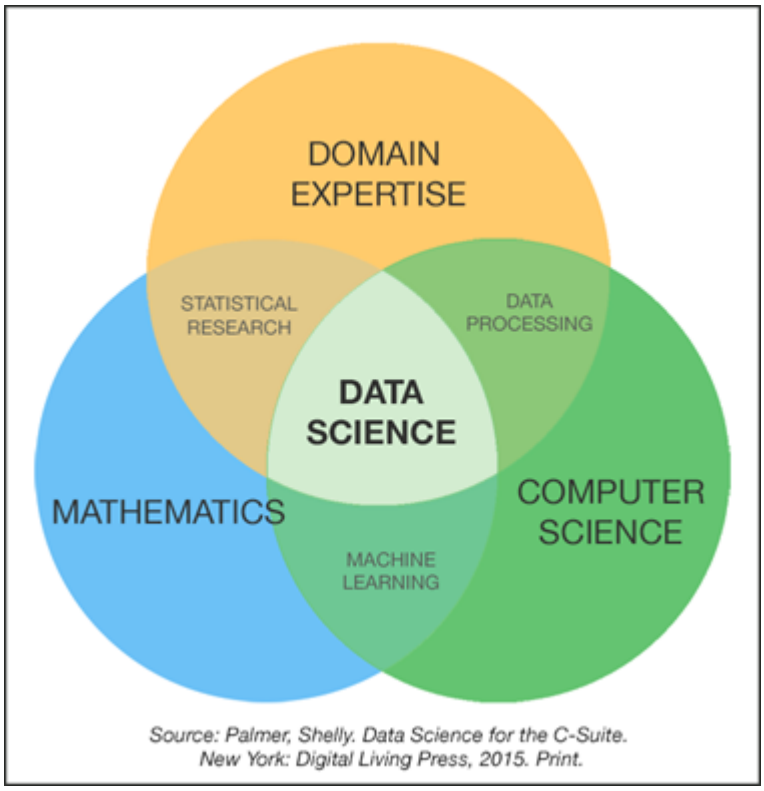
CTIC UNI
CENTRO DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIONES
UNIVERSIDAD NACIONAL DE INGENIERÍA

CIENCIA DE DATOS:
APRENDE LOS FUNDAMENTOS DE MANERA PRÁCTICA



SESION 04
APRENDIZAJE SUPERVISADO
ALGORITMOS ML – REDES NEURONALES

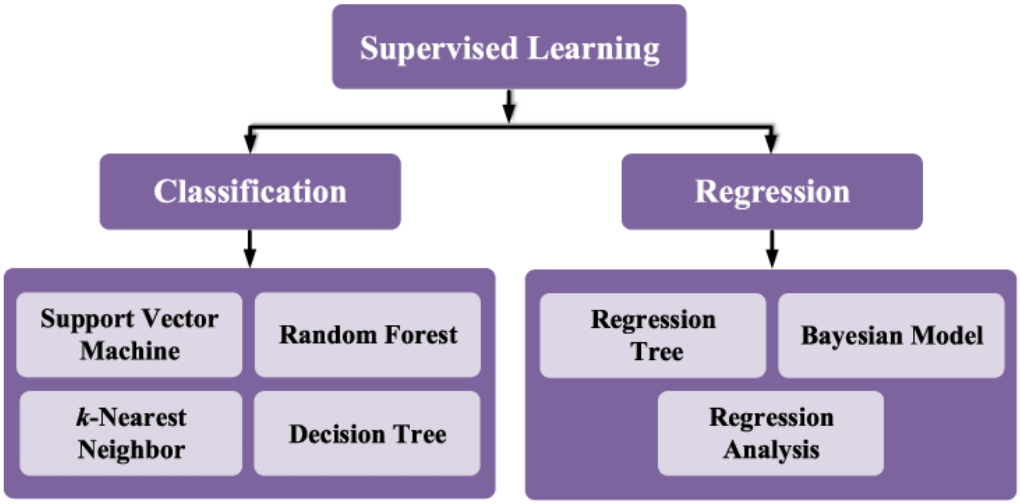
Juan Antonio Chipoco Vidal
jchipoco@gmail.com

