Exposición del 14 y 16 de septiembre del 2021 del curso Machine learning.

Brenda Pamela Pérez Amézcua y José Miguel Calderón León.

PCCM

20 de septiembre de 2021

Naive Bayes Classifiers

Definición

Los clasificadores Naive Bayes son una familia de clasificadores que son bastante similares a los clasificadores lineales.

- Tienden a ser incluso más rápidos en entrenamiento.
- Su rendimiento es ligeramente peor que el rendimiento de los clasificadores lineales como LogisticRegression y LinearSVC

Naive Bayes Classifiers

Definición

Los clasificadores Naive Bayes son una familia de clasificadores que son bastante similares a los clasificadores lineales.

- Tienden a ser incluso más rápidos en entrenamiento.
- Su rendimiento es ligeramente peor que el rendimiento de los clasificadores lineales como LogisticRegression y LinearSVC

Naive Bayes Classifiers

La razón por la que los modelos Bayes son tan eficientes es que aprenden los parámetros al observar cada característica individualmente y recopilar estadísticas simples por cada característica de cada clase.

El teorema de Bayes:

$$P(C|F_1, ...F_n) = \frac{P(C)P(F_1, ..., F_n|C)}{P(F_1, ..., F_n)}$$

El clasificador Bayesiano optimo es clasificar x en la población/clase k, si

$$P_k(x) \geq P_i(x), \forall i$$

esto es clasificar x en la población con la máxima probabilidad de pertenecer.

Naive Bayes Classifiers

La razón por la que los modelos Bayes son tan eficientes es que aprenden los parámetros al observar cada característica individualmente y recopilar estadísticas simples por cada característica de cada clase.

El teorema de Bayes:

$$P(C|F_1, ...F_n) = \frac{P(C)P(F_1, ..., F_n|C)}{P(F_1, ..., F_n)}$$

El clasificador Bayesiano optimo es clasificar x en la población/clase k, si

$$P_k(x) \geq P_i(x), \forall i$$

esto es clasificar x en la población con la máxima probabilidad de pertenecer.

Tipos de Naive Bayes Classifiers

Hay tres tipos de clasificadores Bayes:

- GaussianNB: se puede aplicar a cualquier conjunto de datos continuo, Este tipo almacena el valor promedio así como la desviación estándar de cada característica para cada clase
- BernoulliNB: se aplica a datos binarios.
- MultinomialNB se aplica a datos de recuento (es decir, que cada característica representa un recuento entero de algunos cosa, como la frecuencia con la que aparece una palabra en una oración). Tiene en cuenta la valor promedio de cada característica para cada clase

BernoulliNB y MultinomialNB se utilizan principalmente en la clasificación de datos de un texto.
El clasificador BernoulliNB cuenta la frecuencia con la que cada característica de cada clase es no-cero.

Tipos de Naive Bayes Classifiers

Hay tres tipos de clasificadores Bayes:

- GaussianNB: se puede aplicar a cualquier conjunto de datos continuo, Este tipo almacena el valor promedio así como la desviación estándar de cada característica para cada clase
- BernoulliNB: se aplica a datos binarios.
- MultinomialNB se aplica a datos de recuento (es decir, que cada característica representa un recuento entero de algunos cosa, como la frecuencia con la que aparece una palabra en una oración). Tiene en cuenta la valor promedio de cada característica para cada clase

BernoulliNB y MultinomialNB se utilizan principalmente en la clasificación de datos de un texto.
El clasificador BernoulliNB cuenta la frecuencia con la que cada característica de cada clase es no-cero.

Tipos de Naive Bayes Classifiers

Hay tres tipos de clasificadores Bayes:

- GaussianNB: se puede aplicar a cualquier conjunto de datos continuo, Este tipo almacena el valor promedio así como la desviación estándar de cada característica para cada clase
- BernoulliNB: se aplica a datos binarios.
- MultinomialNB se aplica a datos de recuento (es decir, que cada característica representa un recuento entero de algunos cosa, como la frecuencia con la que aparece una palabra en una oración). Tiene en cuenta la valor promedio de cada característica para cada clase

BernoulliNB y MultinomialNB se utilizan principalmente en la clasificación de datos de un texto.
El clasificador BernoulliNB cuenta la frecuencia con la que cada característica de cada clase es no-cero.

