GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Unión Europea-
NextGenerationEU

UD2-APRENDIZAJE SUPERVISADO

Curso de Especialización en Inteligencia Artificial y Big Data

MP Sistemas de Aprendizaje Automático

Contenidos de la UD2

1. Introducción
2. Review: Data Preprocessing
3. KNN K-Nearest Neighbours
4. Regresión Lineal
5. Regresión Logística
6. Decision Trees y Random Forest
7. Suport Vector Machines (SVM)

¿Qué tienen en común todos los modelos de ML?

Contrastar pdf moodle de resumen (CheatSheet ML algorithms Python and R codes)

Data Preprocessing

Step 1: Importing the required Libraries

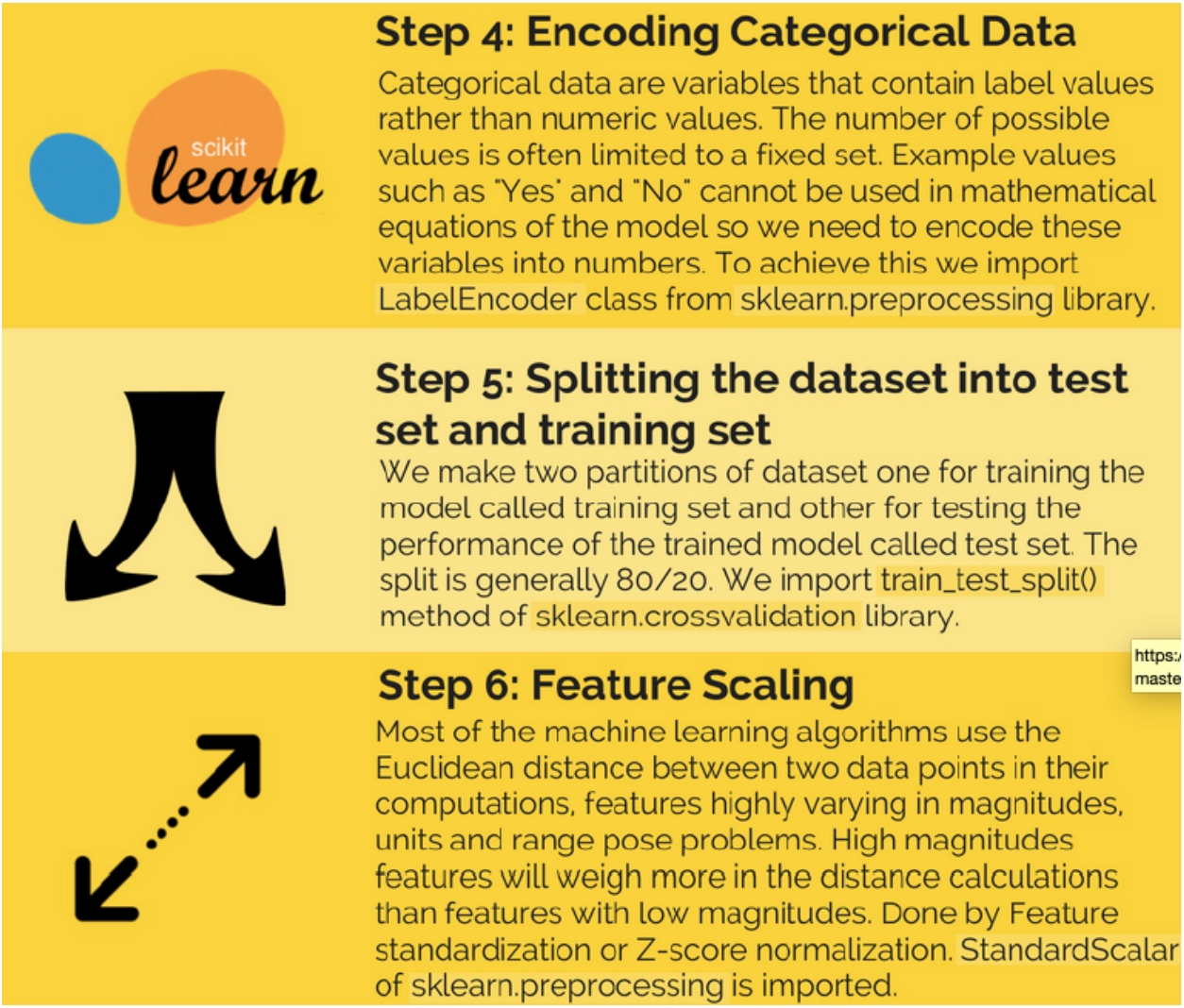
These Two are essential libraries which we will import every time.
NumPy is a Library which contains Mathematical functions.
Pandas is the library used to import and manage the data sets.

Step 2: Importing the Data Set

Data sets are generally available in .csv format. A CSV file stores tabular data in plain text. Each line of the file is a data record. We use the `read_csv` method of the pandas library to read a local CSV file as a dataframe. Then we make separate Matrix and Vector of independent and dependent variables from the dataframe.

Step 3: Handling the Missing Data

The data we get is rarely homogeneous. Data can be missing due to various reasons and needs to be handled so that it does not reduce the performance of our machine learning model. We can replace the missing data by the Mean or Median of the entire column. We use `Imputer` class of `sklearn.preprocessing` for this task.


scikit
learn
Step 4: Encoding Categorical Data
Categorical data are variables that contain label values rather than numeric values. The number of possible values is often limited to a fixed set. Example values such as "Yes" and "No" cannot be used in mathematical equations of the model so we need to encode these variables into numbers. To achieve this we import LabelEncoder class from sklearn.preprocessing library.
Step 5: Splitting the dataset into test set and training set
We make two partitions of dataset one for training the model called training set and other for testing the performance of the trained model called test set. The split is generally 80/20. We import train_test_split() method of sklearn.crossvalidation library.
Step 6: Feature Scaling
Most of the machine learning algorithms use the Euclidean distance between two data points in their computations, features highly varying in magnitudes, units and range pose problems. High magnitudes features will weigh more in the distance calculations than features with low magnitudes. Done by Feature standardization or Z-score normalization. StandardScalar of sklearn.preprocessing is imported.