ÍNDICE

Objetivo

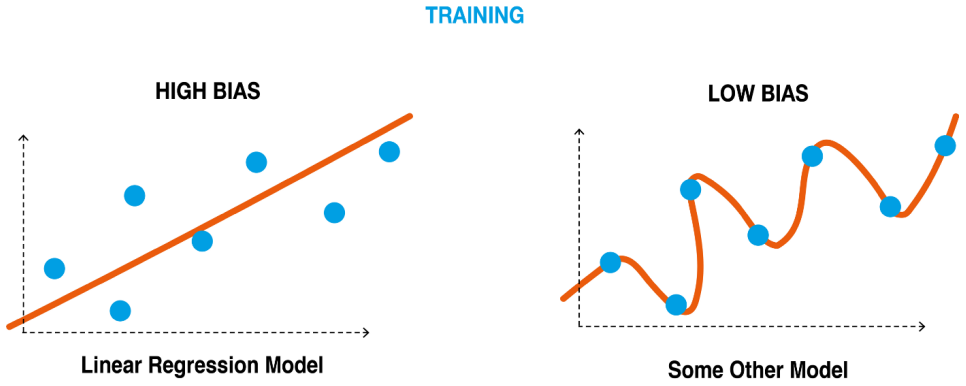
El objetivo de esta sesion es conocer los algoritmos de machine learning mas utilizados para el aprendizaje supervisado.

Revisaremos como funcionan los algoritmos de regresion asi como los algoritmos de clasificacion y luego procederemos a aplicarlos en nuestrar practica semanal.

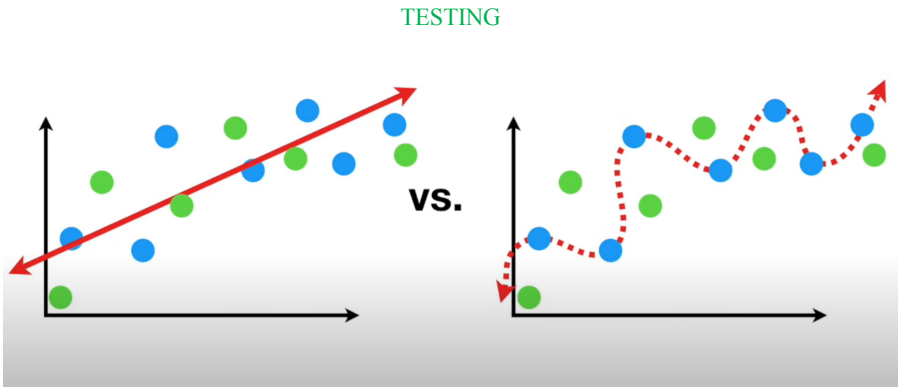


Bias Variance Trade Off

Recordemos que en el ***training set*** el bias es la incapacidad de un metodo de ML para capturar la relacion entre dos variables en este caso la linea recta (imagen izquierda). Al contrario con otro metodo de ML la curva ondulada (imagen derecha) encaja perfectamente en todos los puntos.



Sin embargo cuando vemos el ajuste del modelo para ver su comportamiento con el ***testing set*** obsevamos que la linea recta del primer metodo de ML ajusta mejor que el segundo metodo (linea ondulada).



En otras palabras sera dificil predecir como se comportara la linea ondulada con otros ***testing sets.*** A veces lo hara bien, otras veces lo hara mal. Tiene alta variabilidad.

El modelo con la linea ondulada tiene bajo bias pero alta varianza dado que la suma de sus residuos al cuadrado varian mucho entre diversos ***testing sets.***

El modelo con la linea recta tiene alto bias pero baja varianza dado que la suma de sus residuos al cuadrado varian poco entre diversos ***testing sets.***

Lo que debemos encontrar es un modelo que tenga bajo bias y baja varianza con el objeto que el modelo genere predicciones consistente con diversos ***testing sets.*** Para eso debemos encontrar una curva intermedia entre nuestros dos modelos. Esto se puede lograr utilizando *regularization, boosting o bagging*.

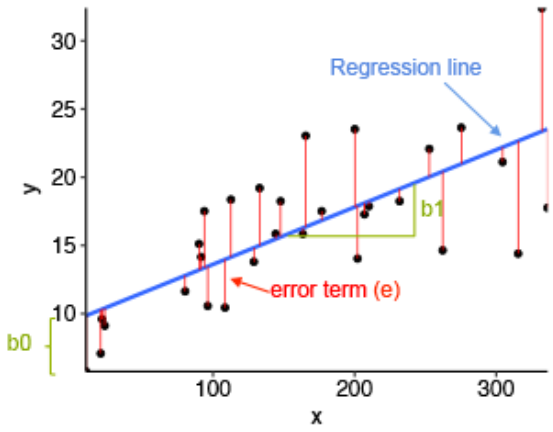
Regresion Lineal

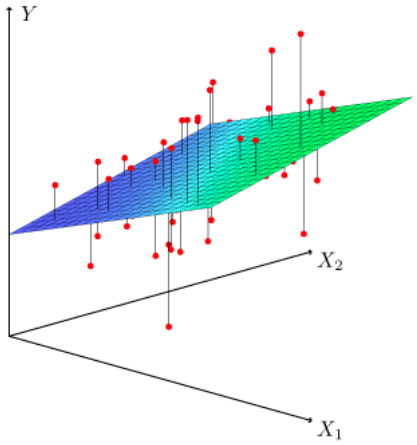
El análisis de regresión se utiliza para crear un modelo que describe la relación entre una variable dependiente y una o más variables independientes (*features*). Dependiendo de si hay una o más variables independientes, se hace una distinción entre análisis de regresión lineal simple y múltiple.

