usando esta nueva muestra.
4. Una medida de imprtancia del predictor Xj es el promedio de las diferencias $OOB_{t(j)}$– OOBt sobre todos los arboles del k-esimo RF.
5. Una medida promedio de importancia del predictor Xj sera el promedio de las medidas del paso 4 en todos los S random Forests.

Random Forest using scikit-learn

```
url= "http://academic.uprm.edu/eacuna/diabetes.dat"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pd.read_table(url, names=names,header=None)
#La variable de respuesta y debe ser binaria (0,1)
y=data['class']-1
X=data.iloc[:,0:8]
clf = RandomForestClassifier(n_estimators=50,max_depth=10, oob_score=True,random_state=0)
clf.fit(X, y)
print "The accuracy estimated is", clf.score(X,y)
```

The accuracy estimated is 0.98697916667

```
#Estimating the accuracy by cross-validation
rfmodel=clf.fit(X, y)
scores = cross_val_score(rfmodel, X, y, cv=10)
```

Random Forest using scikit-learn(cont)

scores

print("CV Accuracy: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std() * 2))

#Accuracy estimated using out-of-Bag

print clf.oob_score_

0.7421875

Finding the most important features

print(clf.feature_importances_)

[0.08178093 0.25522871 0.08433383 0.07064764 0.06914087 0.17326173 0.12167218 0.14393411]

The most important features are plas mass and age

Random Forest using h2o

```
diabetes =
h2o.import_file("https://academic.uprm.edu/eacuna/diabetes.dat")
myx=['C1','C2','C3','C4','C5','C6','C7','C8']
diabetes['C9']=diabetes['C9'].asfactor()
myy="C9"
model=H2ORandomForestEstimator(ntrees=50,nfolds=10,max_dept
h=10)
model.train(myx, myy, training_frame = diabetes)
y_pred=model.predict(diabetes)
print (y_pred['predict']==diabetes['C9']).mean()
[0.9752604166666666]
model.model_performance(diabetes)
```

Ventajas de RF

- Es uno de los algoritmos de Machine Learning que produce un alto nivel de precision. Para muchos conjuntos de datos da el mas alto nivel de precision.
- Funciona eficientemente con grandes conjuntos de datos
- Puede manipular datos con miles de variables, sin hacer seleccion previa (Bioinfomatica). Fuerte competidor de SVM.
- Estima las variables que son mas importantes para la clasificacion.
- Calcula un estimador insesgado el error a medida que se generan mas arboles.
- Tiene un metodo eficiente de estimar los valores faltantes y mantiene la precision aun con un gran porcentaje de datos faltantes(usando medidad de proximidad).

Desventajas

- Se ha observado que Random forests sobreajusta algunos conjuntos que contiene variables ruidosas.
- Para datos que incluyen variables categoricas con diferente numero de niveles, random forests son sesgados en favor de aquellos atributos con mayor numero de niveles. Por lo tanto, los scores de importancia dados por el RF no son muy confiables para este tipo de datos.

Conclusiones y resumen:

- RF es bien rapido
  - RF es rapido para construir. Aun mas rapido para predecir!
  - En la practica, no requerir validacion cruzada, hace que se gane en rapidez de entrenamiento en un factor que puede superar a 100.
  - Es completamente paralelizable, lo cual lo puede hacer aun mas rapido!
- Incluye automaticamente seleccion de variables.
- RF es resistente a sobreajuste.
- RF tiene la capacidad de analizar datos sin la necesidad de preprocesamiento,
  - Los datos no necesitan ser reescalados, transformados o modificados.
  - Es resistente a outliers
  - El tratamiento de valores faltantes esta incluido en el algoritmo.
- Se puede usar para formar clusters usando la matriz de proximidad generada por RF.

