UNIVERSIDAD
NACIONAL
DE COLOMBIA

Analítica Predictiva

Carlos A. Madrigal

Profesor Ocasional

Departamento de Ciencias de la Computación y de la Decisión

Maestría en ingeniería - ingeniería de sistemas

Maestría en ingeniería - analítica

Especialización en sistemas

Nota: Este material se ha adaptado con base a diferentes fuentes de información académica

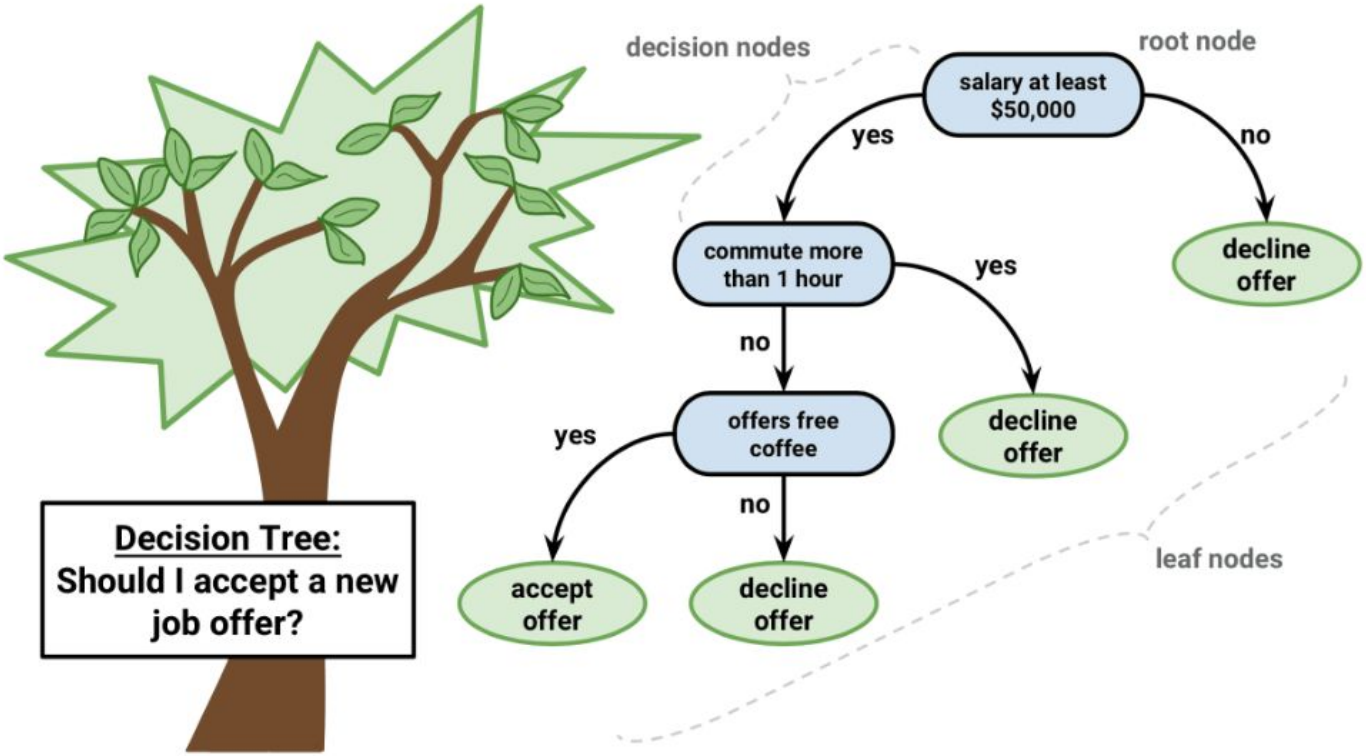
Contenido

Técnicas de Clasificación y Regresión

- Árboles de Decisión
- Random Forest
- XGboost
- Regresión Lineal, Ridge, Lasso

Árboles de Decisión

Son una técnica de aprendizaje supervisado que se aproximan a funciones objetivo de valor discreto. Son usados para tareas de clasificación y regresión.



https://bookdown.org/content/2031/arboles-de-decision-parte-i.html

Árboles de Decisión

- Las funciones de aprendizaje son representadas mediante un ***árbol de decisiones***
- Ayudan a tomar la decisión *más acertada* desde un punto de vista probabilístico.
- Son una de las técnicas ***más usadas*** para inferencia inductiva.
- Divide el espacio de predictores (variables independientes) en regiones distintas y no sobrepuestas.
- Las variables de entrada y salida pueden ser categóricas o continuas.
- La construcción del árbol sigue un enfoque top-down.

Árboles de Decisión

Ventajas:

- Es robusto a los datos ruidosos y capaz de aprender expresiones disyuntivas.
- Es útil en la identificación de variables relevantes. Permite hacer ***interpretabilidad*** de la solución.
- Tiene una concepción muy simple.