En el caso de una regresión lineal simple, el objetivo es examinar la influencia de una variable independiente sobre una variable dependiente. En el segundo caso, una regresión lineal múltiple, se analiza la influencia de varias variables independientes sobre una variable dependiente.

$$Y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n$$

Para esto se debe minimizar la distancia vertical entre todos nuestros datos y nuestra recta de regresion, la cual viene a ser nuestro modelo.





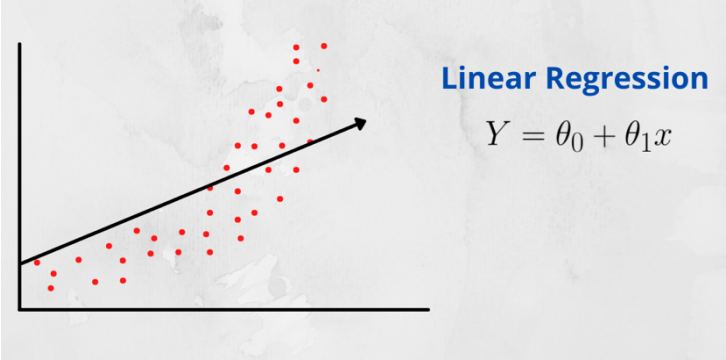
Debemos seleccionar las mejores variables independientes que puedan contribuir a la variable dependiente. Para esto, generamos la matriz de correlación para todas las variables independiente e incluimos la variable dependiente.

El valor de las correlaciones nos dara una idea de qué variables son significativas. A partir de esta informacion, seleccionamos las variables (*feature selection*) independientes en orden decreciente de valor de correlación, entrenamos y ejecutamos nuestro modelo de regresión para estimar los coeficientes minimizando la función de error. Cuando no veamos mejora en nuestro modelo dejamos de agregar o eliminar las variables independientes.

Por otra lado, tengamos en cuenta que si agregamos más variables independientes se crean relaciones entre ellas. Por lo que no solo las variables independientes estaran relacionadas con la variable dependiente, sino que también estaran relacionadas entre sí, esto se conoce como multicolinealidad. Todas las variables independientes se deberian correlacionar con la variable dependiente, pero no entre sí.

Regresion Polinomial

El algoritmo de regresión lineal simple, solo funciona cuando la relación entre los datos es lineal. Pero supongamos que tenemos datos no lineales, la regresión lineal no será capaz de dibujar una línea de mejor ajuste y falla en tales condiciones. Considere el siguiente diagrama que tiene una relación no lineal y puede ver los resultados de la regresión lineal en él. Por lo tanto, introducimos la regresión polinomial para superar este problema, esto ayudara a modelar una relación curvilínea entre las variables independientes y dependientes.

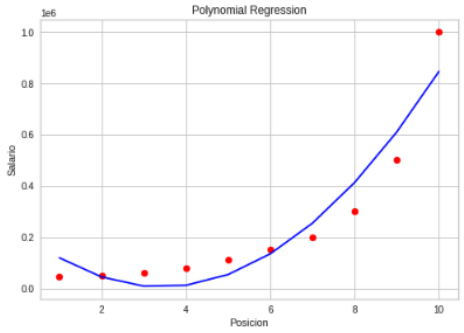


¿Cómo la regresión polinomial resuelve el problema de los datos no lineales?

La regresión polinomial es una forma de regresión lineal en la que debido a la relación no lineal entre las variables dependientes e independientes, procedemos a agregar algunos términos polinómicos a la regresión lineal para convertirla en una regresión polinomial.

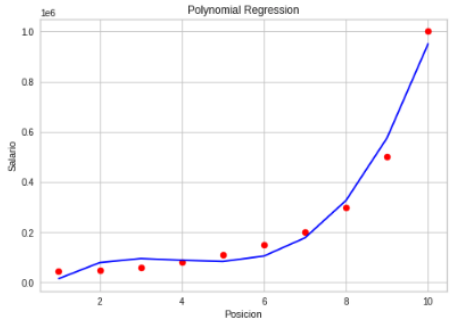
¿Por qué a la regresión polinomial se le llama regresión lineal polinomial?