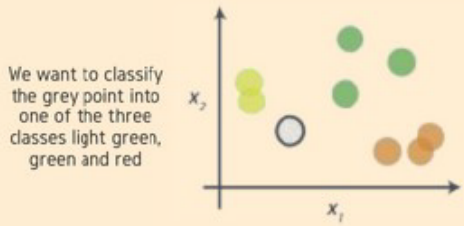
K-NEAREST NEIGHBORS

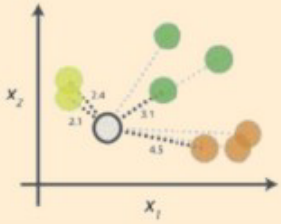
What is k-NN?

K-Nearest Neighbor algorithm is a simple yet most used classification algorithm. It can also be used for regression.

KNN is non-parametric (means that it does not make any assumptions on the underlying data distribution), instance-based (means that our algorithm doesnt explicitly learn a model. Instead, it chooses to memorize the training instances.) and used in a supervised learning setting.

k-NN is also called a lazy algorithm because it is instance based.

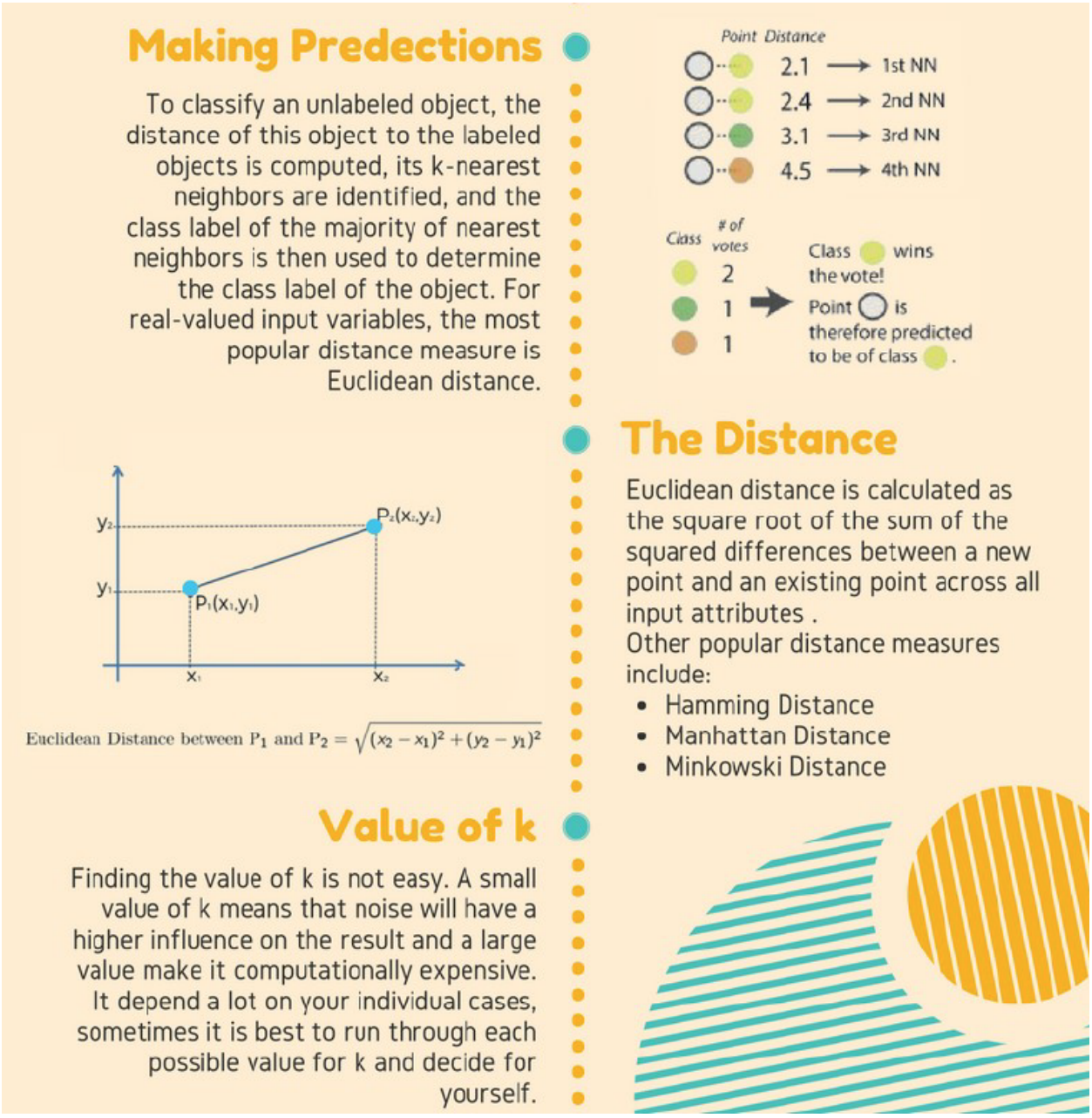




Start by calculating the distance between the grey point and k -nearest points

How Does k-NN Algorithm work?

k-NN when used used for classification—the output is a class membership (predicts a class—a discrete value).
There are three key elements of this approach: a set of labeled objects, e.g., a set of stored records, a distance between objects, and the value of k, the number of nearest neighbors.



https://www.aprendemachinelearning.com/clasificar-con-k-nearest-neighbor-ejemplo-en-python/

Regresión Lineal Univariable y Multivariable

SIMPLE LINEAR REGRESSION

Predicting a response using a single feature.

It is a method to predict dependent variable (Y) based on values of independent variables (X). It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

How to find the best fit line?