Tipos de Naive Bayes Classifiers

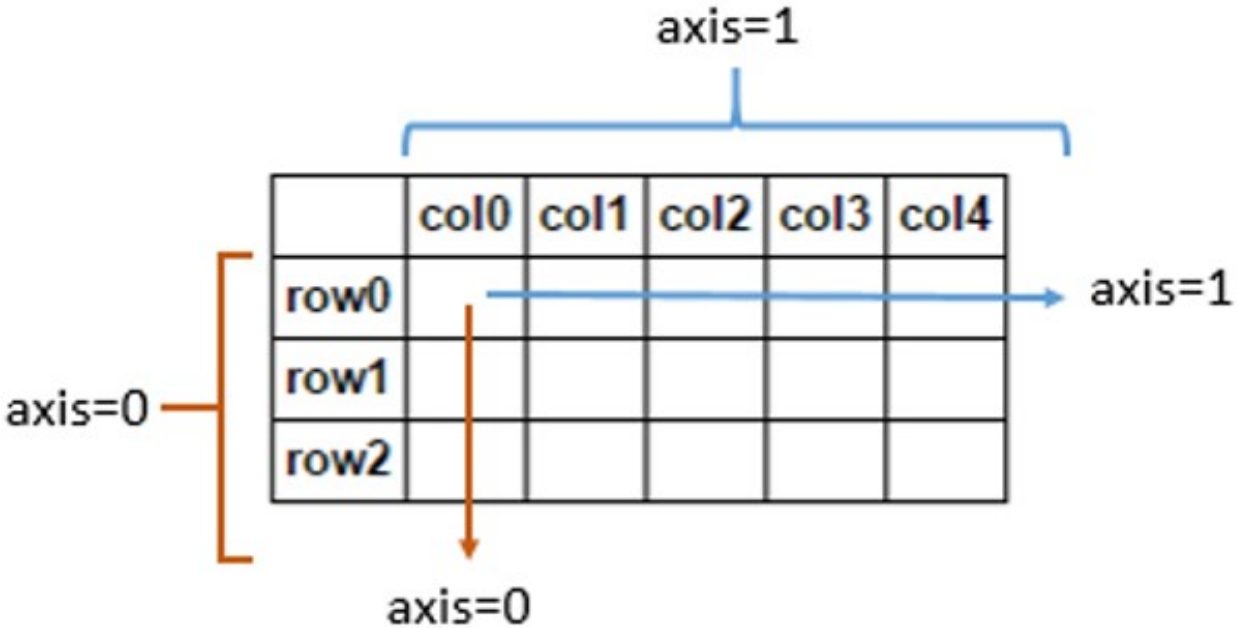
Hay tres tipos de clasificadores Bayes:

- GaussianNB: se puede aplicar a cualquier conjunto de datos continuo, Este tipo almacena el valor promedio así como la desviación estándar de cada característica para cada clase
- BernoulliNB: se aplica a datos binarios.
- MultinomialNB se aplica a datos de recuento (es decir, que cada característica representa un recuento entero de algunos cosa, como la frecuencia con la que aparece una palabra en una oración). Tiene en cuenta la valor promedio de cada característica para cada clase

BernoulliNB y MultinomialNB se utilizan principalmente en la clasificación de datos de un texto.
El clasificador BernoulliNB cuenta la frecuencia con la que cada característica de cada clase es no-cero.

Ejemplo BernoulliNB

Función sum de Numpy

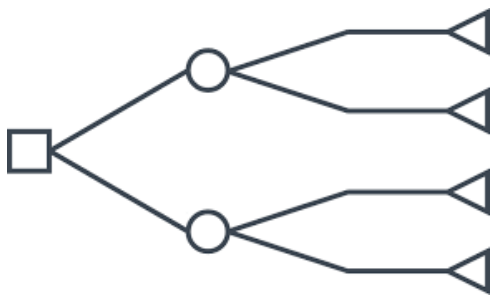


Parámetro de los tipos MultinomialNB y BernoulliNB

MultinomialNB y BernoulliNB tienen un solo parámetro, α, que controla complejidad del modelo. Un α grande significa más suavizado, lo que resulta modelos menos complejos. El rendimiento del algoritmo para la configuración de α es relativamente robusto. El α no es fundamental para un buen rendimiento, sin embargo, afinarlo generalmente mejora un poco la precisión.