Desventajas:

- Puede sufrir fácilmente de sobreajuste
- Normalmente otras técnicas de clasificación obtienen resultados más precisos.
- Inestabilidad en la estructura del árbol.

Árboles de Decisión

Términos.

- Nodo de Decisión (Leaf nodes): Contienen las predicciones o salidas.
- Nodos de probabilidad (Interior nodes): En este punto ocurre un evento aleatorio.
- Nodo raíz (root node): Inicio del arbol

Árboles de Decisión

Un problema algo más complejo:

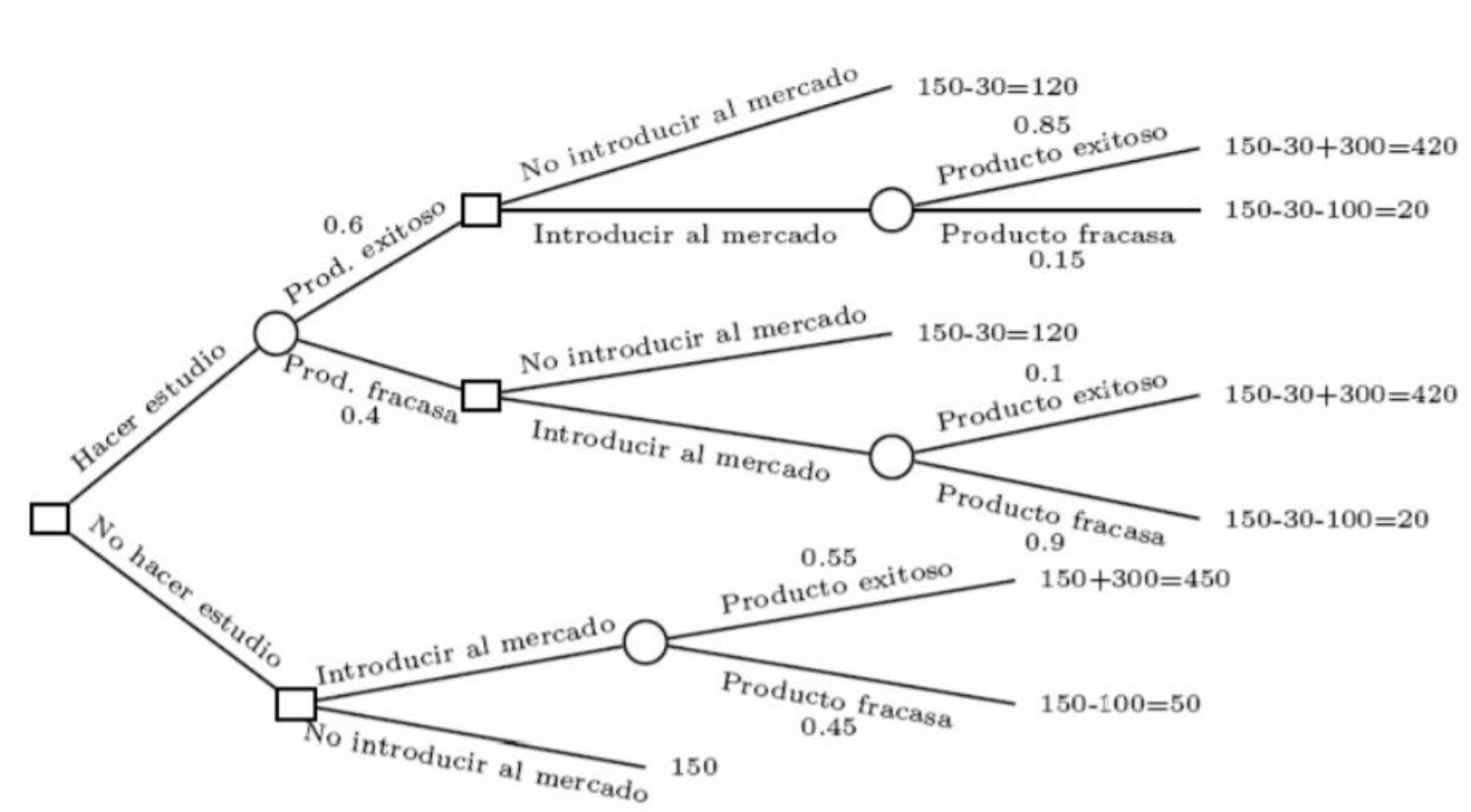
Una fábrica está evaluada en 150 millones. La fábrica desea incorporar un nuevo producto al mercado. Existen tres estrategias para incorporar el nuevo producto:

- **Alternativa 1** Hacer un estudio de mercado del producto de forma de determinar si se introduce o no al mercado.
- **Alternativa 2** Introducir inmediatamente el producto al mercado (sin estudio).
- **Alternativa 3** No lanzar inmediatamente el producto al mercado (sin estudio).