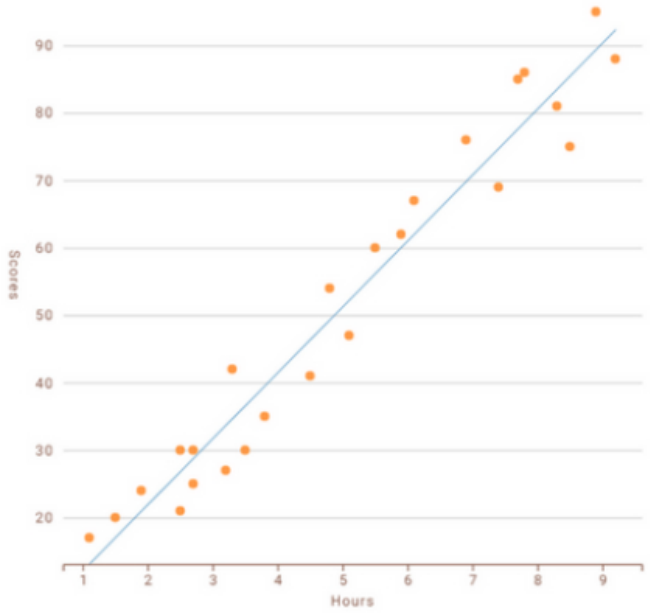
In this regression model, we are trying to minimize the errors in prediction by finding the "line of best fit" – the regression line from the errors would be minimal. We are trying to minimize the length between the observed value (Yi) and the predicted value from our model (Yp).

Predicted Value

$y_i \quad y_p$

Observed Value

$$\min \{\text{SUM}(y_i - y_p)^2\}$$



Dependent Variable

$$y = b_0 + b_1 x_1$$

Independent Variable

In this regression task, we will predict the percentage of marks that a student is expected to score based upon the number of hours they studied.

Slope

$$Score = b_0 + b_1 * hours$$

Y - intercept

STEP 1: PREPROCESS THE DATA

We will follow the same steps as in my previous infographic of Data Preprocessing.

- Import the Libraries.
- Import the DataSet.
- Check for Missing Data.
- Split the DataSet.
- Feature Scaling will be taken care by the Library we will use for Simple Linear Regression Model.

STEP 2: FITTING SIMPLE LINEAR REGRESSION MODEL TO THE TRAINING SET

To fit the dataset into the model we will use `LinearRegression` class from `sklearn.linear_model` library. Then we make an object `regressor` of `LinearRegression` Class. Now we will fit the regressor object into our dataset using `fit()` method of `LinearRegression` Class.

STEP 3: PREDICTING THE RESULT

Now we will predict the observations from our test set. We will save the output in a vector `Y_pred.` To predict the result we use predict method of `LinearRegression` Class on the regressor we trained in the previous step.

STEP 4: VISUALIZATION

The final step is to visualize our results. We will use `matplotlib.pyplot` library to make Scatter Plots of our Training set results and Test set results to see how close our model predicted the Values

MULTIPLE LINEAR REGRESSION

Multiple linear regression attempts to model the relationship between two or more features and a response by fitting a linear equation to observed data. The steps to perform multiple linear regression are almost similar to that of simple linear regression. The difference lies in the evaluation. You can use it to find out which factor has the highest impact on the predicted output and how different variables relate to each other.

Dependent variable

multiple independent variables

$$y = b_\circ + b_1x_1 + b_2x_2 \dots\dots b_nx_n$$

ASSUMPTIONS

FOR A SUCCESSFUL REGRESSION ANALYSIS, IT'S ESSENTIAL TO VALIDATE THESE ASSUMPTIONS.

1. Linearity: The relationship between dependent and independent variables should be Linear.
2. Homoscedasticity (constant variance) of the errors should be maintained.
3. Multivariate Normality: Multiple regression assumes that the residuals are normally distributed.
4. Lack of Multicollinearity: It is assumed that there is little or no multicollinearity in the data. Multicollinearity occurs when the features (or independent variables) are not independent of each other.

NOTE

Having too many variables could potentially cause our model to become less accurate, especially if certain variables have no effect on the outcome or have a significant effect on other variables. There are various methods to select the appropriate variable like -

1. Forward Selection
2. Backward Elimination
3. Bi-directional Comparision

DUMMY VARIABLES

Gender
Female
Female
Male
Female
Male
Male
Male

Male	Female
0	1
0	1
1	0
0	1
1	0
1	0
1	0

Using categorical data in Multiple Regression Models is a powerful method to include non-numeric data types into a regression model.

Categorical data refers to data values which represent categories - data values with a fixed and unordered number of values, for instance, gender (male/female). In a regression model, these values can be represented by dummy variables - variables containing values such as 1 or 0 representing the presence or absence of the categorical value.