Decision Trees

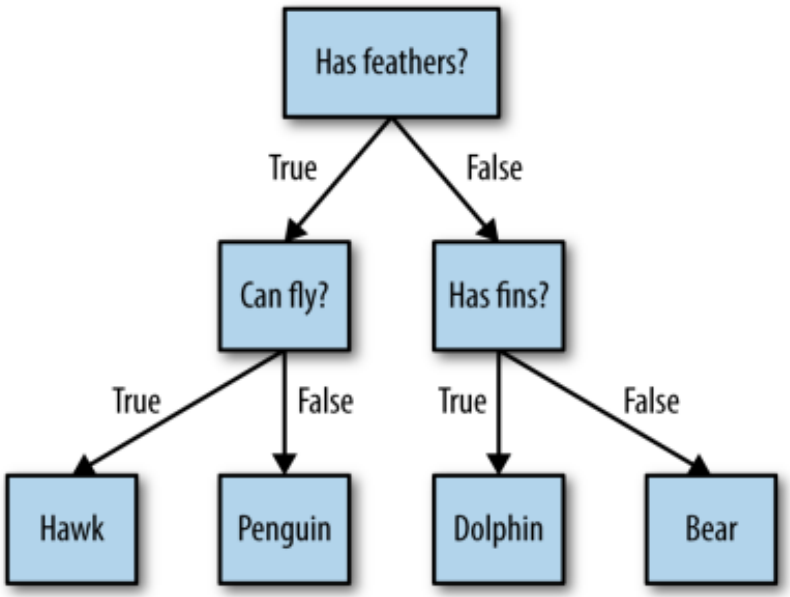
Los árboles de decisión son modelos ampliamente utilizados para tareas de clasificación y regresión, estos son diagramas de flujo que toman decisiones basadas en experiencias previas, para esto aprenden una jerarquía de preguntas if/else, lo que lleva a una decisión.



Ejemplo de Decision Trees

Imagina que quieres distinguir entre los siguientes cuatro animales: osos, halcones, pingüinos y delfines. Su objetivo es llegar a la respuesta correcta preguntando tan solo if / else