Si observamos cuidadosamente la ecuación de regresión polinomial, veremos que estamos tratando de estimar la relación entre los coeficientes e y. Recordemos que los valores de x e y ya los tenemos en la data de entrenamiento, el entrenamiento nos determinara los coeficientes pero el grado de los coeficientes es 1, lo cual es una regresión lineal simple. Por lo tanto, la regresión polinomial también se conoce como regresión lineal polinomial.



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Regresion Polinomial



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$$

La regresion polinomial tiene tendencia a ajustarse excesivamente, hay evitar ajustar con polinomios de grados altos. Siempre visualizar la curva de ajuste y ver que esta se ajuste a la naturaleza del problema.

Support Vector Regression

En el aprendizaje automático, las *Support Vector Machine* son modelos de aprendizaje supervisado con algoritmos de aprendizaje asociados que analizan los datos utilizados para la clasificación y el análisis de regresión. En el caso de la regresion la *Support Vector Regression*, la línea recta que se requiere para ajustar los datos se denomina hiperplano.

La idea principal es minimizar el error, individualizando el hiperplano que maximiza el margen, teniendo en cuenta que se tolera parte del error.

A diferencia de los mínimos cuadrados ordinarios, el modelo SVR establece un límite de tolerancia de error ϵ alrededor de la línea de regresión de modo que todos los puntos de datos dentro de ϵ no sean penalizados por su error.

Los puntos de datos que quedan fuera del tubo ϵ se penalizan por su error:

El error asociado con un punto que esta sobre el ϵ-tube se calcula como la distancia vertical entre el punto y el margen superior del ϵ-tube. Se denota ξ_i*

Si el punto está debajo del tubo, el error es la distancia vertical entre el margen inferior del ϵ-tube y dicho punto. Se denota ξ_i

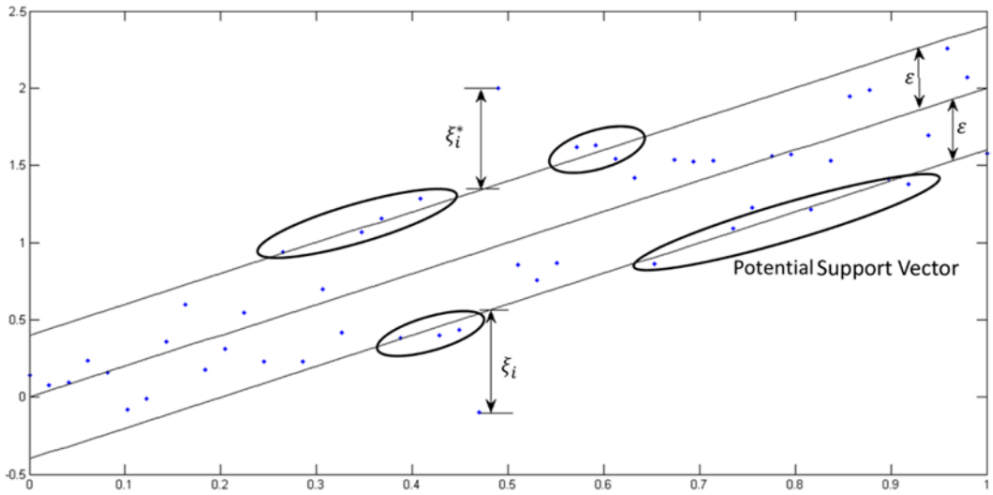
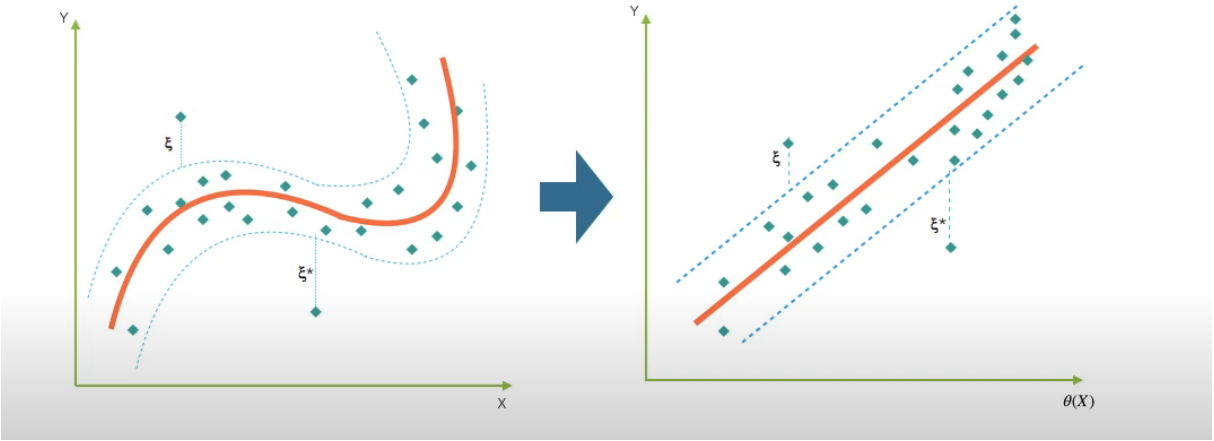


Image Source: Efficient Learning Machines Theories, Concepts, and Application for Engineers and System Designers by Mariette Awad and Rahul Khanna
Imagen utilizada solo con propositos educativos

Support Vector Regression

Para el caso no lineal se aplica el mismo procedimiento pero con la diferencia que se implementa un kernel.