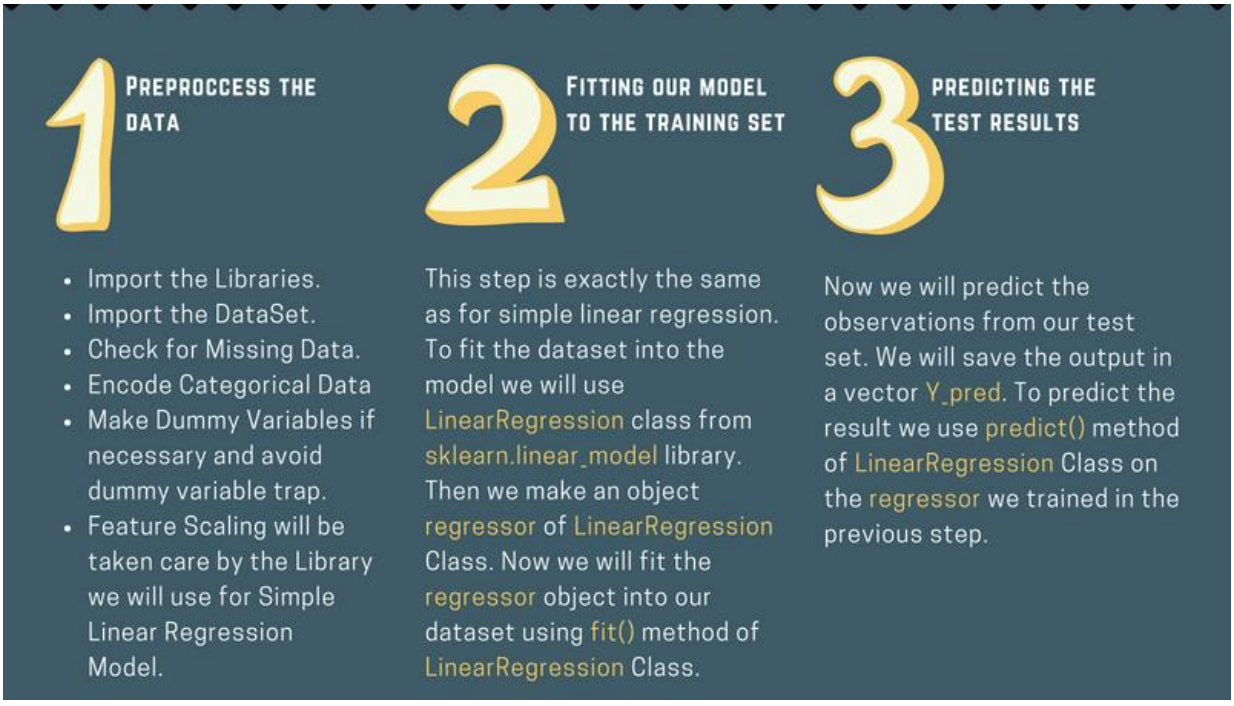
DUMMY VARIABLE TRAP

The Dummy Variable trap is a scenario in which two or more variables are highly correlated; in simple terms, one variable can be predicted from the others. Intuitively, there is a duplicate category: if we dropped the male category it is inherently defined in the female category (zero female value indicate male, and vice-versa).

The solution to the dummy variable trap is to drop one of the categorical variables - if there are m number of categories, use m-1 in the model, the value left out can be thought of as the reference value.

Dummy variable $D_2 = 1 - D_1$ Dummy variable

$$y = b_\circ + b_1x_1 + b_2x_2 + b_3D_1$$


1
PREPROCCESS THE DATA
Import the Libraries.
Import the DataSet.
Check for Missing Data.
Encode Categorical Data
Make Dummy Variables if necessary and avoid dummy variable trap.
Feature Scaling will be taken care by the Library we will use for Simple Linear Regression Model.
2
FITTING OUR MODEL TO THE TRAINING SET
This step is exactly the same as for simple linear regression. To fit the dataset into the model we will use LinearRegression class from sklearn.linear_model library. Then we make an object regressor of LinearRegression Class. Now we will fit the regressor object into our dataset using fit() method of LinearRegression Class.
3
PREDICTING THE TEST RESULTS
Now we will predict the observations from our test set. We will save the output in a vector Y_pred. To predict the result we use predict() method of LinearRegression Class on the regressor we trained in the previous step.

Regresión Logística

WHAT IS LOGISTIC REGRESSION

Logistic regression is used for a different class of problems known as classification problems. Here the aim is to predict the group to which the current object under observation belongs to. It gives you a discrete binary outcome between 0 and 1. A simple example would be whether a person will vote or not in upcoming elections.

How Does It Work?

Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict) and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function.

Making Predictions

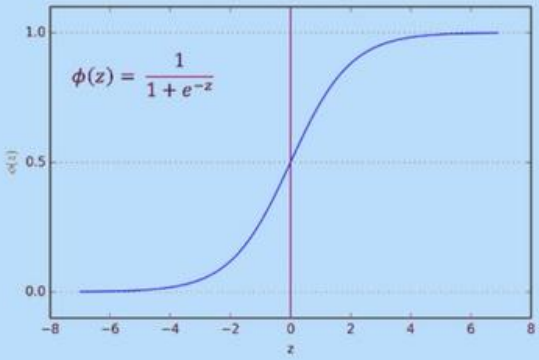
These probabilities must then be transformed into binary values in order to actually make a prediction. This is the task of the logistic function, also called the sigmoid function. This values between 0 and 1 will then be transformed into either 0 or 1 using a threshold classifier.

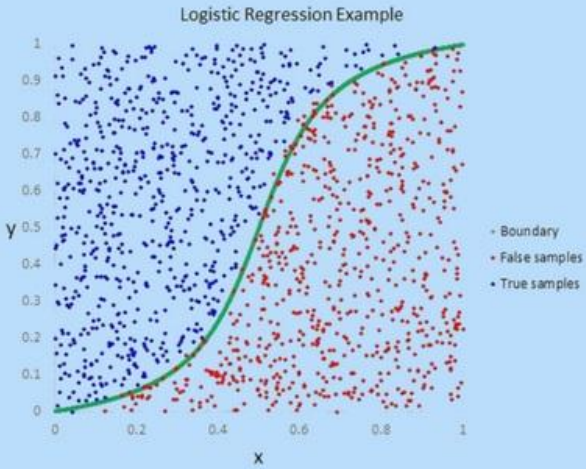
Logistic vs Linear

Logistic regression gives you a discrete outcome but linear regression gives a continuous outcome.

