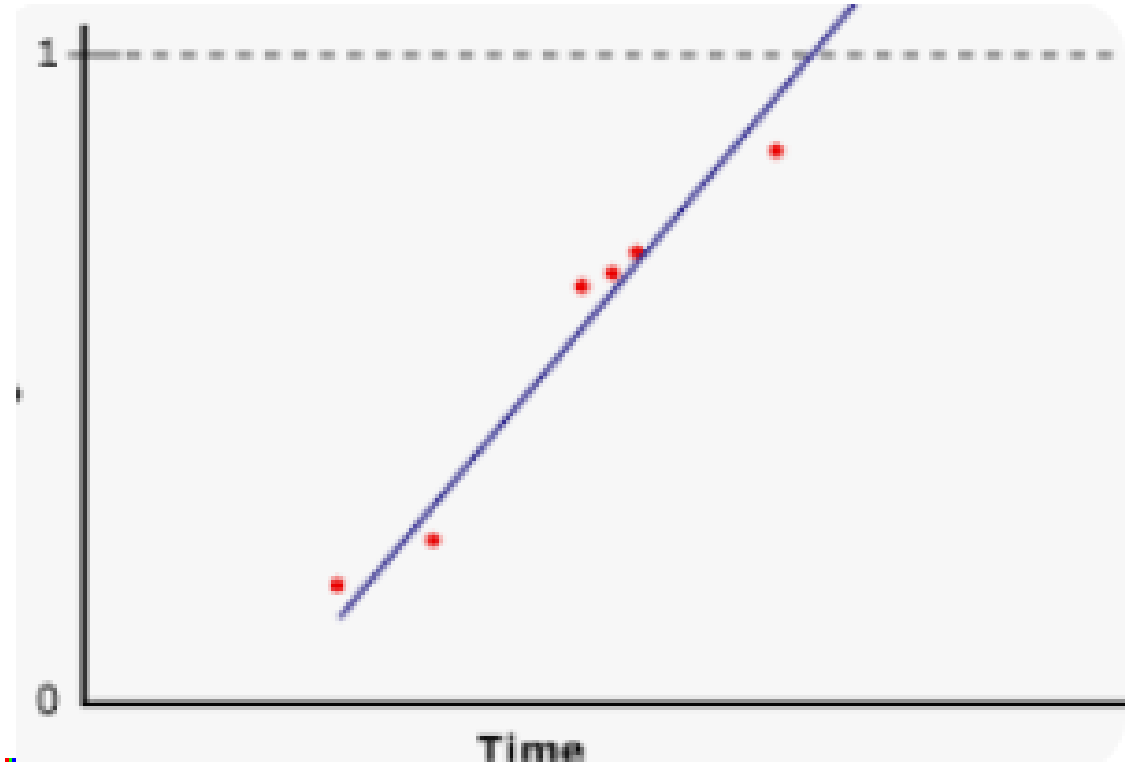
ALGORITMOS DE REGRESIÓN

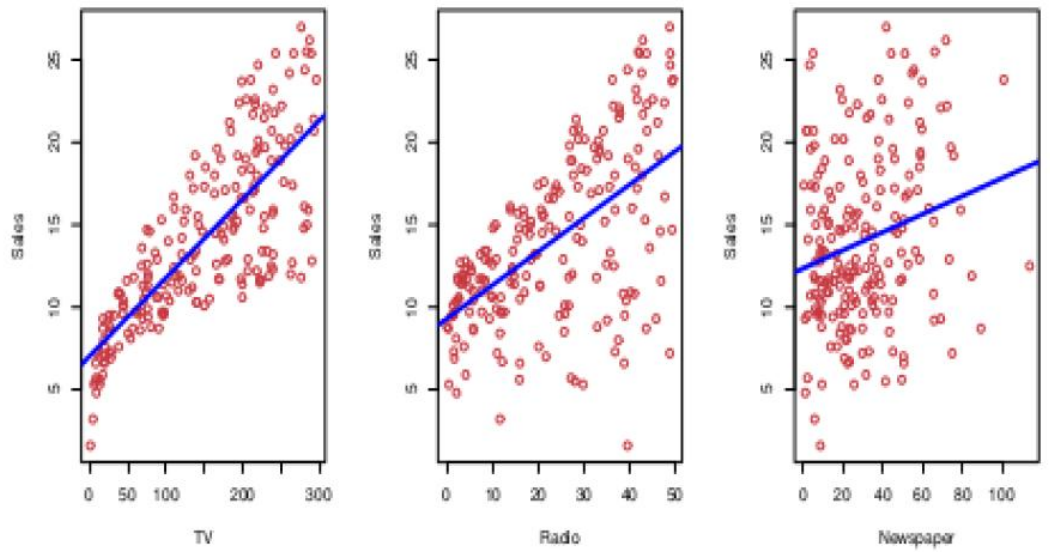
DEFINICIÓN

Los algoritmos de regresión son un tipo concreto de algoritmos de aprendizaje supervisado y consisten en realizar una predicción de una variable numérica o cuantitativa



PREDICCIÓN DE VENTAS

- El gráfico muestra la distribución de las ventas en función del gasto publicitario en TV, radio o prensa escrita.



$$\texttt{Sales} \approx f(\texttt{TV}, \texttt{Radio}, \texttt{Newspaper})$$

$$X = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \end{pmatrix} \qquad Y = f(X) + \epsilon$$

REGRESIÓN: F(X) IDEAL

- ¿Cuál es el valor de Y cuando X=4?