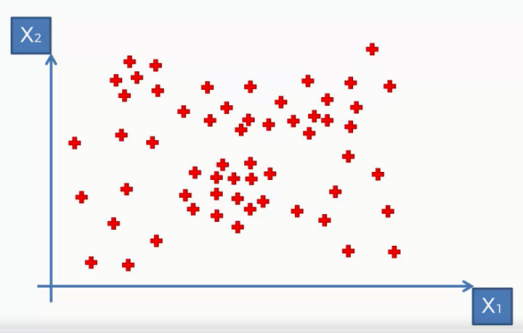


Decision Tree Regression

Los árboles de decisión son modelos predictivos formados por reglas binarias (si/no) con las que se consigue repartir las observaciones en función de sus atributos y predecir así el valor de la variable respuesta.

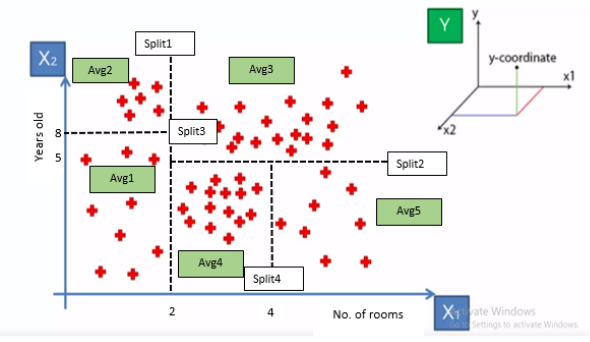
El algoritmo del árbol de decisiones cae dentro de la categoría de algoritmos de aprendizaje supervisado. Funciona tanto para variables de salida continuas como categóricas.

Los árboles de regresión son el subtipo de árboles de predicción que se aplica cuando la variable respuesta es continua. En términos generales, en el entrenamiento de un árbol de regresión, las observaciones se van distribuyendo por bifurcaciones (nodos) generando la estructura del árbol hasta alcanzar un nodo terminal. Cuando se quiere predecir una nueva observación, se recorre el árbol acorde al valor de sus predictores hasta alcanzar uno de los nodos terminales. La predicción del árbol es la media de la variable respuesta de las observaciones de entrenamiento que están en ese mismo nodo terminal.



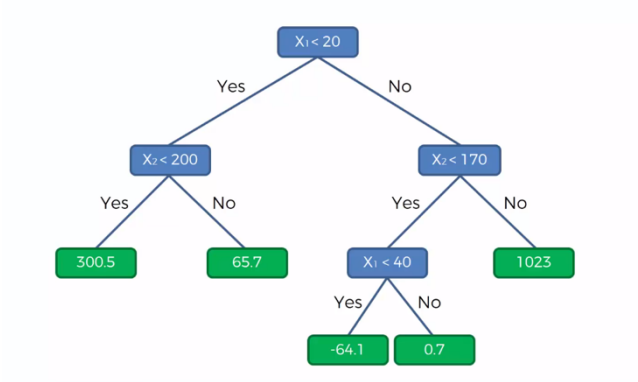
Aquí x1 y x2 son variables independientes e y se llama variable dependiente. Por ejemplo, si queremos predecir los costos de vivienda de diferentes localidades, podemos tomar x1 como el número de habitaciones y x2 como la antigüedad de la casa en años. Se considera que estas dos variables dependientes afectan el precio, "y".

El siguiente gráfico de dispersión muestra el gráfico 2D de x1 y x2, y y es una tercera dimensión, por ejemplo, la fijación de precios. En Decision Tree Regression, primero subdividimos el gráfico en varias divisiones llamadas virtualmente hojas. Formando así las llamadas hojas del árbol.