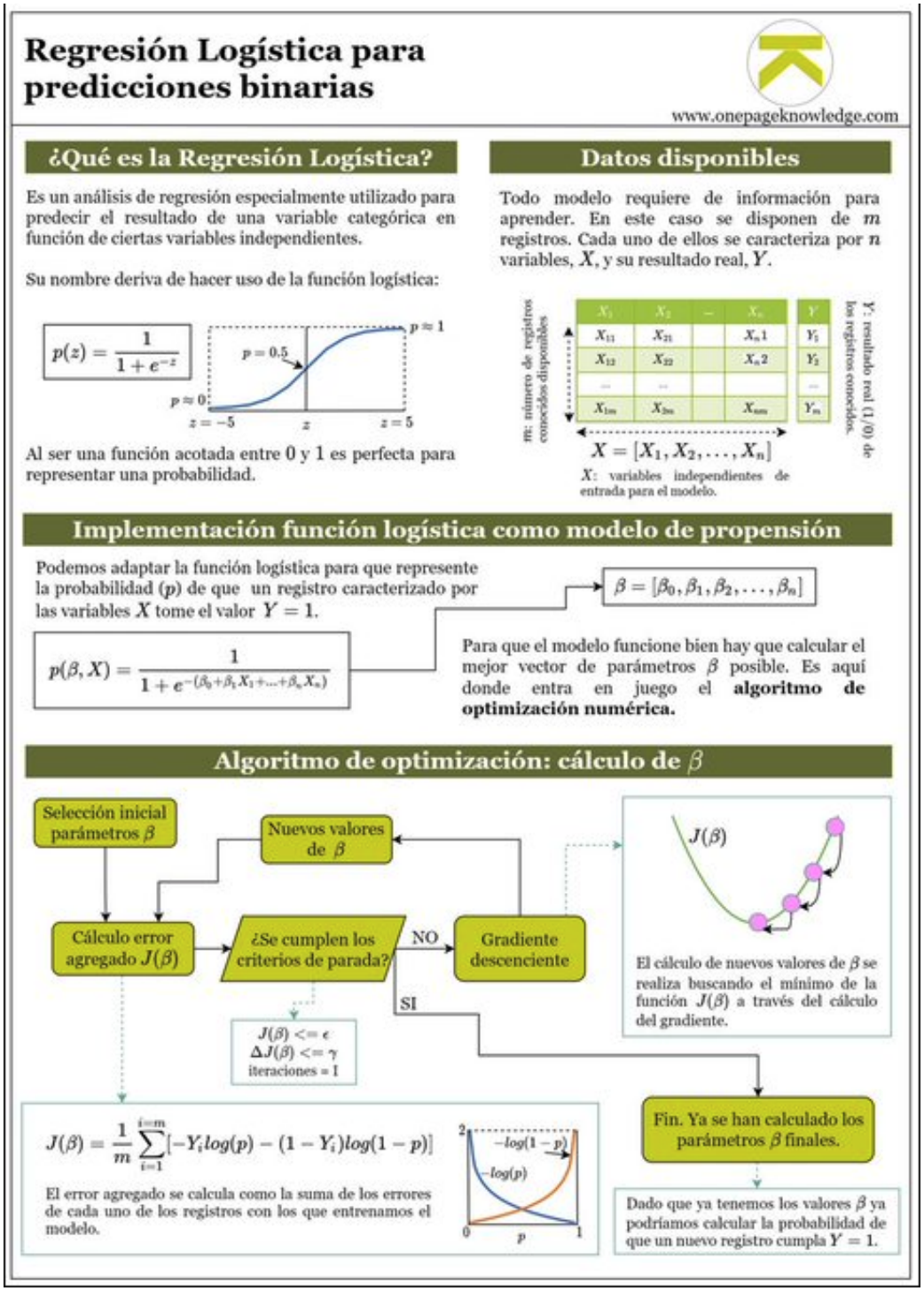
Sigmoid Function

The Sigmoid-Function is an S-shaped curve that can take any real-valued number and map it into a value between the range of 0 and 1, but never exactly at those limits.





Ampliación: Documentación extraída de Álvaro de onepageknowledge:
https://twitter.com/1_pageknowledge/status/1358376337979944960



Vamos a hablar sobre un modelo cuyo objetivo es calcular la probabilidad de que ocurra un cierto evento.

Este tipo de problemas aparecen continuamente en la industria:

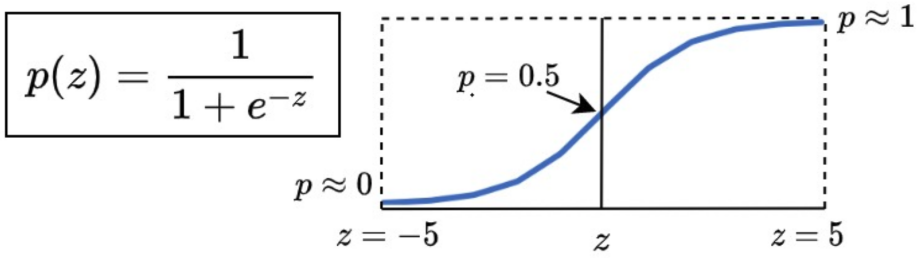
- Probabilidad de impago
- Probabilidad de que un cliente compre un producto

A modo de ejemplo, imaginemos que buscamos un modelo que nos dé la probabilidad de que una empresa suba en bolsa más de un 5% este año.

Uno de los modelos más sencillos y utilizados para resolver este tipo de problemas se llama 'Regresión Logística'. Fácil. Porque se basa en la función logística.

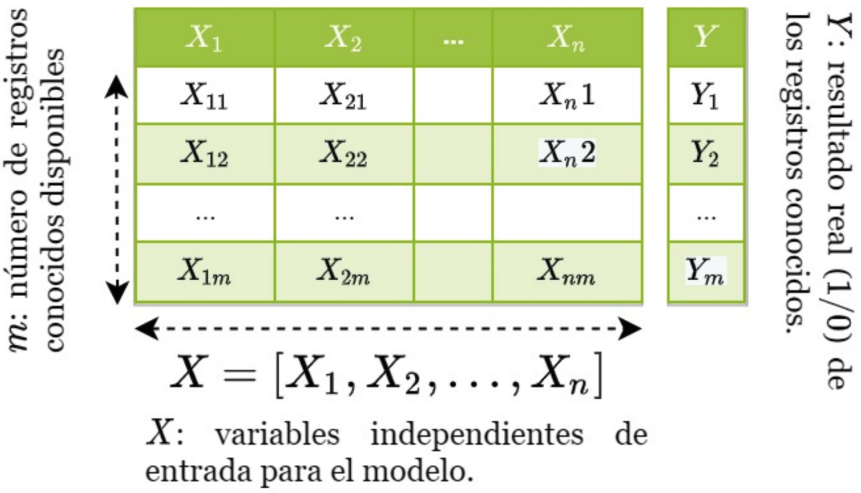
Una de las propiedades de esta función es que está acotada entre 0 y 1 (ver imagen). Por tanto es muy apropiada para representar una probabilidad.

Veamos cómo esta función podría darnos la probabilidad de hacer un +5% que buscamos.