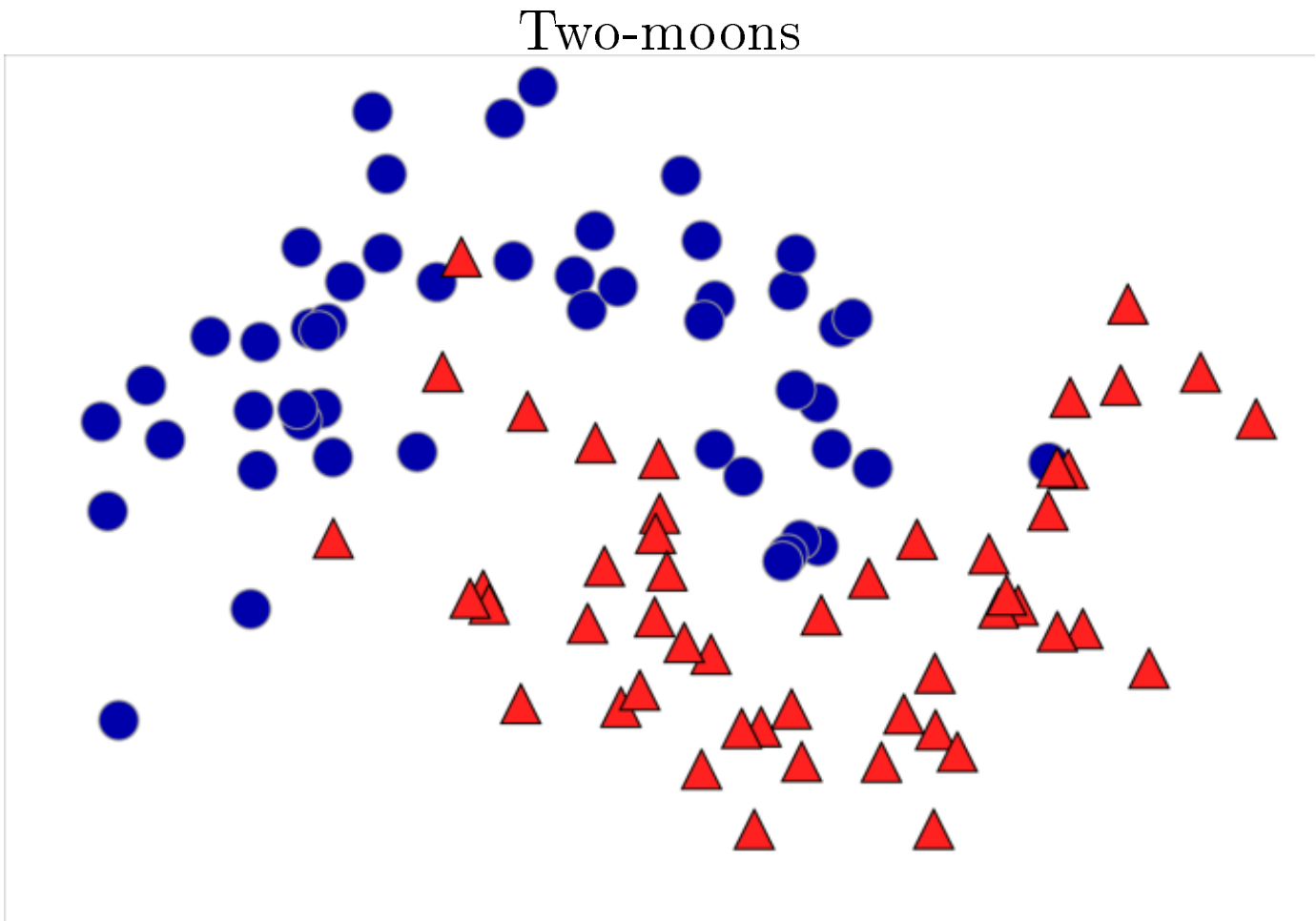
Árbol de animales



Construyendo árboles de decisión

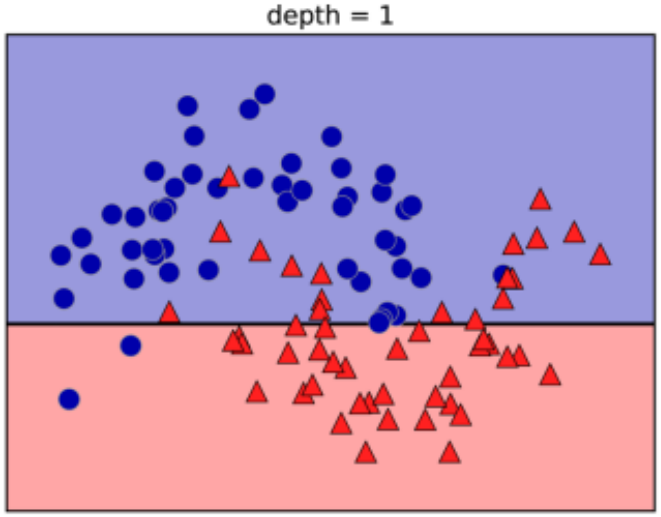
Repasemos el proceso de creación de un árbol de decisiones para los datos de clasificación 2D. El conjunto de datos consta de dos formas de media luna, cada clase consta de 75 datos. Nos referiremos a este conjunto de datos como two-moons El aprendizaje de un decision tree significa que aprenda la secuencia de preguntas if / else que nos lleva a la respuesta verdadera más rápidamente. En el entorno del aprendizaje automático, estas preguntas son llamadas pruebas

Ejemplo de decision tree



Ejemplo de decision tree

Two-moons



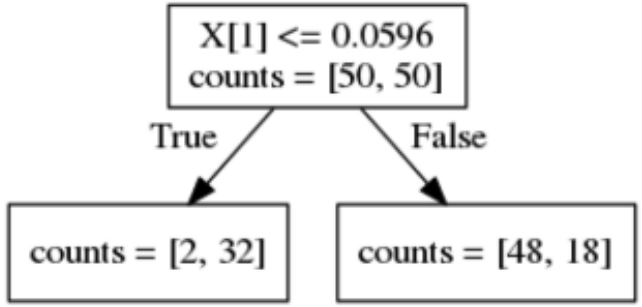
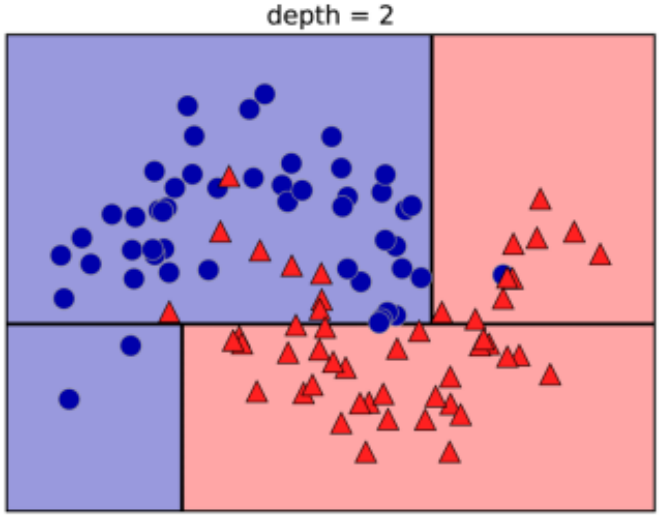
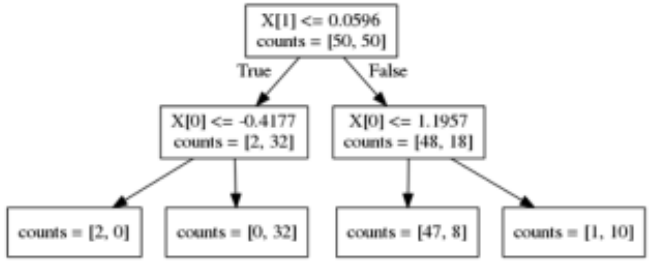


Figure 2-24. Decision boundary of tree with depth 1 (left) and corresponding tree (right)





Ejemplo de decision tree

Este proceso recursivo produce un árbol binario de decisiones, con cada nodo que contiene un prueba. Alternativamente, puede pensar en cada prueba como dividir la parte de los datos que actualmente se está considerando a lo largo de un eje.
La partición recursiva de los datos se repite hasta que cada región de la partición (cada hoja en el árbol de decisión) solo contiene un único valor objetivo (una sola clase o un valor de regresión único).