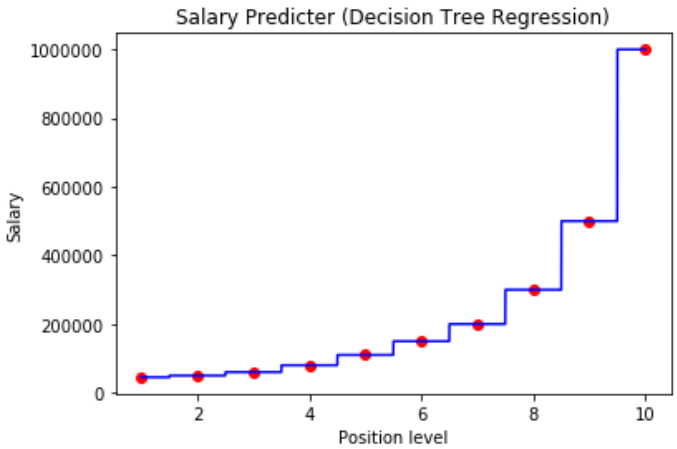
Decision Tree Regression

El algoritmo decide el número de divisiones teniendo en cuenta el valor de la información añadida al realizar la división. Por lo tanto, se construye un árbol de decisión de la siguiente manera:

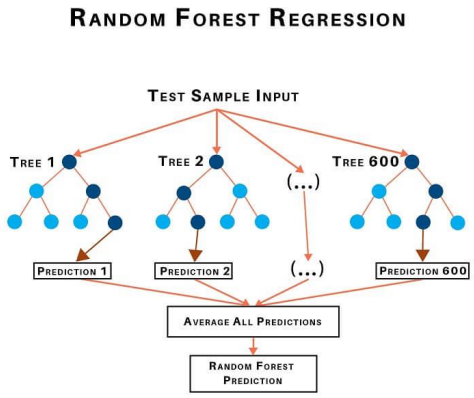


Se toma el promedio de cada división y se asigna a cada división denominada *Terminal Leaf* del Árbol de Decisión. Al dividir los datos en divisiones relevantes, el algoritmo de aprendizaje automático puede predecir con mayor precisión el valor de la variable independiente.

Una aplicación práctica, por ejemplo, predecir el salario de un empleado se puede hacer en Python. Tomando los datos de salario y puesto como variables dependientes, podemos calcular el salario predicho de cualquier empleado dado su puesto utilizando el modelo de regresión de árbol de decisión.



Random Forest Regression



Recordemos que la varianza es la medida de la variabilidad de un conjunto de datos que indica hasta qué punto se distribuyen los diferentes valores. Matemáticamente, se define como la suma de los cuadrados de las diferencias entre una variable y su media, dividido entre el numero de datos.

Veamos el caso de los árboles de decisión. Como sabemos, pueden reconstruir patrones muy complejos, pero tienden a tener un rendimiento inferior incluso si se producen cambios menores en los datos. Es por eso que un árbol de decisiones independiente no obtendrá grandes resultados. Aún así, si compone muchos de estos árboles, el rendimiento predictivo mejorará drásticamente. Esto es un método de conjunto llamado Random Forest.

¿Qué es el esemble learning?

La idea general del *esemble learning* es bastante simple. Debe entrenar varios algoritmos de ML y combinar sus predicciones de alguna manera. Tal enfoque tiende a hacer predicciones más precisas que cualquier modelo individual. Un modelo Ensemble es un modelo que consta de muchos modelos base.

Bagging

Bootstrap Aggregating o Bagging es una técnica bastante simple pero realmente poderosa. Comprender el concepto general de Bagging es realmente crucial, ya que es la base del algoritmo Random Forest (RF). Revisemos en profundidad el algoritmo general de bagging.

Para empezar, supongamos que tiene algunos datos originales que desea utilizar como conjunto de entrenamiento (conjunto de datos D). Quiere tener K modelos base en nuestro conjunto.

Para promover la variación del modelo, el bagging requiere entrenar cada modelo en el conjunto en un subconjunto elegido al azar del conjunto de entrenamiento. El número de muestras en cada subconjunto suele ser como en el conjunto de datos original (por ejemplo, N), aunque puede ser menor.

Random Forest Regression

Para crear cada subconjunto, debe utilizar una técnica de arranque:

Primero, extraiga aleatoriamente una muestra de su conjunto de datos original D y colóquela en su subconjunto

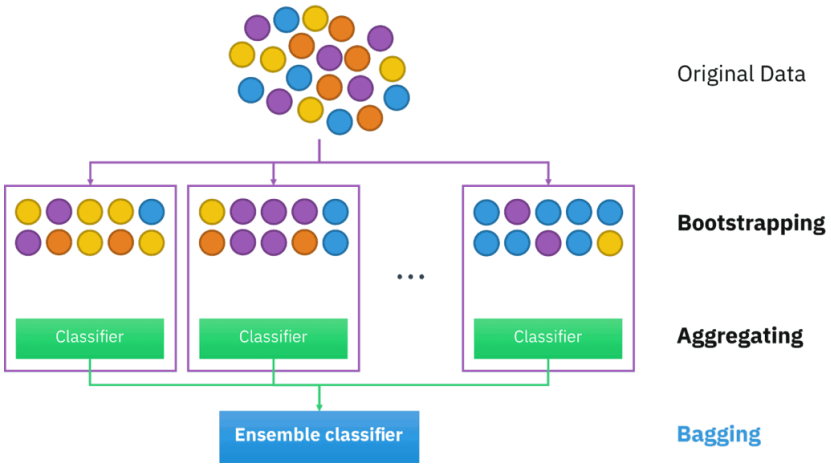
Segundo, devuelva la muestra a D (esta técnica se llama muestreo con reemplazo)