Todo modelo de machine learning necesita datos de los que 'aprender' patrones.

Imaginemos que tenemos la información de 'm' empresas y sabemos si subieron +5% en 2020 (Y=1) o no (Y=0).
También disponemos de características de las empresas (X)



X1 podría ser la capitalización bursátil.
X2 podría ser el número de empleados.
X3 podría ser la variación del precio de la acción en 2019.

Y podríamos tener muchas más variables de este estilo. Cuanto más relacionadas están las variables con las futuras subidas en bolsa, mejor.
Al final lo que estamos tratando de hacer es explicarle al modelo que pasó en el pasado con muchas empresas para que pueda buscar patrones y nos sirva para hacer inferencias de qué podría pasar en el futuro.
Bien. Ya tenemos datos y una función. Podemos combinar ambas cosas para crear una función logística.

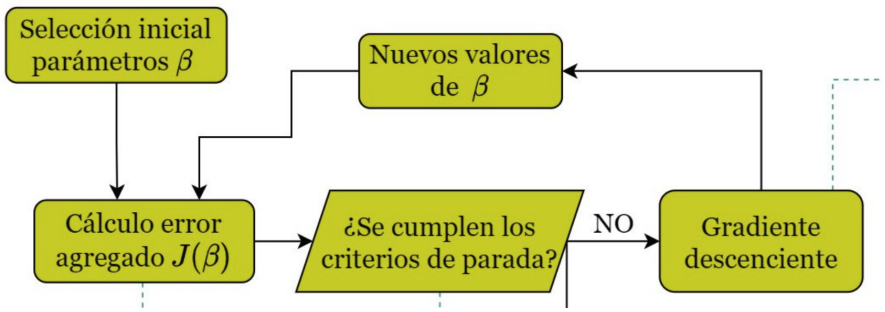
La probabilidad 'p' depende de X (es decir, de los datos de la empresa en cuestión) y de unos parámetros BETA.

'X' los conocemos. Pero qué pasa con los valores BETA?

$$p(\beta, X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + ... + \beta_n X_n)}}$$

El kit de la cuestión es calcular los parámetros BETA que hagan que nuestra función para calcular 'p' funcione correctamente.

¿Cómo los calculamos? Aquí entra un algoritmo de optimización numérica.
Este proceso de cálculo lo podemos pintar de esta forma:



1.- Calcular BETA iniciales

Por simplificar imaginemos que nos los inventamos.

Al inventarnos los valores de los parámetros nuestra función no va a funcionar correctamente. Es decir, no va a calcular las probabilidades bien.

Pero ¿qué significa funcionar bien o mal?

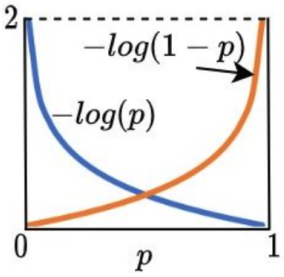
2.- Cálculo del error J

J es una medida del error agregado que comete nuestra función logística. La forma de calcular el error depende del algoritmo y modelo que estemos utilizando.La cuestión es que de algún modo tenemos que cuantificar el error.

En el caso que estamos trabajando podríamos utilizar esta fórmula:

$$J(\beta) = \frac{1}{m}\sum_{i=1}^{i=m}[-Y_i log(p) - (1 - Y_i) log(1 - p)]$$

El error agregado se calcula como la suma de los errores de cada uno de los registros con los que entrenamos el modelo.



Fíjate que el error se calcula como la suma de los errores cometidos en las 'm' empresas con las que estamos tratando de enseñar a nuestro modelo.

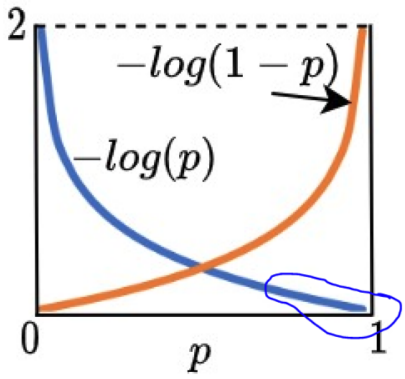
Por cierto, a la función J se le llama **función de coste**.

Imagina que uno de los ejemplos con los que estamos entrenando a nuestro modelo es una empresa que SÍ subió +5% en el 2020. Entonces Y=1.

Entonces el error asociado es -log(p) porque el término de la derecha desaparece.
¿Qué pasa entonces? Pues si nuestro modelo con parámetros inventados da una probabilidad cercana a 1 el error será bajo.

Si es cercana a 0 el error será grande.

Esto tiene sentido: sabemos que la empresa SI subió +5% por lo que nuestro modelo debería dar una probabilidad alta.



3.- Criterios de parada

Esto son simplemente unas reglas para decidir si nuestro algoritmo de optimización debe terminar. Por poner un ejemplo sencillo: podemos decidir que nuestro algoritmo pare cuando J sea menor que un valor umbral. En la primera iteración nuestra función logística será muy mala y el valor de J será muy alto (es lo que tiene inventarse los parámetros…). Por tanto tenemos que buscar unos parámetros mejores.

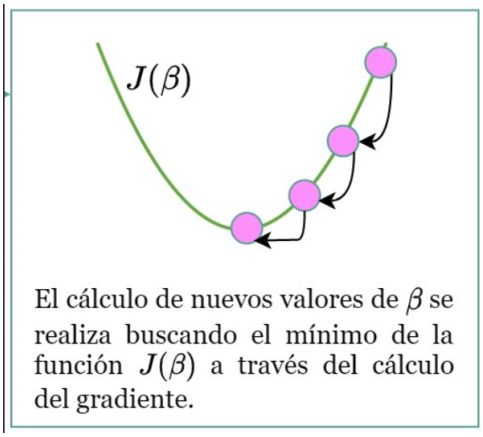
¿Cómo hacemos esto?

Vamos a uno de los puntos más importantes:

4.- Cálculo de nuevos parámetros beta

¿Te acuerdas cuando en el instituto tratabas de encontrar el mínimo de una función haciendo la derivada e igualarla a 0?