En **ausencia de estudio de mercado**, la fábrica estima que el producto tiene un **55%** de posibilidades de ser **exitoso** y de **45%** de ser un **fracaso**. **Si el producto es exitoso, la fábrica aumentaría en 300 millones** su valor, **si el producto fracasa se devaluaría en 100 millones**. El estudio de mercado vale 30 millones. El **estudio predice** que existe un **60%** de probabilidad de que el **producto sea exitoso**. Si el **estudio de mercado determina que el producto sería exitoso, existe un 85% de posibilidades de que efectivamente lo sea**. Si el estudio de mercado determina que el **producto sería un fracaso**, existe sólo un **10% de posibilidades de que el producto sea exitoso**. Si la empresa no desea correr riesgos (desea maximizar el valor esperado de la empresa). **¿Qué estrategia debería seguir ?**

Árboles de Decisión

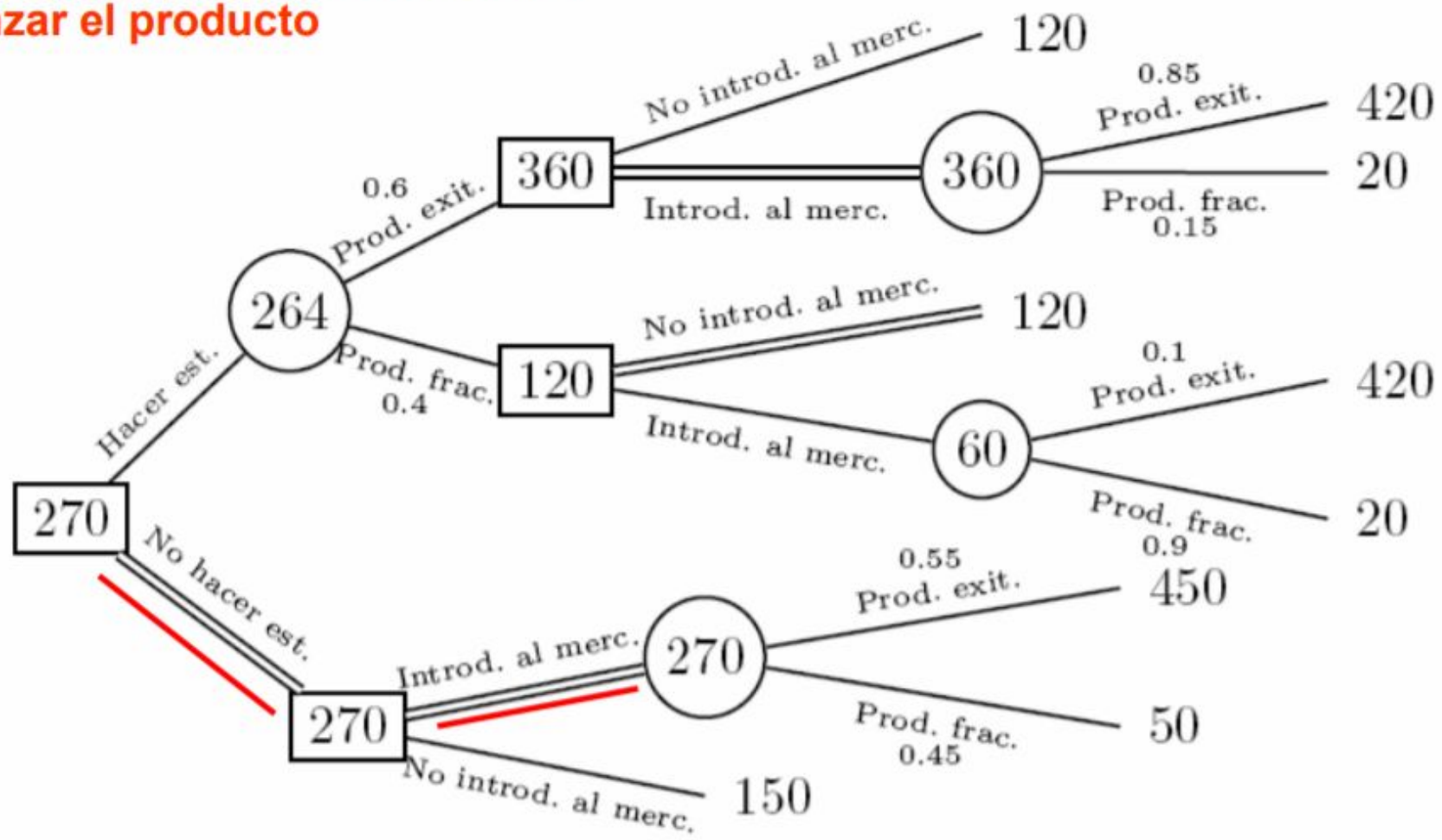
Árbol asociado



Árboles de Decisión

Solución:

No hacer estudio de mercado y lanzar el producto



Árboles de Decisión

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from adspy_shared_utilities import plot_decision_tree
from sklearn.model_selection import train_test_split


iris = load_iris()