Ejemplo de decision tree

Two-moons

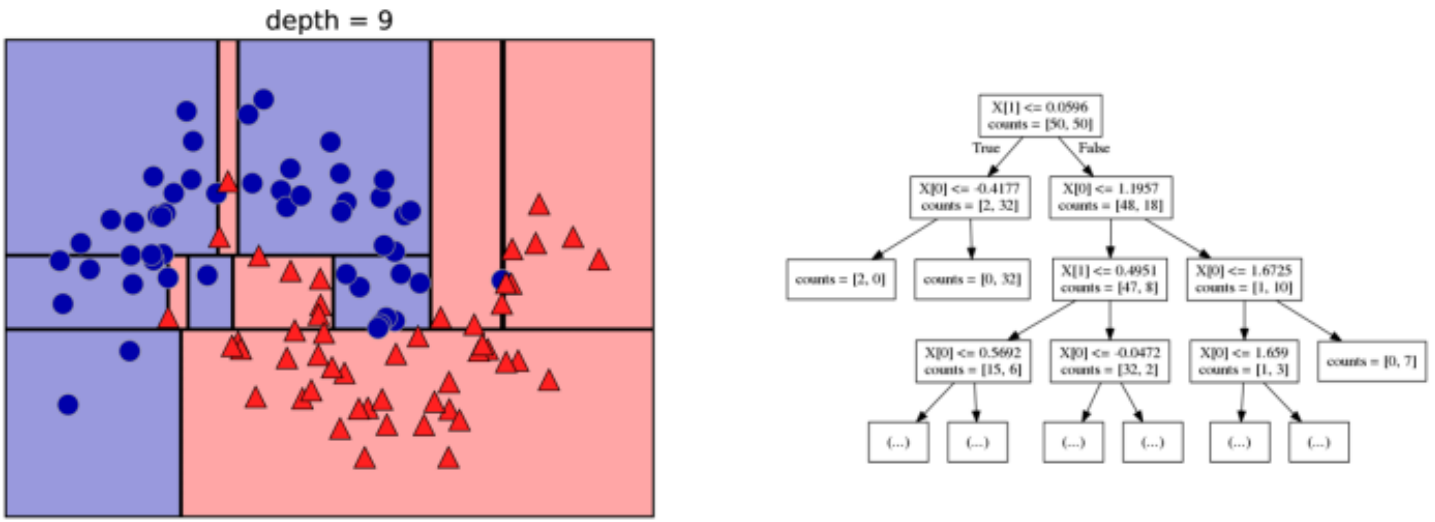


Figure 2-26. Decision boundary of tree with depth 9 (left) and part of the corresponding tree (right); the full tree is quite large and hard to visualize

Controlling complexity of decision trees

Por lo general, construir un árbol como se describe aquí y continuar hasta que todas las hojas estén puras. conduce a modelos que son muy complejos y muy sobreajustados a los datos de entrenamiento.

Evitar el sobreajuste

Hay dos estrategias comunes para evitar el sobreajuste: detener la creación del árbol temprano (también llamado poda previa), o construir el árbol pero luego quitarlo o colapsar nodos que contienen poca información (también llamado post-poda o simplemente poda). Los posibles criterios para la poda previa incluyen limitar la profundidad máxima del árbol, limitar el número máximo de hojas o exigir un número mínimo de puntos en un nodo para seguir dividiéndolo

Controlling complexity of decision trees

Por lo general, construir un árbol como se describe aquí y continuar hasta que todas las hojas estén puras. conduce a modelos que son muy complejos y muy sobreajustados a los datos de entrenamiento.

Evitar el sobreajuste

Hay dos estrategias comunes para evitar el sobreajuste: detener la creación del árbol temprano (también llamado poda previa), o construir el árbol pero luego quitarlo o colapsar nodos que contienen poca información (también llamado post-poda o simplemente poda). Los posibles criterios para la poda previa incluyen limitar la profundidad máxima del árbol, limitar el número máximo de hojas o exigir un número mínimo de puntos en un nodo para seguir dividiéndolo

ÁRBOLES DE DECISIÓN CON REGRESIÓN

Su funcionamiento es similar al de los árboles de decisión tradicionales, sin embargo, difieren en que no pueden predecir datos que se encuentren fuera del rango de los datos de entrenamiento.

Ejemplo: utilizaremos el data set "precio por año de memorias de computadora" para hacer predicciones usando árboles con regresión y regresión lineal.