Pues es algo similar. En nuestro caso queremos minimizar el valor de J (el error agregado).



El cálculo de nuevos valores de β se realiza buscando el mínimo de la función $J(\beta)$ a través del cálculo del gradiente.

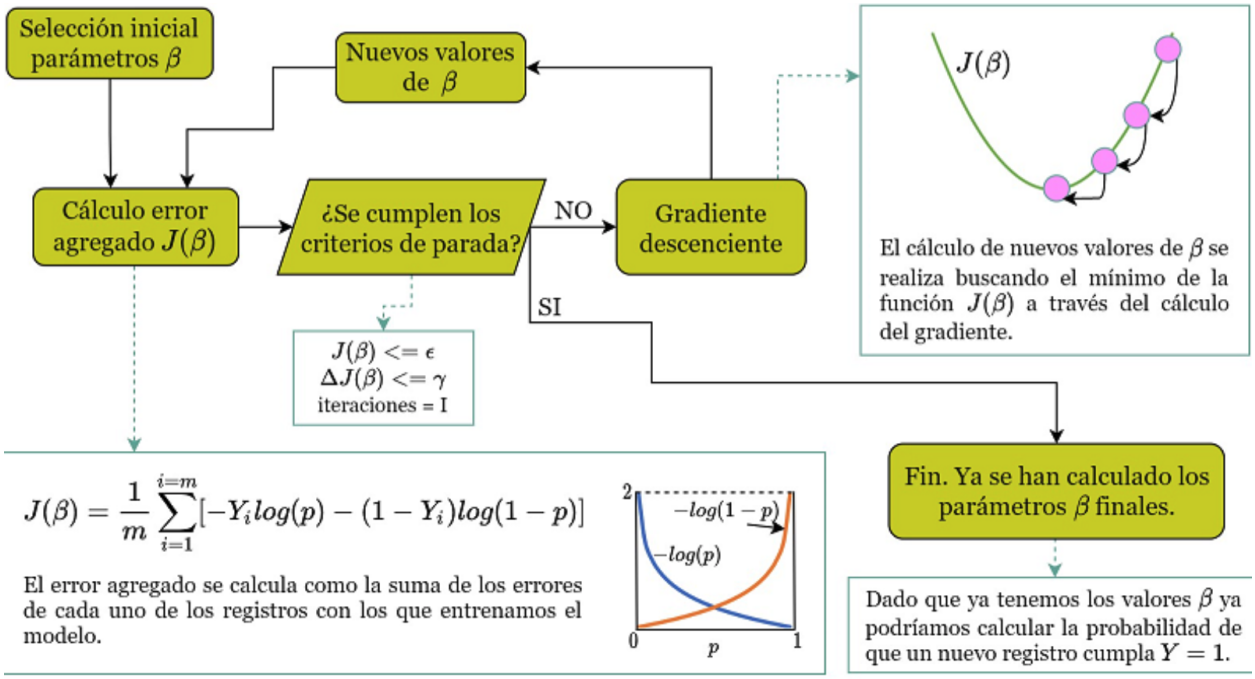
Ojo! Es similar pero no tan sencillo. Hay que calcular el **gradiente (vector de derivadas parciales).**

La cuestión es que gracias a las derivadas somos capaces de obtener unos nuevos parámetros BETA. ¿Estos nuevos parámetros serán mejores que los anteriores? En principio sí.

Luego se repite el proceso:

- Con los nuevos parámetros volvemos a calcular el error J (que ahora será menor que antes)
- Vemos si cumplimos nuestros criterios de parada
- En caso negativo, volvemos a recurrir al gradiente para obtener nuevos Beta.

Y este proceso iterativo se repite hasta cumplir un criterio de parada. Cuando esto ocurra tendremos unos parámetros BETA buenos para nuestra función logística.