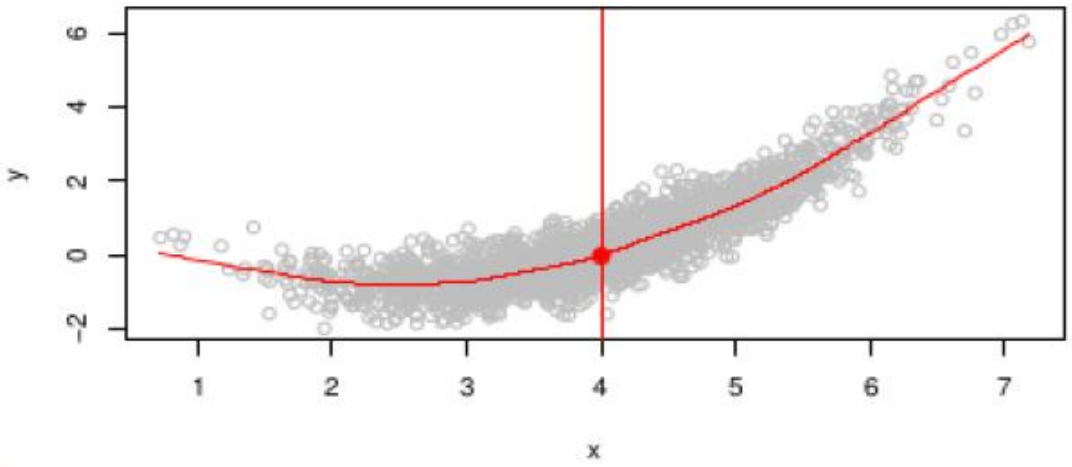
$$f(4) = E(Y|X = 4)$$

- Al f(x) ideal se le conoce como función de regresión

$$f(x) = E(Y|X = x)$$

- También esta definida para un vector de **X**

$$f(x) = f(x_1, x_2, x_3) = E(Y|X_1 = x_1, X_2 = x_2, X_3 = x_3)$$

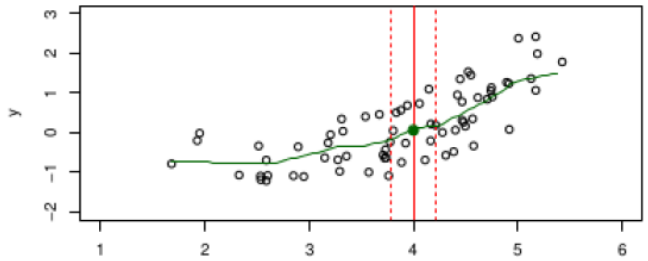


CÓMO ESTIMAR EL F(X) IDEAL?

- Normalmente tendremos pocos o ningún punto en X=4
- Por tanto, no se puede calcular E(Y|X =x)
- La solución es relajar la definición:
  - P,: número características
  - N: número de ejemplos

$$\hat{f}(x) = \text{Ave}(Y|X \in \mathcal{N}(x))$$

where $\mathcal{N}(x)$ is some *neighborhood* of x.



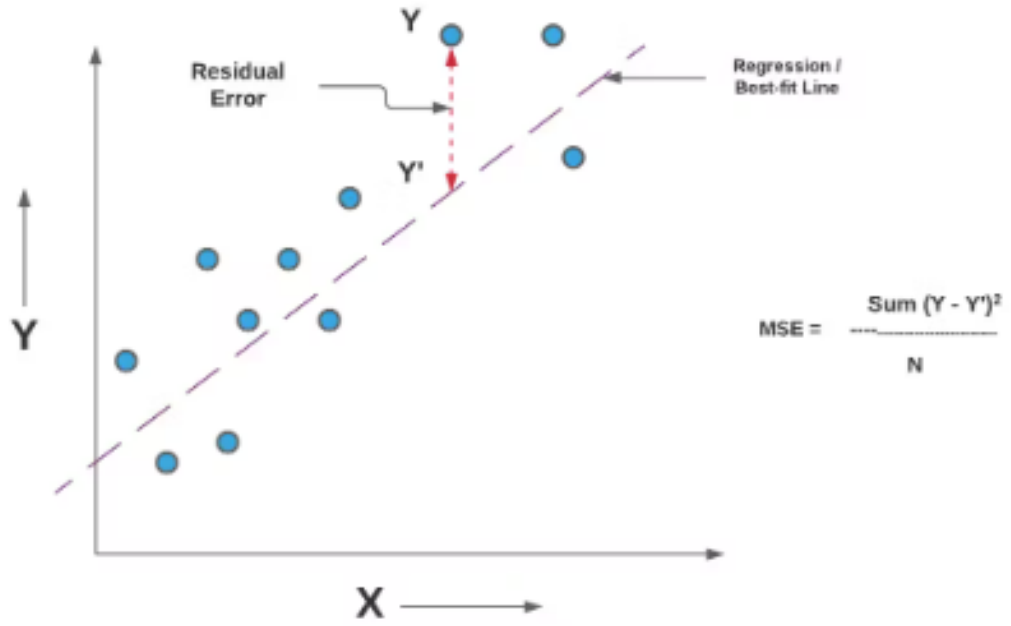
Nota: Estimar los puntos utilizando **Nearest-neighbor averaging** funciona bien para problemas con p pequeña (i.e. p < 5) y N muy grande. (Curse of dimensionality)

MÉTRICAS DE ERROR

MEAN SQUARE ERROR

Se define como la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado

$$MSE = 1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$



MEAN ABSOLUTE ERROR

- Se define como la diferencia en valor absoluto entre el valor real
- y el valor predicho.

$$MAE = 1/n \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

ROOT MEAN SQUARE ERROR (RMSE)

- Se define como la raíz cuadrada de la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado
- Comparada con el error absoluto medio (MAE), amplifica y penaliza los errores grandes.

$$RMSE = \sqrt{1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

ROOT MEAN SQUARE LOGARITHMIC ERROR (RMSE)

- Logaritmo de la raíz del error cuadrático medio, Root Mean Logarithmic Square Error (RMLSE):
- Esta métrica penaliza una under-prediction más que una over-prediction

$$RMLSE = \sqrt{1/n \sum_{i=1}^{n} (\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$$

UNDER PREDICTION (SUBPREDICCIÓN)

Definición

- La subpredicción ocurre cuando un modelo predice un valor más bajo que el valor real

Ejemplos

•**En finanzas**: Un modelo que predice ingresos futuros de una empresa y estima un valor mucho más bajo de lo que realmente será.

•**En diagnóstico médico**: Un modelo que predice una baja probabilidad de enfermedad para un paciente que en realidad tiene una alta probabilidad de padecerla.