Tercero, realice los pasos a y b N (o menos) veces para llenar su subconjunto

Luego realice los pasos a, b y c K – 1 vez para tener K subconjuntos para cada uno de sus K modelos base

Construya cada uno de los modelos base K en su subconjunto

Combine sus modelos y haga la predicción final



En el caso de la regresión, solo debe tomar el promedio de las predicciones del modelo K. Aún así, recuerde que su lógica de combinación puede diferir de la presentada en este artículo. Sin embargo, debes ser lógico cuando juegues con él.

En general, el bagging es una buena técnica que ayuda a manejar el sobreajuste y reduce la varianza.

¿Qué es un bosque aleatorio?

Random Forest es un algoritmo de aprendizaje supervisado que se basa en el método de esemble learning y muchos árboles de decisión. Random Forest es una técnica de bagging, por lo que todos los cálculos se ejecutan en paralelo y no hay interacción entre los árboles de decisión al construirlos. RF se puede utilizar para resolver tareas de clasificación y regresión.

Random Forest Regression

El nombre "bosque aleatorio" proviene de la idea de bagging de la aleatorización de datos (aleatorio) y la construcción de múltiples árboles de decisión (bosque). En general, es un poderoso algoritmo de ML que limita las desventajas de un modelo de árbol de decisiones.

Para aclarar las cosas, vemaos el algoritmo exacto de Random Forest:

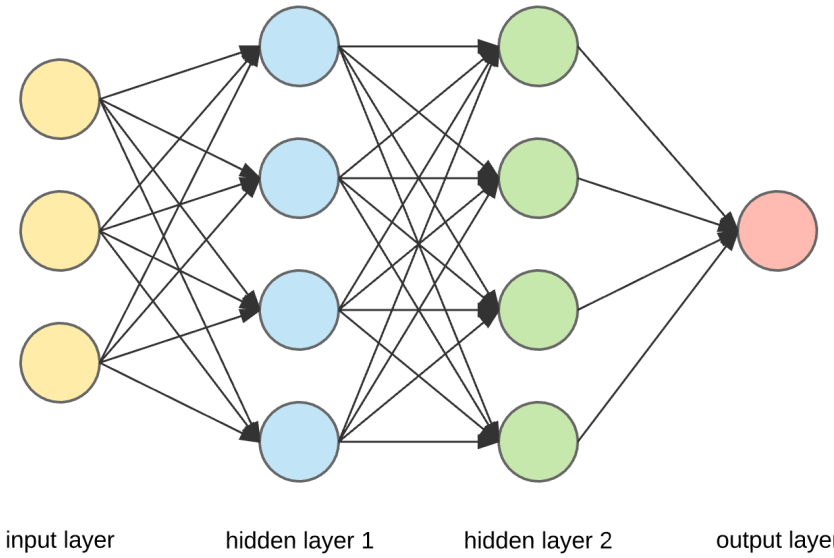
Tenemos un conjunto de datos originales D, queremos tener K árboles de decisión en nuestro conjunto. Además, tenemos un número N: se construirá un árbol hasta que haya muestras menores o iguales a N en cada nodo (para la regresión, la tarea N suele ser igual a 5). Además, tiene un número F: número de características que se seleccionarán aleatoriamente en cada nodo del árbol de decisión. La función que se usará para dividir el nodo se selecciona de estas funciones F (para la tarea de regresión, F suele ser igual a sqrt (número de funciones del conjunto de datos original D)

Todo lo demás es bastante simple. Random Forest crea K subconjuntos de los datos del conjunto de datos original D. Las muestras que no aparecen en ningún subconjunto se denominan muestras "listas para usar".