EN RESUMEN…

Los árboles de decisión suelen tener ciertas ventajas

- El modelo resultante se visualiza y se entiende fácilmente
- Los algoritmos son invariantes si escalamos los datos

Y desventajas…

- Suelen "sobre-ajustar" los datos (fallan en la generalización)

Ya hemos visto alternativas para evitar el sobreajuste, como el podado de los árboles, próximamente revisaremos dos nuevas alternativas: Random Forest/ Gradient boosted decision trees.

RANDOM FOREST

Random Forest es un modelo de predicción que se basa en la construcción de varios árboles de decisión los cuales deben ser distintos entre sí e individualmente deben ser buenos prediciendo. Para poder construir árboles distintos, se necesitará el azar (de ahí el nombre *Random*).

Hay dos formas de involucrar al azar para construir árboles distintos:

1) Seleccionar un subconjunto de data points que participaran en la "construcción del árbol".
2) Seleccionar las características sobre las cuales se expandirá el árbol.

PASOS PARA CONSTRUIR UN RANDOM FOREST

Sea n el número de data points.

Decidir el número de árboles a construir

n_estimators

Construir cada árbol con n data points escogidos en forma aleatoria con reemplazo.

Cada nodo de cada árbol decide aleatoriamente un subconjunto de características sobre las cuales se tomarán decisiones.

max_features

Para realizar una nueva predicción de un dato x, cada árbol analiza la probabilidad que tiene x se estar en cada clase.

Se toma un promedio de las probabilidades por clase, y x se clasifica en la clase con el mayor promedio.

VARIACIÓN DE N_ESTIMATORS Y MAX_FEATURES

n_estimators

Se pierde complejidad en el modelo.

Se consume más tiempo y memoria. Se dificulta visualizar cada árbol individualmente. Aumenta la complejidad del modelo.