•**En pronósticos meteorológicos**: Un modelo que subestima la cantidad de lluvia que caerá

OVER PREDICTION (SOBREPREDICCIÓN)

Definición

La sobrepredicción, por otro lado, ocurre cuando un modelo predice un valor más alto que el valor real

Ejemplos

En finanzas: Un modelo que predice ingresos futuros de una empresa y estima un valor mucho más alto de lo que realmente será, lo que puede llevar a decisiones de inversión erróneas

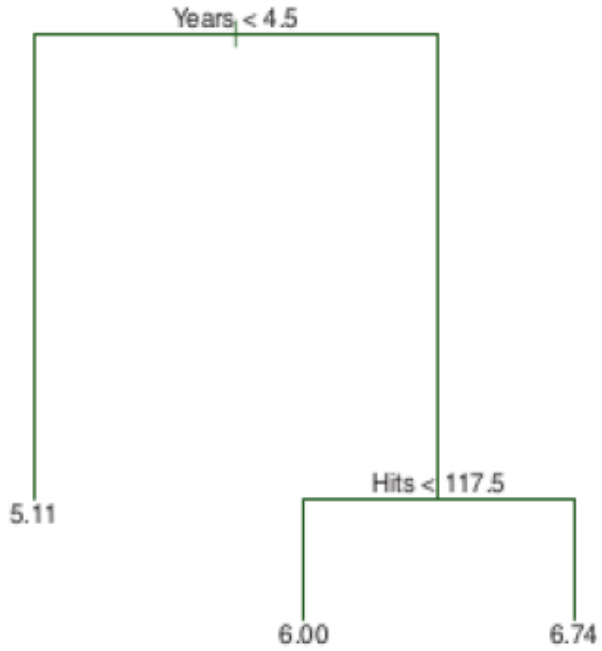
En diagnóstico médico: Un modelo que predice una alta probabilidad de enfermedad para un paciente que en realidad tiene una baja probabilidad de padecerla, lo que puede llevar a pruebas y tratamientos innecesarios

n pronósticos meteorológicos: Un modelo que sobrestima la cantidad de lluvia que caerá, lo que puede llevar a preparaciones excesivas

REGRESIÓN Y CLASIFICACIÓN CON ÁRBOLES DE DECISIÓN

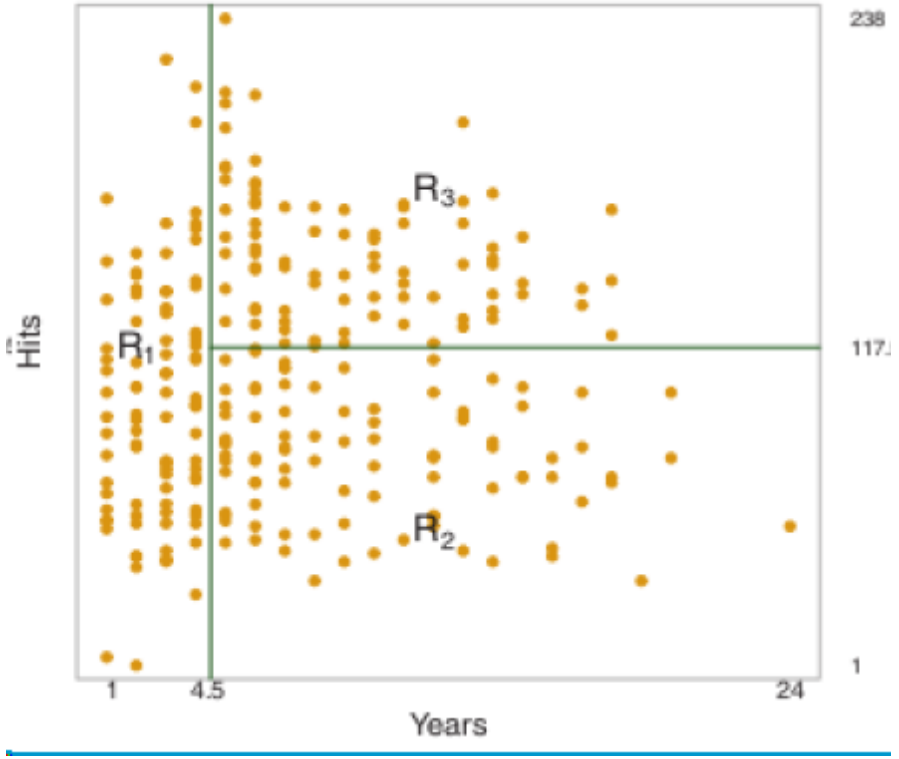
INTRODUCCIÓN

- Una de las técnicas más populares en data mining: permiten resolver problemas de regresión y clasificación.
- Dividen o segmentan el espacio de las variables predictoras en una serie de regiones.
- Para predecir una observación se utiliza la media o la moda de las observaciones que pertenecen a esa región
- Las reglas para separar las variables predictoras se resumen en forma de árbol, y se les conoce como árboles de decisión
- Se trata de métodos simples y fáciles de interpretar