X_train, X_test, y_train, y_test = train_test_split(iris.data, iris.target,
                                                    random_state = 3)
clf = DecisionTreeClassifier().fit(X_train, y_train)

print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 0.97
```

```
clf2 = DecisionTreeClassifier(max_depth = 3).fit(X_train, y_train)

print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf2.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf2.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 0.98
Accuracy of Decision Tree classifier on test set: 0.97
```

Árboles de Decisión

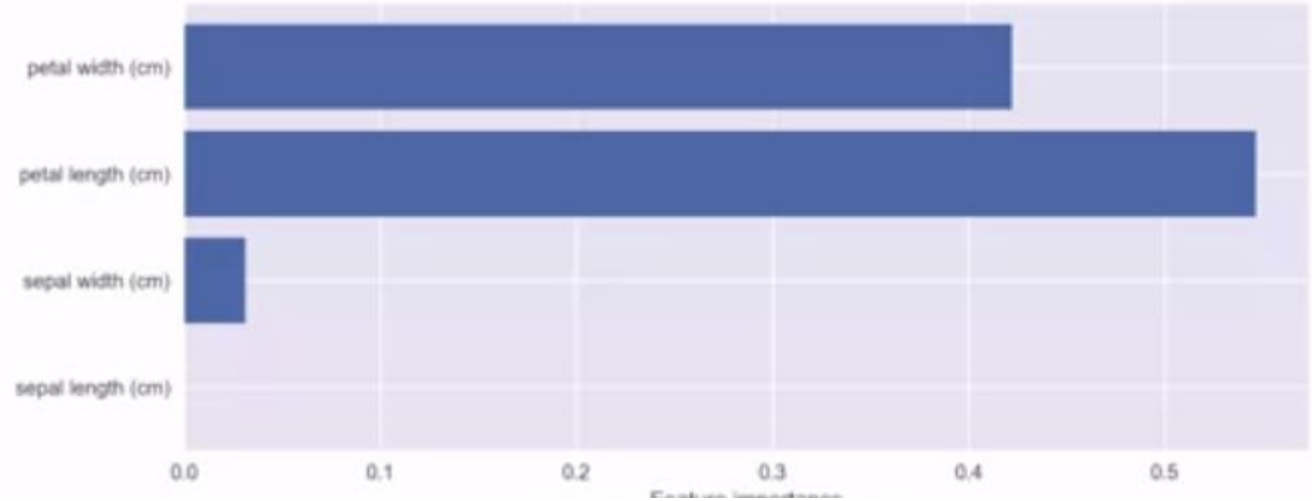
```
clf2 = DecisionTreeClassifier(max_depth = 3).fit(X_train, y_train)

print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf2.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf2.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 0.98
Accuracy of Decision Tree classifier on test set: 0.97
```

Árboles de Decisión

```
from adspy_shared_utilities import plot_feature_importances

plt.figure(figsize=(10,4))
plot_feature_importances(clf, iris.feature_names)
plt.show()

print('Feature importances: {}'
      .format(clf.feature_importances_))
```

Figure 7



Árboles de Decisión

```
from sklearn.tree import DecisionTreeClassifier
from adspy_shared_utilities import plot_decision_tree
from adspy_shared_utilities import plot_feature_importances

X_train, X_test, y_train, y_test = train_test_split(X_cancer, y_cancer,
                                                    random_state = 0)

clf = DecisionTreeClassifier(max_depth = 4, min_samples_leaf = 8,
                             random_state = 0).fit(X_train, y_train)