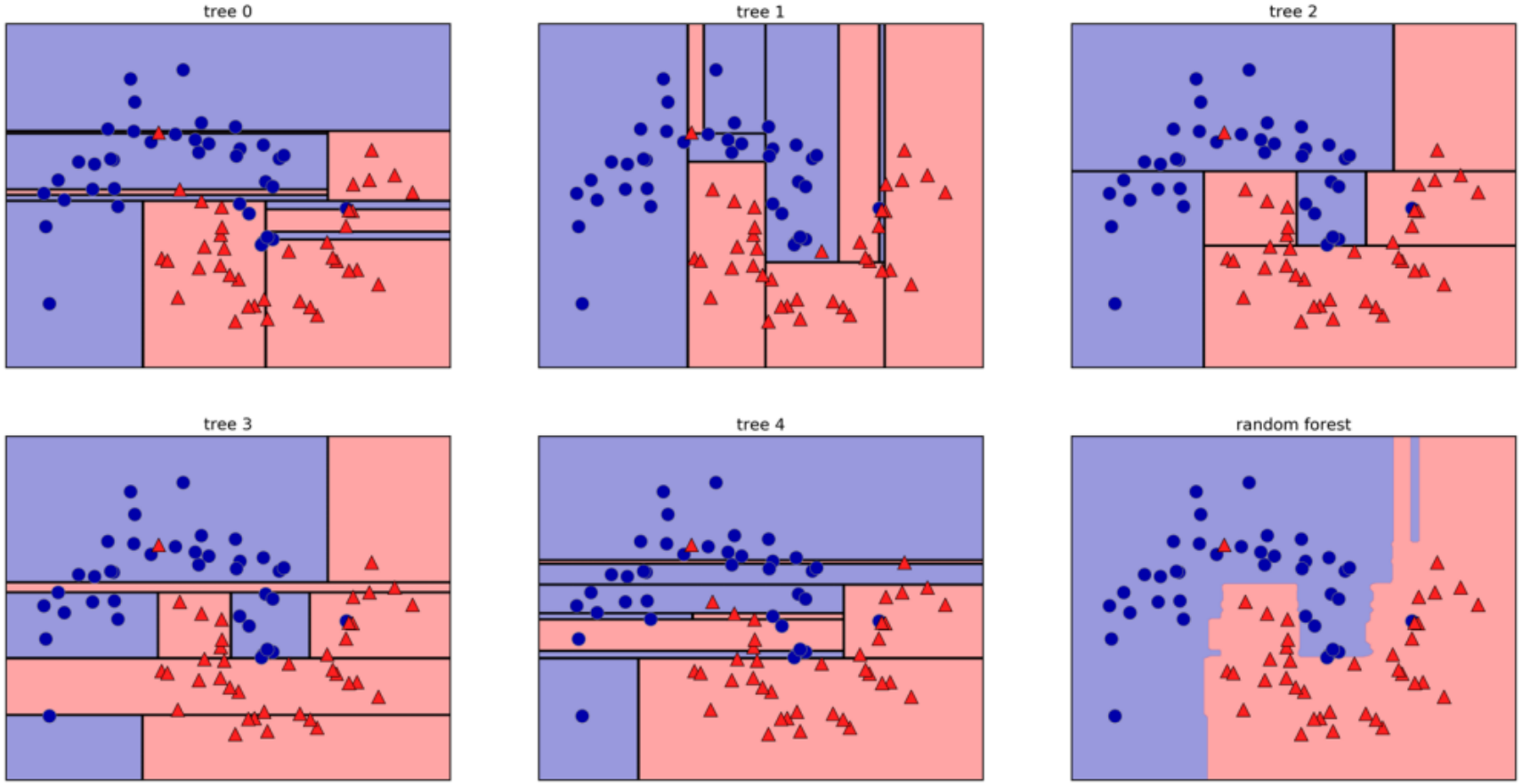
max_features

No le damos opciones a escoger, se necesitará profundidad para compensar.
Reduce el sobre ajuste (overfitting)

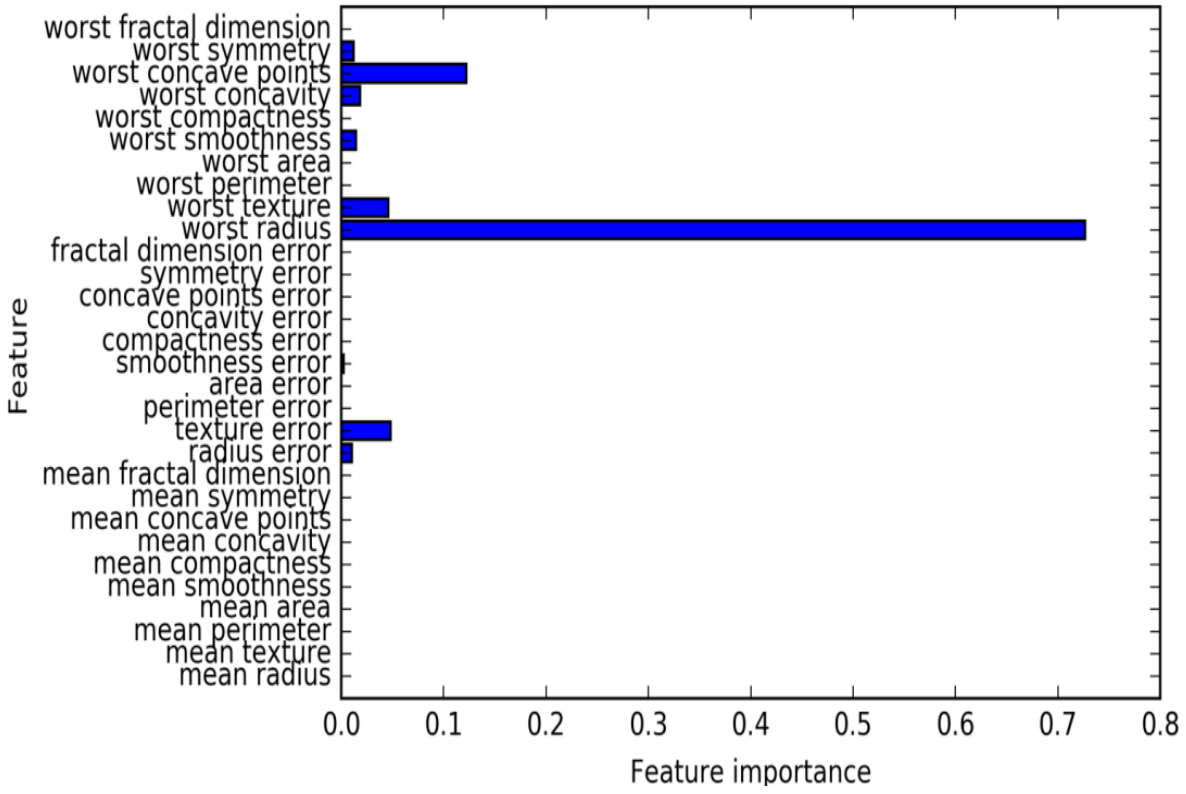
Se pierde la aleatoriedad. Aumenta la complejidad del modelo.