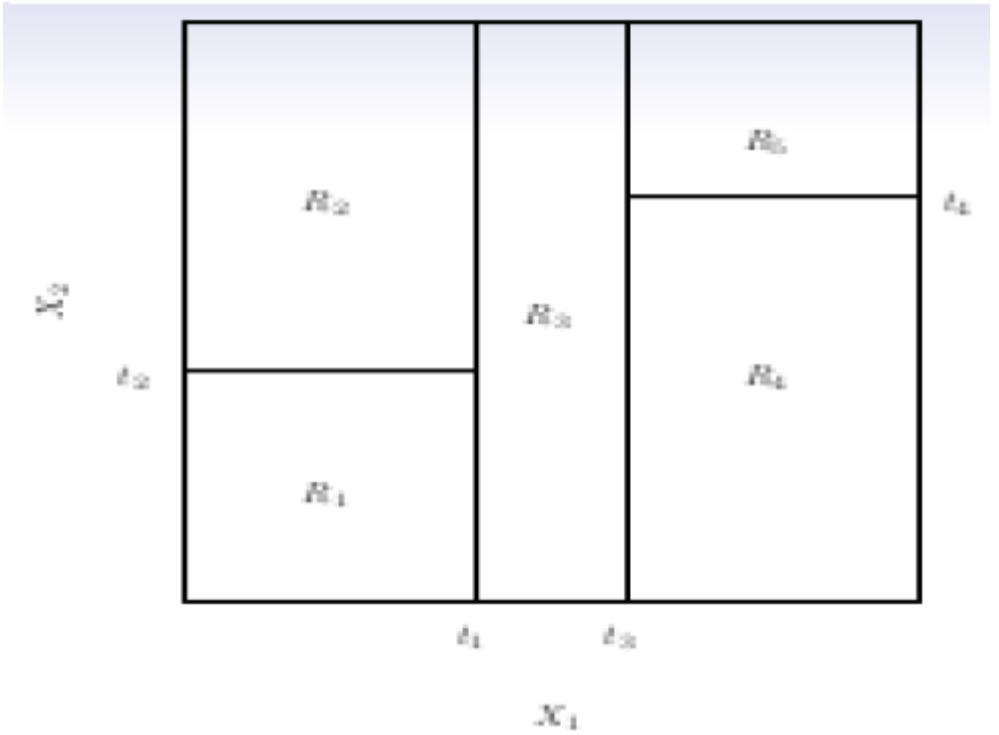
ÁRBOLES DE DECISIÓN

- Dividen el espacio en rectángulos en high-dimensión o boxes que minimizan el error de la predicción
- Es computacionalmente imposible considerar cualquier posible partición del espacio de entrada.
- Se utiliza un enfoque top-down greedy conocidocomo recursive binary splitting.



ALGORITMO

- En el nodo raíz se elige aquella variable que es más predictiva del target. Los ejemplos se dividen en grupos con distintos valores para esta clase
- El algoritmo continúa dividiendo los nodos con la elección de la mejor variable hasta que se alcance el criterio de parada:
  - Todos (casi todos) los ejemplos del nodo son de la misma clase.
  - No existen variables para distinguir entre los ejemplos.
  - El árbol ha alcanzado un tamaño predefinido.



MEJOR CORTE

- Sí los segmentos de una división contienen valores de una sola clase se consideran que es una división pura.
- Existen varias métricas de pureza para identificar los criterios de corte
- **Entropía**. Un valor de 0 indica que la muestra es completamente homogénea, mientras 1 indica desorden completo.

$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

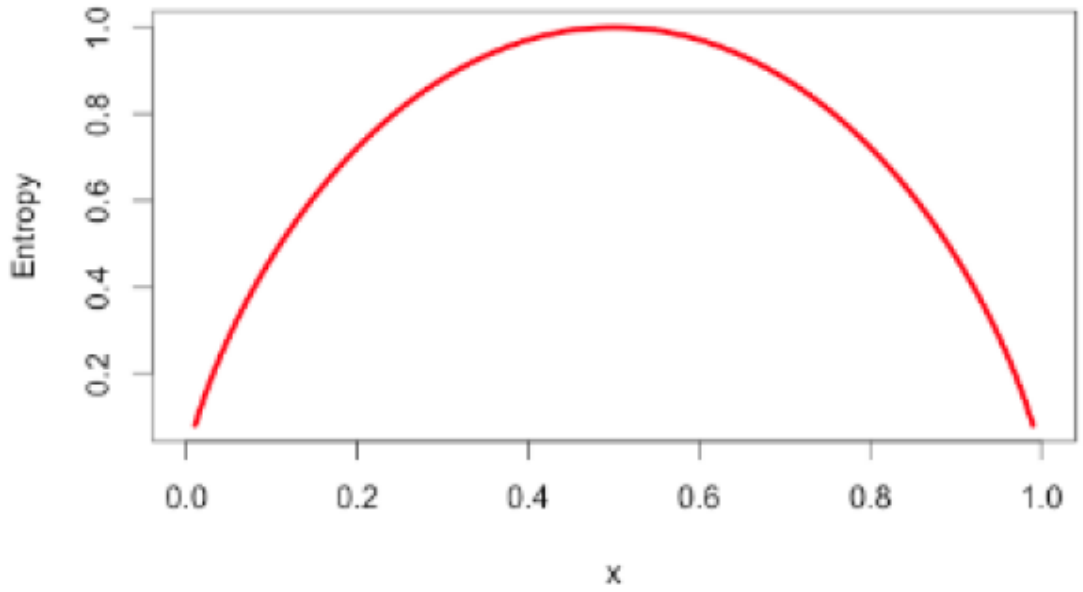
MEJOR CORTE

Por ejemplo si tenemos una partición de los datos en dos clases 60% y 40%.

```
> -0.60 * log2(0.60) – 0.40 * log2(0.40)
[1] 0.9709506
```

Si conocemos que la proporción de una clase es x, se puede examinar la entropía de todas las posibles divisiones de dos clases utilizando la función *curve()*

```
>curve(-x * log2(x) – (1 – x) * log2(1 – x), col = “red”, xlab = “x”, ylab = “Entropy”, lwd = 4)
```



MEJOR CORTE

- Con esta medida de pureza el algoritmo tiene que decidir con que variable hacer el corte. Se utiliza la entropía para calcular el cambio resultante de hacer el corte en esa variable.
- Se calcula la Information Gain (IG) que es la diferencia entre la entropía en el segmento antes de hacer el split (S1) y la partición resultante de hacer el split (S2).
- Después de un split los datos se pueden dividir en más de una partición, por tanto se considera la entropía a lo largo de las N particiones ponderando la entropía de cada partición con el número de instancias de esa partición

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \, \text{Entropy}(P_i)$$

El proceso top-down greedy puede generar buenas predicciones en el training set, pero también sufre de overfitting.