plot_decision_tree(clf, cancer.feature_names, cancer.target_names)
```

Cuál característica es mejor?

- Lo ideal es seleccionar como nodo raíz la característica más útil en el proceso de clasificación.
- La ***ganancia de información*** mide cuán bien una característica separa los ejemplos de entrenamiento de acuerdo a los etiquetas de salida.
- En árboles de decisión es común usar como medida de ganancia de información la ***entropía.***
- La ***entropía*** mide la pureza o impureza de un conjunto de ejemplos***.***
- Los métodos ID3 y C4.5 usan las entropía

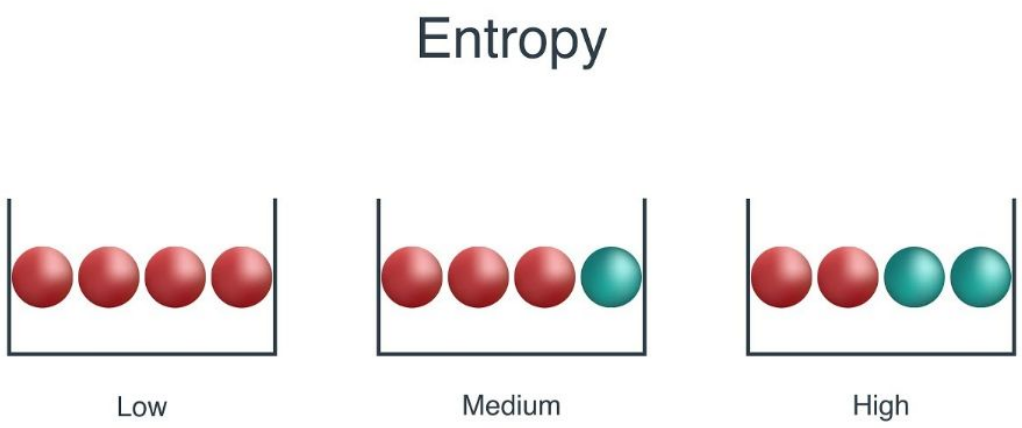
ENTROPÍA

Dado un conjunto de ejemplos *S*, positivos y negativos con respecto a una etiqueta de salida, la ***entropía*** de *S* será:

$$Entropia(S) = -p_+ log_2\, p_+ - p_- log_2\, p_-$$

p_- : Proporción de ejemplos negativos

p_+ : Proporción de ejemplos positivos



Entropía

$$\text{Entropy}([9+,5-] = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0.940$$

$$\text{Entropy}([12+,4-] = -(12/16)\log_2(12/16) - (4/16)\log_2(4/16) = 0.811$$

$$\text{Entropy}([12+,5-] = -(12/17)\log_2(12/17) - (5/17)\log_2(5/17) = 0.874$$

$$\text{Entropy}([8+,8-] = -(8/16)\log_2(8/16) - (8/16)\log_2(8/16) = 1.0$$