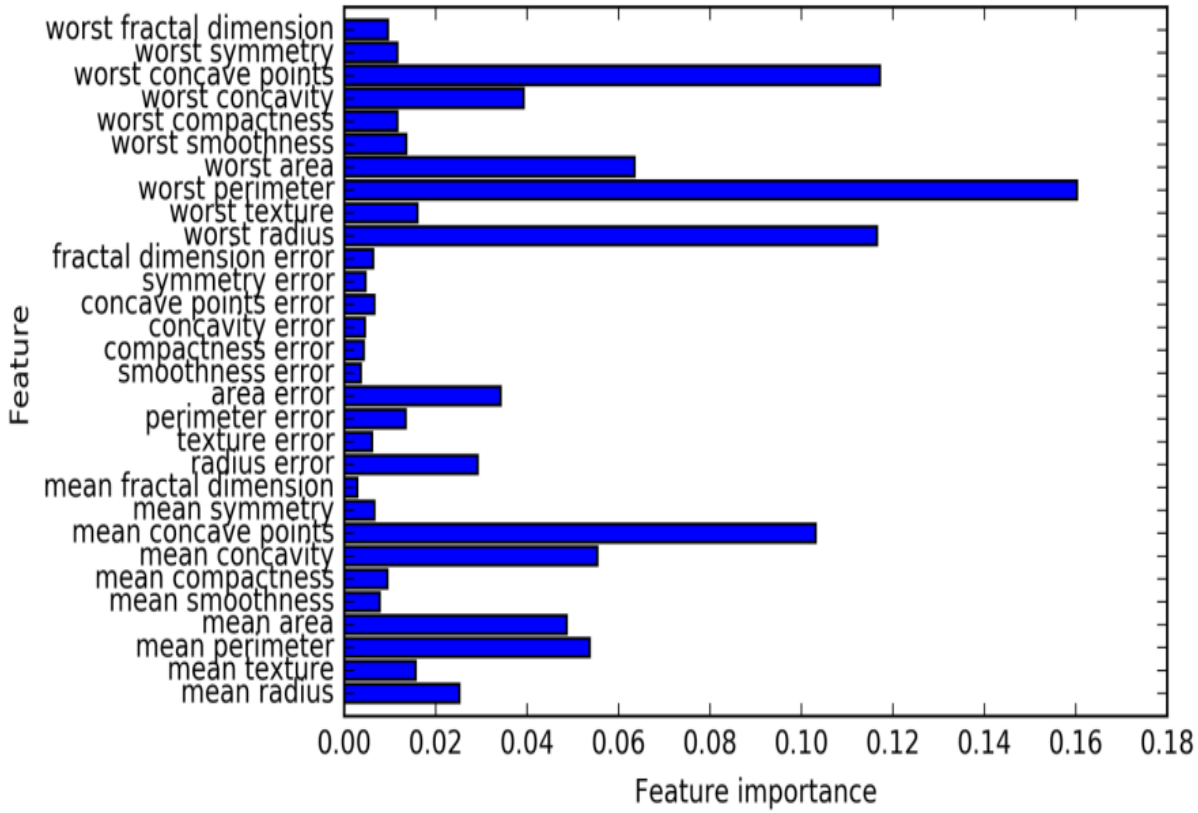
EJEMPLOS

Ejemplo 1: Las formas de la luna (data set: two_moons)



Ejemplo 2: Las clasificación de tumores (dataset: cancer)





GRADIENT BOOSTED REGRESSION TREES

Gradient boosted regression trees es un método, que al igual que el random forest, combina múltiples árboles de decisión para tomar decisiones. Este método construye árboles en secuencia donde cada sucesor trata de corregir los errores hechos por el árbol previo.

En vez de utilizar la aleatoriedad para tener un mejor desempeño, este método utiliza otras herramientas:

- Podado o pre-prunning (variable: max_depth)
- Número de árboles (variable n_estimators)
- Capacidad de aprendizaje (variable: learning_rate)

VARIACIÓN DE MAX_DEPTH, N_ESTIMATORS Y LEARNING_RATE

max_Depth

Se pierde complejidad en el modelo.

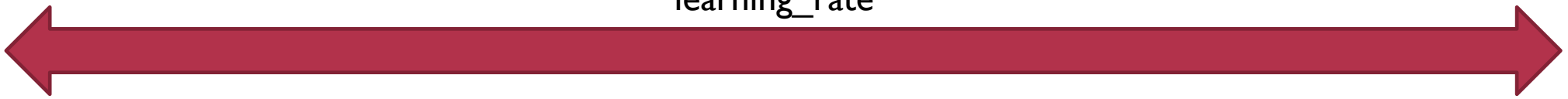
La cota superior en este método suele ser 5.

n_estimators

Se pierde complejidad en el modelo.

Mayor complejidad pero más overfitting

learning_rate



Más árboles tendrán que ser utilizados para obtener complejidad .

Mayor complejidad pero más overfitting

RESUMEN

El desempeño de los ramdom forest puede verse opacado por el tiempo de espera, una buena idea ante la optimización de tiempo, es aplicar Gradient boosted regression trees.

Así mismo, gradient boosted regression trees es un método sensible ante a variación de parámetros, por lo que un pequeño ajuste podría generar un cambio significativo en el modelo.

Ambos métodos cuentan con las mismas ventajas que tienen los árboles individuales (una de las más notorias: no es necesario escalar los datos) y además, es mejor su desempeño ya que cuentan con más herramientas para evitar el overfitting.