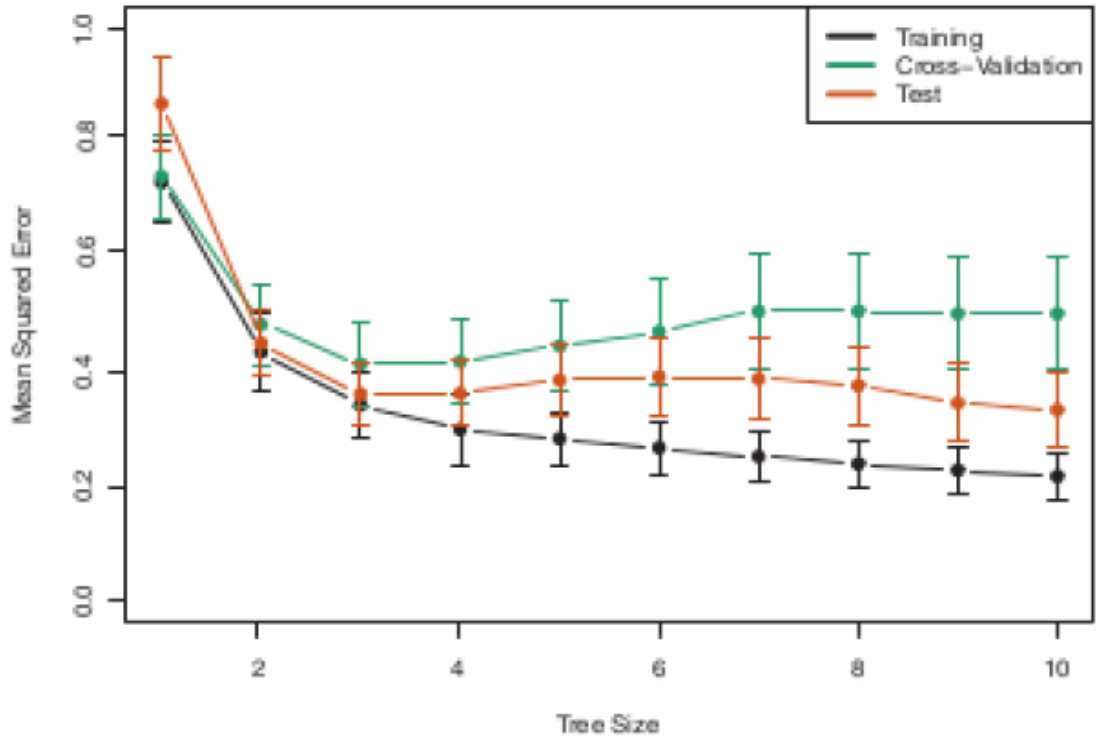
Esto es porque el árbol puede ser muy complejo. Un árbol más pequeño puede dar lugar a una menor varianza y mejor interpretación con el coste de un pequeño bias(sesgo).

La estrategia suele ser: generar un árbol muy grande y luego podarlo para obtener un sub-árbol. (post-prunning)

¿Como seleccionamos el sub-árbol?. Utilizando aquel que proporcione un menor error de test con cross-validation o bien en un conjunto de validación.

PODA

MEAN SQUAURED ERROR



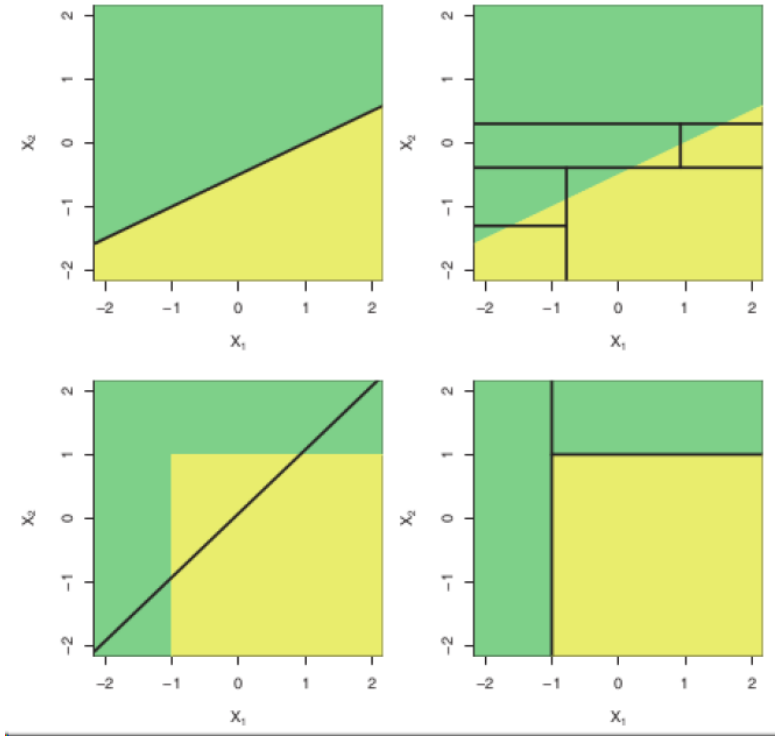
En árboles de regresión la predicción corresponde a la media de las observaciones de ese nodo terminal. En

clasificación con la clase que tiene mayor número de ocurrencias.

RESULTADO

ÁRBOLES VS. MODELOS LINEALES QUE MODELO ES MEJOR?

- Depende. Si las relaciones entre las variables y la variable respuesta se pueden aproximar bien por un modelo lineal, una regresión lineal funciona bien y supera a un árbol de regresión (que no puede modelar esta estructura).
- Sin embargo, sí el problema es no-lineal y con relaciones complejas entre las variables, los árboles funcionan mejor.



FORTALEZAS Y DEBILIDADES

- Clasificador de propósito general que se comporta bien en la mayoría de los problemas
- Utiliza solo las variables más importantes
- Se puede utilizar con pocos o muchos datos de entrenamiento
- Da como resultado un modelo que se puede interpretar sin conocimientos matemáticos
- Suelen estar sesgados a splits en variables que tienen muchos niveles
- Pequeños cambios en los datos de entrenamiento dan lugar a grandes cambios en la lógica de decisión