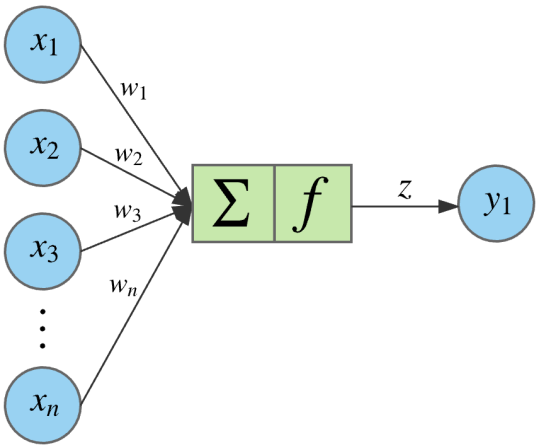
Los árboles K se construyen utilizando un solo subconjunto. Además, cada árbol se construye hasta que haya menos o igual a N muestras en cada nodo. Además, en cada nodo, las características F se seleccionan aleatoriamente. Uno de ellos se utiliza para dividir el nodo.
K modelos entrenados forman un conjunto y el resultado final de la tarea de regresión se produce promediando las predicciones de los árboles individuales

Redes Neuronales

Las redes neuronales artificiales (ANN) son redes neuronales multicapa totalmente conectadas que se ven como la figura a continuación. Constan de una capa de entrada, varias capas ocultas y una capa de salida. Cada nodo en una capa está conectado a todos los demás nodos en la siguiente capa. Profundizamos la red aumentando el número de capas ocultas.



Un nodo dado toma la suma ponderada de sus entradas y la pasa a través de una función de activación no lineal. Esta es la salida del nodo, que luego se convierte en la entrada de otro nodo en la siguiente capa. La señal fluye de izquierda a derecha y la salida final se calcula realizando este procedimiento para todos los nodos. Entrenar esta red neuronal profunda significa aprender los pesos asociados con todos los bordes.



Redes Neuronales

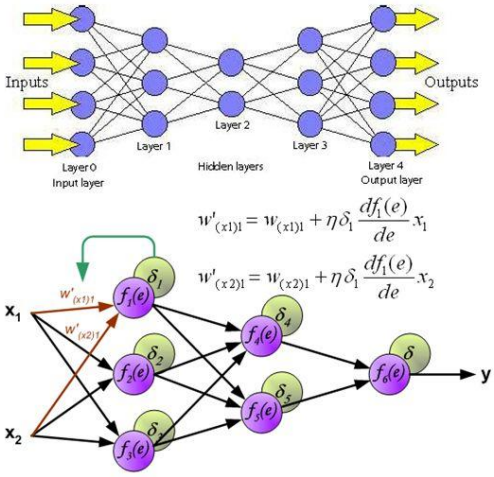
Se omitio el término de sesgo por simplicidad. Bias es una entrada para todos los nodos y siempre tiene el valor 1. Permite desplazar el resultado de la función de activación hacia la izquierda o hacia la derecha. También ayuda al modelo a entrenarse cuando todas las características de entrada son 0.

$$z = f(b + x \cdot w) = f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

$$x \in d_{1\times n},\ w \in d_{n\times 1},\ b \in d_{1\times 1},\ z \in d_{1\times 1}$$

Hasta ahora, hemos descrito el pase hacia adelante, es decir, dada una entrada y pesos, cómo se calcula la salida. Una vez que se completa el entrenamiento, solo ejecutamos el pase hacia adelante para hacer las predicciones. Pero primero necesitamos entrenar a nuestro modelo para que realmente aprenda los pesos, y el procedimiento de entrenamiento funciona de la siguiente manera:

Feed forward/Backpropagation Neural Network



Feed forward algorithm:

- Activate the neurons from the bottom to the top.

Backpropagation:

- Randomly initialize the parameters
- Calculate total error at the top, $f_6(e)$
- Then calculate contributions to error, δ_n, at each step going backwards.

13

1 Inicialice aleatoriamente los pesos para todos los nodos. Existen métodos de inicialización inteligente que exploraremos luego.

2 Para cada ejemplo de entrenamiento, realice un pase hacia adelante utilizando los pesos actuales y calcule la salida de cada nodo de izquierda a derecha. La salida final es el valor del último nodo.

3 Compare el resultado final con el objetivo real en los datos de entrenamiento y mida el error usando una función de pérdida.

4 Realice un *backwards* de derecha a izquierda y propague el error a cada nodo individual mediante la *backpropagation*. Calcule la contribución de cada peso al error y ajuste los pesos en consecuencia utilizando el descenso de gradiente. Propaga los gradientes de error desde la última capa.

La *backpropagation* con descenso de gradiente es literalmente la "magia" detrás de los modelos de aprendizaje profundo. Es un tema bastante largo e implica algo de cálculo, por lo que no entraremos en detalles.

En el mundo de ML estándar, esta arquitectura *feed forward* se conoce como perceptrón multicapa. La diferencia entre ANN y perceptron es que ANN usa una función de activación no lineal como sigmoide pero el perceptron usa la función escalaon. Esa no linealidad le da a la ANN su gran poder.