$$\text{Entropy}([8+,0-] = -(8/8)\log_2(8/8) - (0/8)\log_2(0/8) = 0.0$$

$$\text{Entropy}([0+,8-] = -(0/8)\log_2(0/8) - (8/8)\log_2(8/8) = 0.0$$

- It is assumed that $\log_2(0)$ is 0

Ganancia de Información

Mide la reducción esperada en entropía si se particiona los ejemplos de acuerdo a un atributo.

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

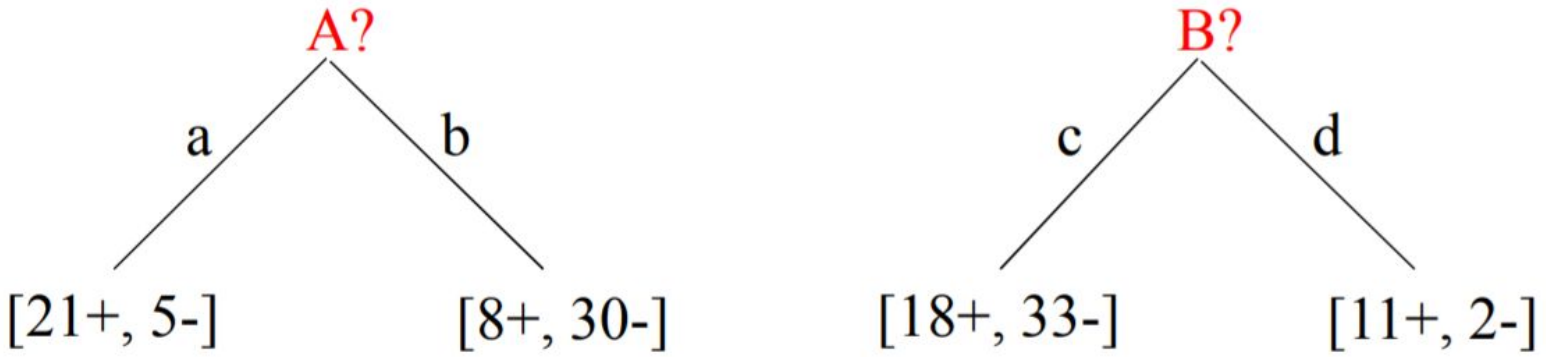
- S – a collection of examples
- A – an attribute
- Values(A) – possible values of attribute A
- S_v – the subset of S for which attribute A has value v

Ganancia de Información

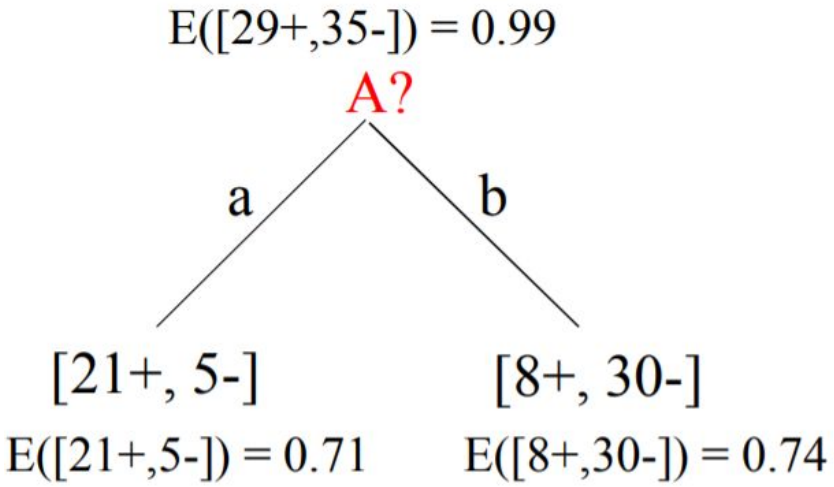
$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14) Entropy(S_{Weak}) \\
&\quad - (6/14) Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

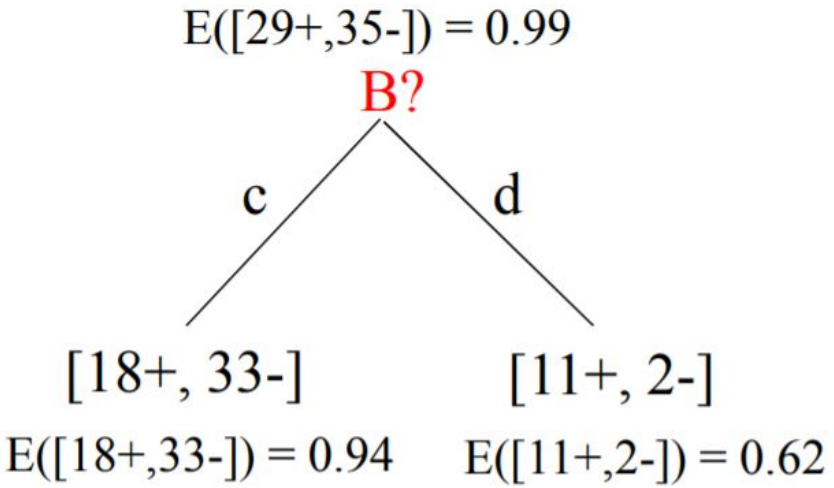
Cuál característica es mejor clasificador?

- S: [29+,35-] Attributes: A and B
- possible values for A: a,b possible values for B: c,d
- Entropy([29+,35-]) = $-29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$



Cuál característica es mejor clasificador?





Gain(S,A) = Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
= **0.27**

Gain(S,B) = Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
= **0.12**

A provides greater information gain than B.