EJERCICIO

Definición de columnas del conjunto de datos

- instant: indice de registro
- dteday : fecha
- season : estacion (1: invierno, 2: primavera, 3: verano, 4: otoño)
- yr : año (0: 2011, 1:2012)
- mnth : mes (1 to 12)
- hr : hora (0 to 23)
- holiday : dia feriado (extracted from [Web Link])
- weekday : dia de la semana
- workingday : si el dia no es fin de semana ni feriado entonces es 1 caso contraio 0.
- weathersit :
  - 1: despejado, pocas nubes, parcialmente nublado, parcialmente nublado
  - 2: Niebla + Nublado, Niebla + Nubes rotas, Niebla + Pocas nubes, Niebla
  - 3: nieve ligera, lluvia ligera + tormenta eléctrica + nubes dispersas, lluvia ligera + nubes dispersas
  - 4: lluvia intensa + paletas de hielo + tormenta eléctrica + niebla, nieve + niebla
- temp : Temperatura normalizada en grados Celsius. Los valores se derivan a través de (t-t_min) / (t_max-t_min), t_min = -8, t_max = + 39 (solo en escala por hora)
- atemp: Temperatura de sensación normalizada en grados Celsius. Los valores se derivan a través de (t-t_min) / (t_max-t_min), t_min = -16, t_max = + 50 (solo en escala por hora)
- hum: Humedad normalizada. Los valores se dividen en 100 (máx.)
- windspeed: Velocidad del viento normalizada. Los valores se dividen en 67 (máx.)
- casual: recuento de usuarios ocasionales
- registered: recuento de usuarios registrados
- cnt: recuento del total de bicicletas de alquiler, incluidas las casuales y las registradas

LIBRERIAS NECESARIAS

```
import pandas as pd
import numpy as np

import warnings
warnings.filterwarnings("ignore")

from sklearn.tree import DecisionTreeRegressor, DecisionTreeClassifier
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn import metrics
from sklearn.model_selection import train_test_split
import pydot
from prettytable import PrettyTable
from sklearn.tree import export_graphviz
import graphviz
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

RECUPERACIÓN DE DATOS PASO 1

Recuperación de datos

```
datos = pd.read_csv('hour.csv')
```

```
datos.head()
```

ANÁLISIS DESCRIPTIVO PASO 2

- Para realizar analisis descriptivo separamos las columnas en categóricas y numéricas
- Se definen la columnas de variables (features)
- Se define la columna de clases (objetivo)

```
# columnas con valores categoricos
variables_categoricas = ['season', 'yr', 'mnth', 'hr', 'holiday', 'weekday', 'workingday', 'weathersit']
# columnas con valores numericos
variables_numericas = ['temp', 'atemp', 'hum', 'windspeed']
# lista de columnas de valores
variables = variables_categoricas + variables_numericas
# definicion de clases o variable objetivo
clases = ['cnt']
variables_clases = variables + clases
```

```
# Análsiis de variables numéricas
print(datos[variables_numericas].describe())
```

ANÁLISIS DESCRIPTIVO PASO 2

```
# Análsiis de variables categóricas
print(datos[variables_categoricas].astype('category').describe())
```

```
# Verificar valores missing
print(datos.isnull().any())
```

```
# Análsiis de clases
print(datos[clases].describe())
```

TRATAMIENTO DE DATOS PASO 3

Al analizar la variable a predecir cnt vemos que existe una distancia bastante grande entre el máximo y su cuartil 75, por lo que procederemos a realizar una eliminación de outtlers

```
# Rango intercuartilico 25
q1 = datos.cnt.quantile(0.25)
# Rango intercuartilico 75
q3 = datos.cnt.quantile(0.75)
# Diferencia entre rangos quartilicos
iqr = q3 - q1

# Definir el limite inferior
limite_inferior = q1 -(1.5 * iqr)
# Definir el limite superior
limite_superior = q3 +(1.5 * iqr)

# Filtrar los datos de acuerdo a limites inferior y superior
datos_preprocesados = pd.DataFrame(datos[variables_clases].loc[(datos.cnt >= limite_inferior) & (datos.cnt <= limite_superio
print("Ejemplos del grupo de entrenamiento con outliers: {}".format(len(datos)))
print("Ejemplos del grupo de entrenamiento sin outliers: {}".format(len(datos_preprocesados)))

# Desplegar la grafica
sns.distplot(datos.cnt)
sns.distplot(datos_preprocesados.cnt)
```

ANÁLSIS DE LA CORRELACIÓN DE VARIABLES PASO 3

```
# Construir matriz de correlacion
matrix = datos[variables_numericas + clases].corr()
heat = np.array(matrix)
heat[np.tril_indices_from(heat)] = False
# Construir mapa de calor
fig,ax= plt.subplots()
fig.set_size_inches(20,10)
sns.heatmap(matrix, mask=heat,vmax=1.0, vmin=0.0, square=True,annot=True, cmap="Blues")
```

ENTRENAMIENTO Y EVALUACIÓN

```python
# Conjunto de valores
valores = datos_preprocesados[variables_procesar]
nombre_columnas_valores = datos_preprocesados[variables_procesar].columns.values

# Conjunto de Clases
valores_clases = datos_preprocesados[clases]

# Dividir en Conjuntos de Valores y Clases en Conjuntos de Entrenamiento y Test
X_train, X_test, y_train, y_test = train_test_split(valores, valores_clases, test_size=0.20,random_state=0)


# Se define la profundidad del árbol basándose en el error cuadrático medio,
max_profundidad = 15

# Se Crea el modelo utilizando la entropía para la definición del split
Modelo = DecisionTreeRegressor(max_depth=max_profundidad)

# Evaluar el modelo
evaluar_metrica_regresion(Modelo, X_train, y_train, X_test, y_test)
```

```
Máxima profundidad definida: 14
+-----------------------+-------+-------+----------+
|         Model         |  RMSE | RMSLE | R² score |
+-----------------------+-------+-------+----------+
| DecisionTreeRegressor | 53.24 |  0.44 |   0.88   |
+-----------------------+-------+-------+----------+
```

EVALUACIÓN

```python
def evaluar_metrica_regresion(estimador, X_train, y_train, X_test, y_test):
    """
    Función Para evaluar las métricas en algoritmos de regresión
    Inputs:
        estimador: Modelo estimador a evaluar
        X_train: Conjunto de Valores de Entrenamiento
        y_train: Conjunto de Clases de Entrenamiento
        X_test: Conjunto de Valores de Test
        y_test: Conjunto de Clases de Test
    """