Random Forest using scikit-learn

```
url= "http://academic.uprm.edu/eacuna/diabetes.dat"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
data = pd.read_table(url, names=names,header=None)
#La variable de respuesta y debe ser binaria (0,1)
y=data['class']-1
X=data.iloc[:,0:8]
clf = RandomForestClassifier(n_estimators=50,max_depth=10, oob_score=True,random_state=0)
clf.fit(X, y)
print "The accuracy estimated is", clf.score(X,y)
```

The accuracy estimated is 0.986979166667

```
#Estimating the accuracy by cross-validation
rfmodel=clf.fit(X, y)
scores = cross_val_score(rfmodel, X, y, cv=10)
```

Random Forest using scikit-learn(cont)

scores

print("CV Accuracy: %0.3f (+/- %0.3f)" % (scores.mean(), scores.std() * 2))

#Accuracy estimated using out-of-Bag

print clf.oob_score_

0.7421875

Finding the most important features

print(clf.feature_importances_)

[0.08178093 0.25522871 0.08433383 0.07064764 0.06914087 0.17326173 0.12167218 0.14393411]

The most important features are plas mass and age

"
"

Random Forest using h2o

```
diabetes =
h2o.import_file("https://academic.uprm.edu/eacuna/diabetes.dat")
myx=['C1','C2','C3','C4','C5','C6','C7','C8']
diabetes['C9']=diabetes['C9'].asfactor()
myy="C9"
model=H2ORandomForestEstimator(ntrees=50,nfolds=10,max_dept
h=10)
model.train(myx, myy, training_frame = diabetes)
y_pred=model.predict(diabetes)
print (y_pred['predict']==diabetes['C9']).mean()
[0.9752604166666666]
model.model_performance(diabetes)
```

Random Forest using h2o(cont)

```
ModelMetricsBinomial: drf
** Reported on test data. **
MSE: 0.0428155400192
RMSE: 0.206919163006
LogLoss: 0.18842735731
Mean Per-Class Error: 0.0117313432836
AUC: 0.999526119403
Gini: 0.999052238806
Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.466578477621:
Maximum Metrics: Maximum metrics at their respective thresholds
```