A is a better classifier than B.

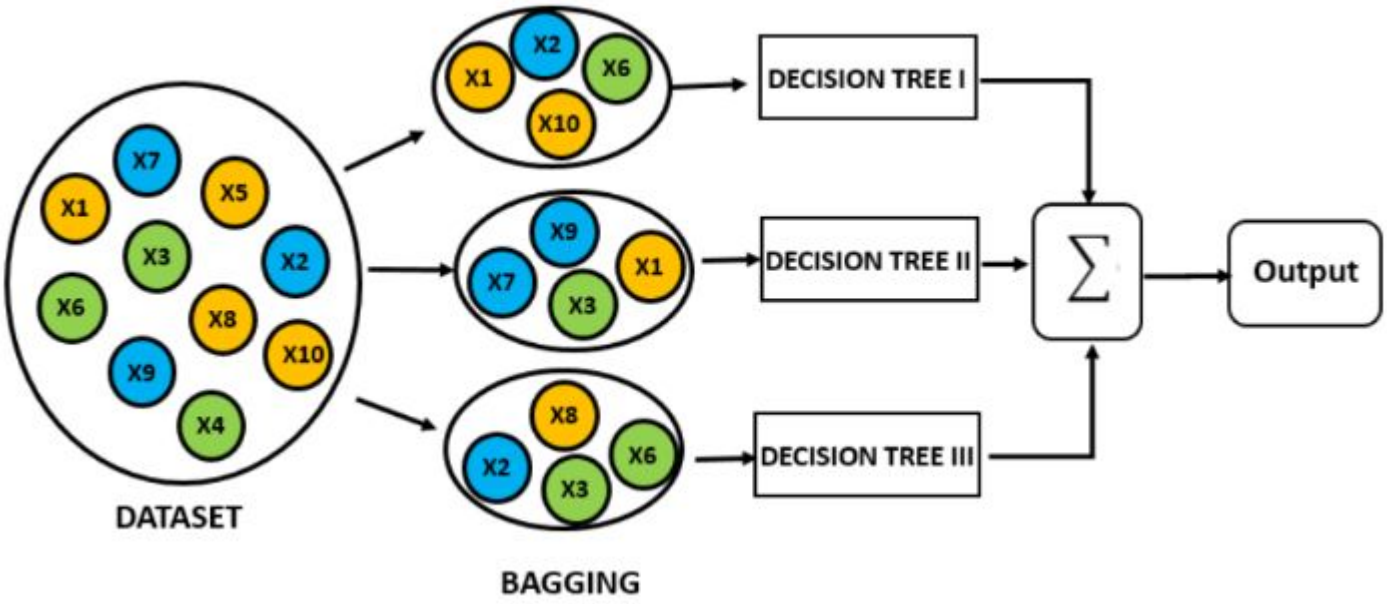
ÍNDICE GINI

- A mayor índice GINI mayor la homogeneidad. La pureza del nodo se incrementa.
- Solo para divisiones binarias
- CART (Classification and Regression Tree) usa el método de Gini para la división binaria.

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

ENSAMBLES

Los métodos tipo ensamblador están formados de un grupo de modelos predictivos que permiten alcanzar una mejor precisión y estabilidad del modelo. Estos proveen una mejora significativa a los modelos de árboles de decisión. Los ensambladores más comunes son: Bagging, Boosting y Stacking. Random Forest es tipo Bagging.



RANDOM FOREST

- Random forest es considerado uno de los métodos más robustos.
- Puede usarse como técnica para la reducción de la dimensionalidad.
- Para regresión, se toma el promedio de las salidas (predicciones) de todos los árboles.
- Random Forest no hace ninguna suposición acerca de la escala y normalización de los datos.
- Si la tarea es de clasificación se escoge el árbol con mayor votos. si la tarea es de regresión se calcula el promedio de la salida de cada árbol.

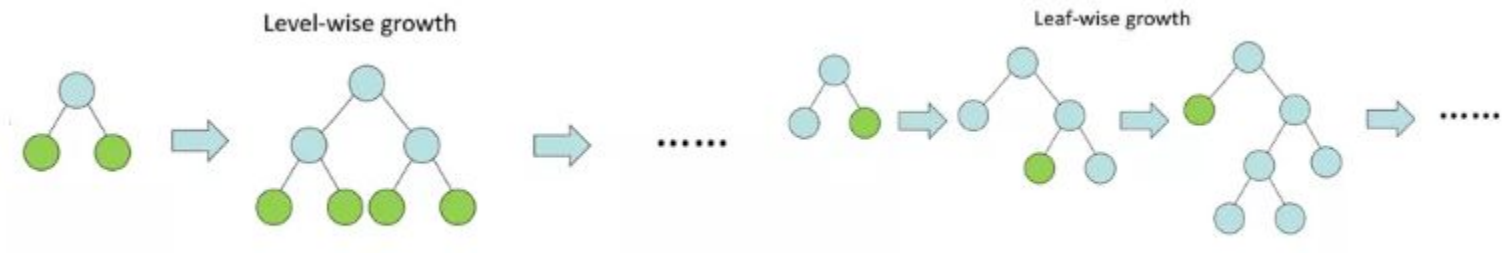
RANDOM FOREST