    # Construir una Tabla
    table = PrettyTable()
    # Definir nombre de columnas de la tabla
    table.field_names = ["Model", "RMSE", 'RMSLE', "R² score"]
    # Entrenar el modelo
    estimador.fit(X_train, y_train)
    # Realizar prediccion
    y_pred = estimador.predict(X_test)
    # Obtener Metrica RMSE
    rmse = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
    # Obtener Metrica Score
    score = estimador.score(X_test, y_test)
    # Obtener Metrica RMSLE
    rmsle = np.sqrt(metrics.mean_squared_log_error(y_test, y_pred))
    # Agregar a la tabla los valores de las métricas
    table.add_row([type(estimador).__name__, format(rmse, '.2f'), format(rmsle, '.2f'), format(score, '.2f')])
    # imprimir resultados
    print(table)


    # Grafica Diagrama de Dispersion x=reales, y=prediccion
    plt.figure(figsize=(10,8))
    plt.scatter(x=y_test, y=y_pred, marker='o', c='b', edgecolors=(0, 0, 0), alpha=0.5)
    plt.title('Dispersion {}'.format(type(estimador).__name__))
    plt.ylabel('Predichos')
    plt.xlabel('Reales')
    plt.legend()
    plt.show()
```

PRÁCTICA

- Generar una función para determinar el número de árboles óptimo

REGRESIÓN CON **RANDOM FOREST**

INTRODUCCIÓN

- Uno de los inconvenientes de los árboles de decisión o regresión (poca precisión) se puede solventar utilizando un **ensemble de árboles**.
- Los árboles se pueden combinar utilizando las técnicas de Bagging o Boosting (selección de variables aleatoria)

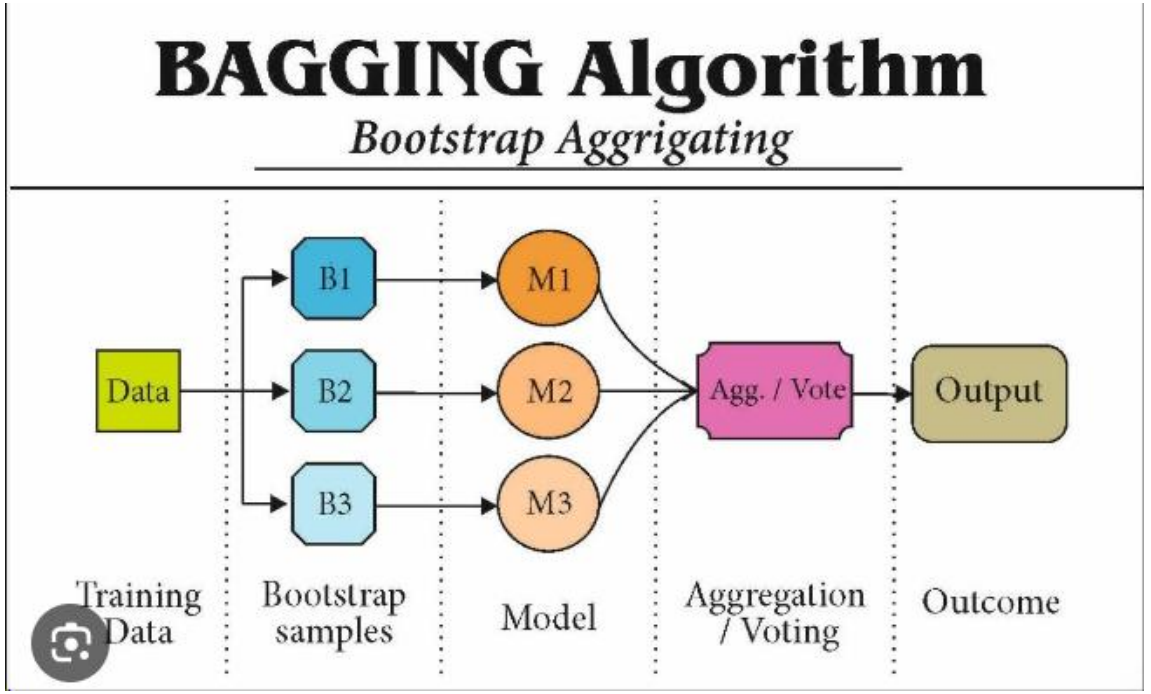
INTRODUCCIÓN

Para problemas de clasificación: en cada observación de test se almacena la clase predicha por cada uno de los B árboles y se obtiene un voto de la mayoría. La predicción global es la clase que más veces ocurre a lo largo de las B predicciones.

En el caso de los problemas de regresión, la predicción global es la media de las predicciones de cada uno de los árboles

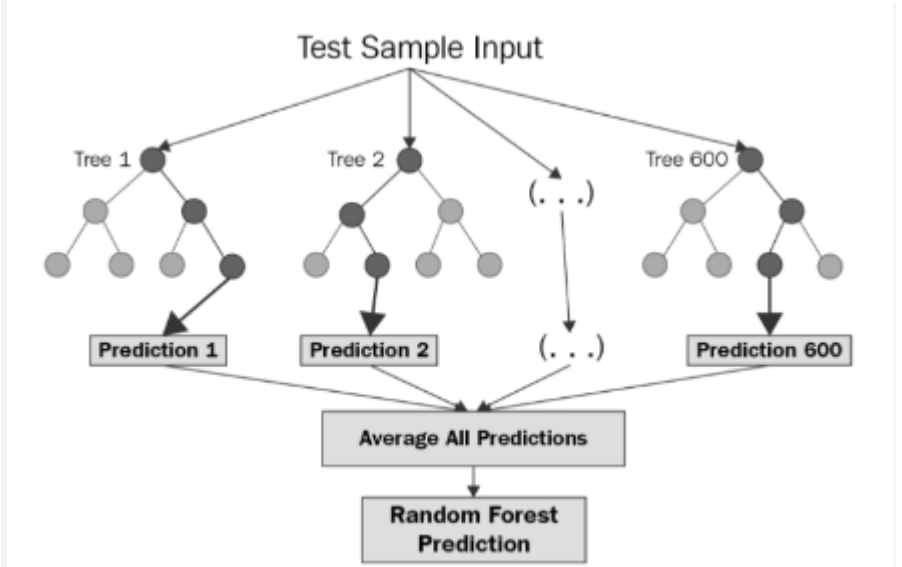