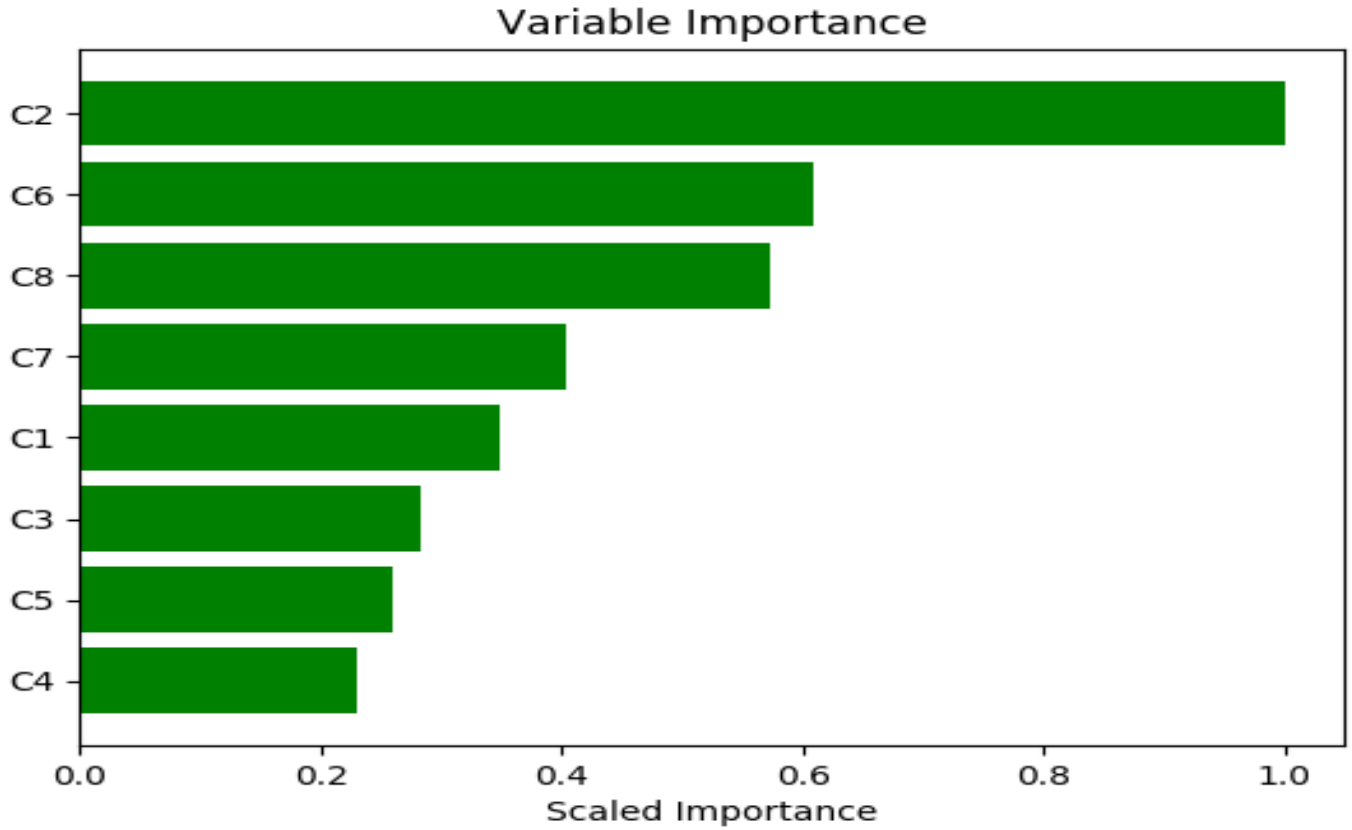
	1	2	Error	Rate
1	497.0	3.0	0.006	(3.0/500.0)
2	6.0	262.0	0.0224	(6.0/268.0)
Total	503.0	265.0	0.0117	(9.0/768.0)

Random Forest using h2o (cont)

Finding the importance of each feature

```
import matplotlib.pyplot as plt
plt.rcdefaults()
fig, ax = plt.subplots()
variables = model._model_json['output']['variable_importances']['variable']
y_pos = np.arange(len(variables))
scaled_importance =model._model_json['output']['variable_importances']['scaled_importance']
ax.barh(y_pos, scaled_importance, align='center', color='green', ecolor='black')
ax.set_yticks(y_pos)
ax.set_yticklabels(variables)
ax.invert_yaxis()
ax.set_xlabel('Scaled Importance')
ax.set_title('Variable Importance')
plt.show()l
```

Random Forest using h2o (cont)



Pros of using RF

- It is one of the Machine Learning algorithms that produces a high level of precision. For many data sets gives the highest level of accuracy.
- Works efficiently with large datasets
- You can manipulate data with thousands of variables, without making previous selection (Bioinformatics). Strong competitor of SVM.
- Estimate the variables that are most important for the classification.
- Calculate an unbiased estimator of the error as more trees are generated

Cons of using RF

- It has been observed that random forests overfits some sets that contain noisy variables.
- For data that includes categorical variables with different number of levels, random forests are biased in favor of those attributes with the highest number of levels. Therefore, the scores given by the RF are not very reliable for this type of data.

Summary and some conclusions:

- RF is very fast
  - RF is quick to build. Even faster to predict!
  - In practice, it does not require cross-validation, it makes you gain speed of training in a factor that can exceed 100
  - It is completely parallelizable, which makes it even faster!
- Automatically includes selection of variables.
- RF is resistant to overfitting.
- RF has the ability to analyze data without the need for pre-processing
  - The data does not need to be rescaled, transformed or modified
  - It is resistant to outliers
  - The treatment of missing values is included in the algorithm.
- It can be used to form clusters using the proximity matrix generated by RF.