```
from sklearn.tree     import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix


bc = load_breast_cancer()
X = bc.data
y = bc.target

# Create our test/train split
X_train, X_test, y_train, y_test = train_test_split(X,y, random_state=42)


## build our models
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier(n_estimators=100)

## Train the classifiers
decision_tree.fit(X_train, y_train)
random_forest.fit(X_train, y_train)

# Create Predictions
dt_pred = decision_tree.predict(X_test)
rf_pred = random_forest.predict(X_test)

# Check the performance of each model
print('Decision Tree Model')
print(classification_report(y_test, dt_pred, target_names=bc.target_names))

print('Random Forest Model')
print(classification_report(y_test, rf_pred, target_names=bc.target_names))

#Graph our confusion matrix
dt_cm = confusion_matrix(y_test, dt_pred)
rf_cm = confusion_matrix(y_test, rf_pred)
```

XGBOOST: eXTREME GRADIENT BOOSTING

Xgboost es un algoritmo de ensamble, tipo boosting, de árboles de decisión. En un ensamble boosting los árboles son construidos de manera secuencial, por lo que cada árbol siguiente reduce los errores de los árboles previos. Cada árbol aprende de sus predecesores y actualiza los errores.

Xgboost es uno de los métodos más populares de modelado de bases de datos tabulares de cualquier tamaño, es muy rápido y escalable.



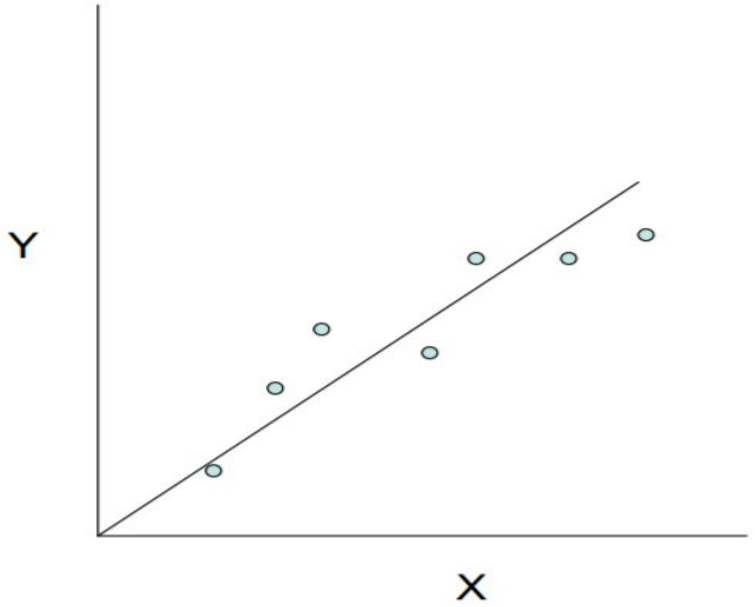
An illustration demonstrating the difference between level-wise and leaf-wise growth

Regresión Lineal

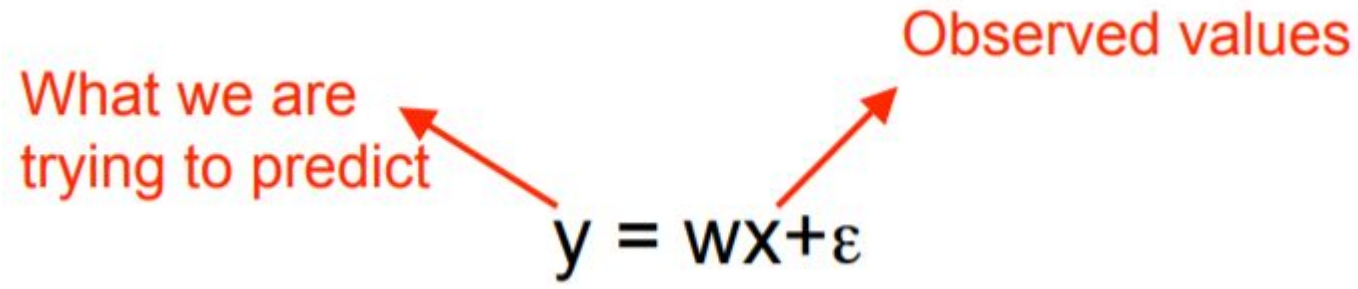
Es una técnica de machine learning usada para predecir el valor de una variable a partir del valor de otra variable. El objetivo es obtener estimaciones razonables de Y para distintos valores de X a partir de una muestra de n pares de valores (x1, y1), . . . ,(xn, yn).

Ejemplos:

- Estimar el precio de una vivienda en función de su superficie.
- Estimar el valor de un tiquete aéreo internacional a partir de la tasa de cambio.



Regresión Lineal



w representa los parámetros y ε representa el ruido. El objetivo es estimar w desde un conjunto de datos de entrenamiento (xi,yi), lo cual podría hacerse usando mínimos cuadrados.

$$\arg\min_w \sum_i (y_i - wx_i)^2$$

Regresión Lineal

$$\arg\min_w \sum_i (y_i - wx_i)^2$$

Usando el descenso del gradiente como la técnica de optimización.

$$\frac{\partial}{\partial w} \sum_i (y_i - wx_i)^2 = 2 \sum_i - x_i (y_i - wx_i) \Rightarrow$$

$$2 \sum_i x_i (y_i - wx_i) = 0 \Rightarrow$$

$$\sum_i x_i y_i = \sum_i w x_i^2 \Rightarrow$$

$$w = \frac{\sum_i x_i y_i}{\sum_i x_i^2}$$

Regresión Multivariada

Si se tiene más de una variable independiente, el problema se convierte en tarea de regresión multivariada.

$$y = w_0 + w_1 x_1 + \ldots + w_k x_k + \varepsilon$$

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1k} \\ 1 & x_{21} & \cdots & x_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

Regresión Multivariada

$$y = w_0 + w_1 x_1 + \ldots + w_k x_k + \varepsilon$$

$$\mathbf{y = Xw + \varepsilon}$$

$$\mathbf{w}^T = [w_0, w_1, \ldots, w_k]$$

$$\mathbf{w = (X^T X)^{-1} X^T y}$$

Regresión Ridge

La regresión ridge sigue el concepto de regresión lineal usando mínimos cuadrados, adicionando una penalidad para variaciones grandes de los parámetros w. Este parámetro de penalidad también es llamado parámetro de regularización y ayuda a prevenir el sobreajuste a los datos.

$$RSS_{RIDGE}(\boldsymbol{w}, b) = \sum_{\{i=1\}}^{N} (\boldsymbol{y}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))^2 + \alpha \sum_{\{j=1\}}^{p} w_j^2$$

$$RSS_{LASSO}(\boldsymbol{w}, b) = \sum_{\{i=1\}}^{N} (\boldsymbol{y}_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b))^2 + \alpha \sum_{\{j=1\}}^{p} |w_j|$$

Ejemplos

Arboles de Decisión https://www.kaggle.com/dmilla/introduction-to-decision-trees-titanic-dataset

Random Forest https://github.com/WillKoehrsen/Machine-Learning-Projects/blob/master/Random%20Forest%20Tutorial.ipynb

XgBoost https://www.datacamp.com/community/tutorials/xgboost-in-python

Regresión Ridge y Lasso
https://github.com/Q-shick/fundamentals_of_data_science/blob/master/mathematical%20_model/Ridge%20and%20Lasso.ipynb

Preguntas

UNIVERSIDAD
NACIONAL
DE COLOMBIA