RANDOM FORESTS

- El modelo de random forests combina los principios de bagging con selección de variables aleatorias para añadir diversidad a los árboles de decisión. Una vez es generado el ensemble de árboles (forest) el modelo utiliza el mecanismo de votación o la media para generar las predicciones



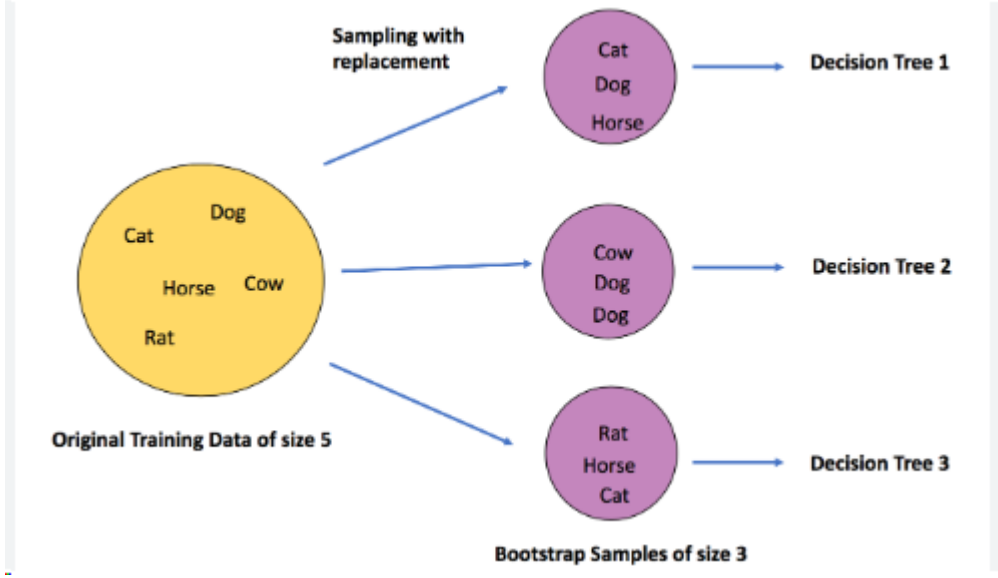
RANDOM FORESTS

- Este modelo se basa en la utilización de un gran número de árboles. La razón de utilizar un gran número de árboles es para que cada variable, de entre todas las posibles, tenga la oportunidad de aparecer en varios modelos



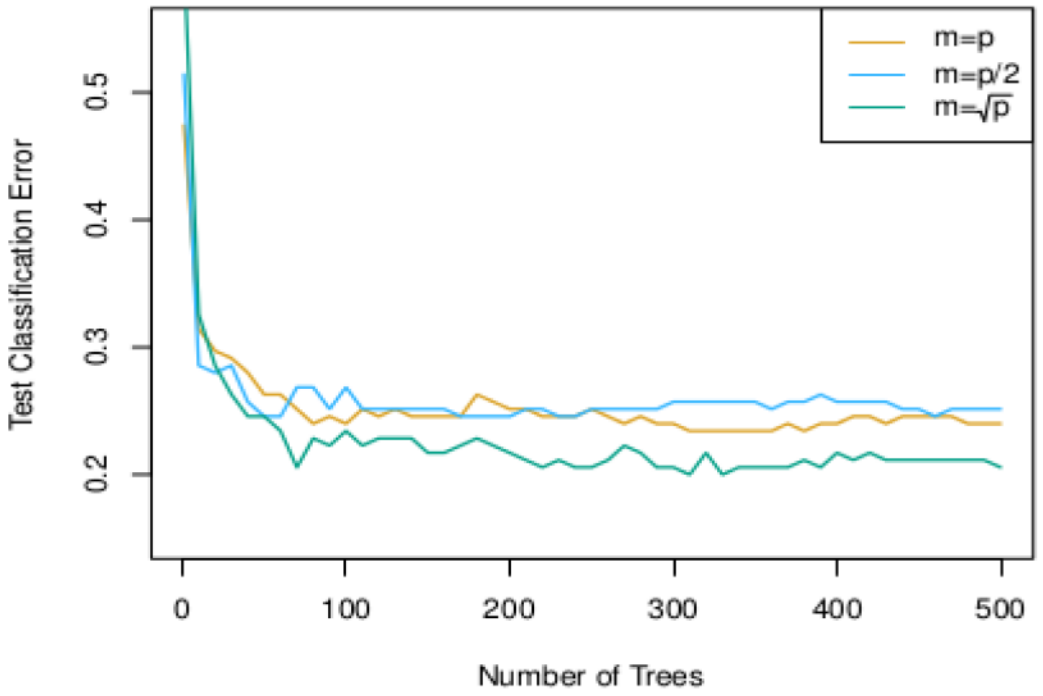
ERROR OUT OF BAG (OOB)

- Existe una forma sencilla de estimar el error de test en unmodelo bagged.
- Se puede demostrar que en media cada bagged tree hace uso de 2/3 de las observaciones.
- El 1/3 restante de las observaciones que no se utilizan para ajustar un bagged tree se llaman observaciones out-
- of-bag (OOB).
- Para cada observación se puede predecir la respuesta de los bagged trees donde era OOB. Esto proporciona
- alrededor de B/3 predicciones para la observación i-ésima.
- Si B es grande, está estimación es equivalente al Leave One Out cross-validation error



NÚMERO DE ÁRBOLES E IMPORTANCIA DE VARIABLES

- Los modelos basados en random forest tiene dos parámetros para optimizar
  - Número de árboles
  - Número de variables que se evalúan en cada split



PRACTICA

Calidad del vino

Donado el 6/10/2009

Se incluyen dos conjuntos de datos relacionados con muestras de vino verde tinto y blanco del norte de Portugal. El objetivo es modelar la calidad del vino basándose en pruebas fisicoquímicas (véase [Cortez et al., 2009],...

Características del conjunto de datos
Multivariante

Área temática
Negocio

Tareas asociadas
Clasificación, Regresión

Tipo de característica
Real

 # Instancias
4898

 # Características
11

GRACIAS

ALGUIEN@EJEMPLO.COM