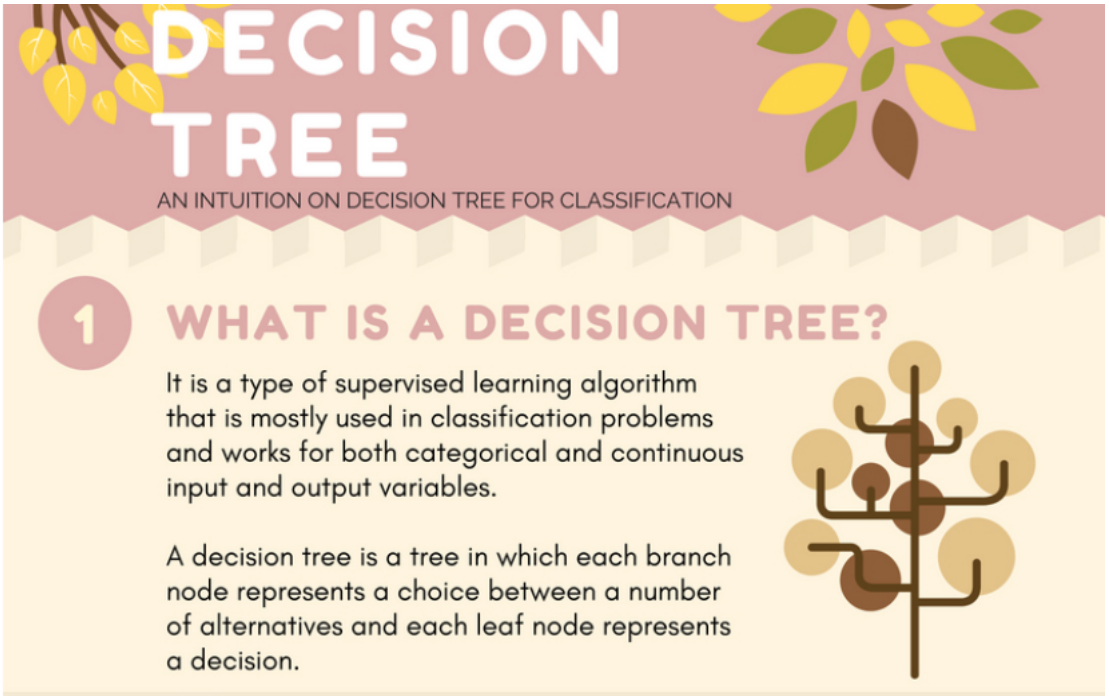


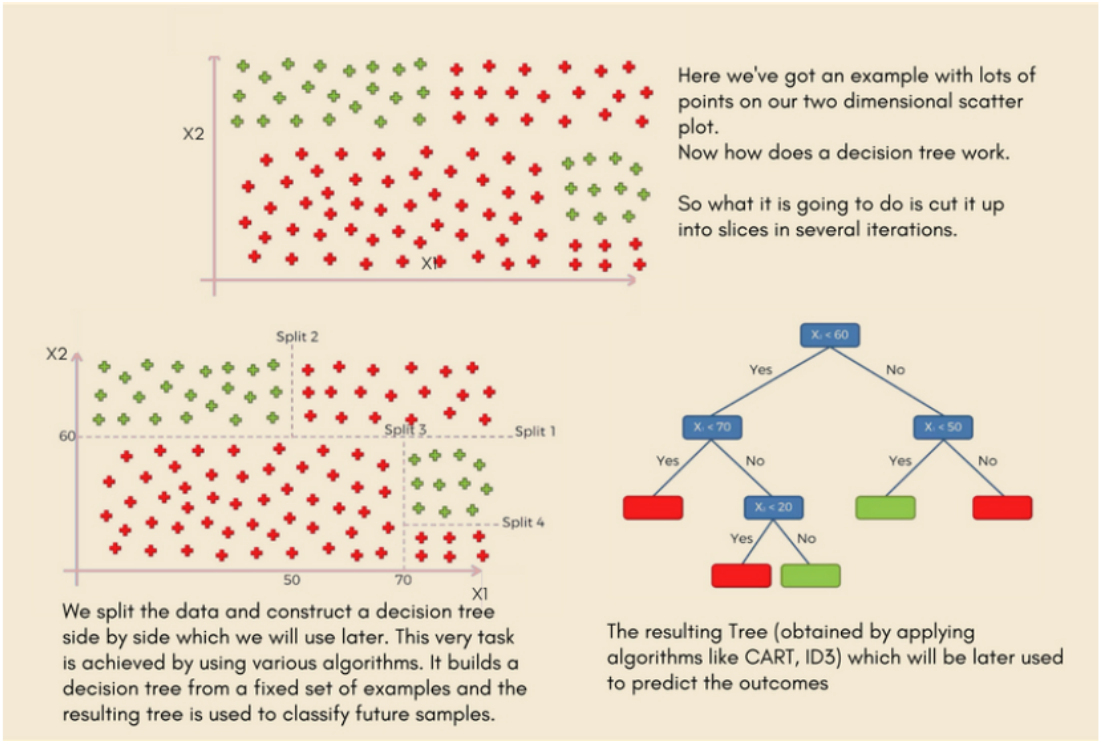
Y ya está. Ya tendríamos nuestra función para calcular p.

Para utilizarlo solo tendríamos que introducir los valores de X de la empresa que queramos y obtener la probabilidad de que suba +5% este año.

Al final todo es cuestión de construir una función matemática más o menos compleja que depende de ciertos parámetros (ojo, pueden ser millones de parámetros).
Cuando se habla de 'entrenar un modelo' se hace referencia al cálculo de esos parámetros desconocidos de nuestra función.

DECISION TREE





3 DECISION TREE ALGORITHM: ID3

ID3 stands for Iterative Dichotomizer 3. The basic idea is to construct the decision tree by employing a top-down, greedy search through the given sets to test each attribute at every tree node.
Sounds simple — but which node should we select to build the correct and most precise decision tree? How would we decide that? Well, we have some measures that can help us in selecting the best choice!

Loop:
A –> Best Attribute
Assign A as decision attribute for node.
For each value of A, create a descendant of node.
Sort training examples to leaves.
If examples perfectly classified:
STOP
Else:
Iterate over leaves

INFORMATION GAIN

The best attribute is the one which gives us maximum Information Gain. Broadly speaking, it is a mathematical way to capture the amount of information we want by picking a particular attribute. But what it really speaks about us the reduction in the randomness, over the tables that we have with the set of data, based upon knowing the value of a particular attribute. Information gain is defined by:

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

S = Collection of training examples
A = Particular attribute
|Sv| = Number of elements in Sv
|S| = Number of elements in S
v = All the possible values of the attribute

ENTROPY

Entropy in machine learning also carries almost the same meaning as it does in Thermodynamics. It is a measure of randomness.

$$Entropy = -\sum_{v} p(v) \log_2 p(v)$$

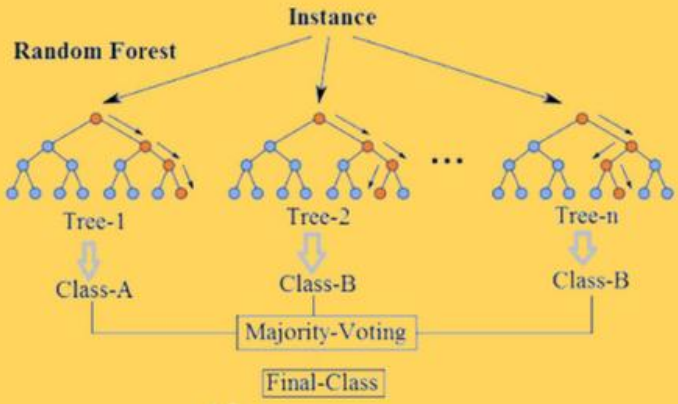
Where v = possible values for the attribute.

Steps :
1.compute the entropy for data-set
2.for every attribute/feature:
1. calculate entropy for all categorical values
2. take average information entropy for the current attribute
3. calculate gain for the current attribute
3. pick the highest gain attribute.
4. Repeat until we get the tree we desired

RANDOM FORESTS ARE SUPERVISED ENSEMBLE-LEARNING MODELS USED FOR CLASSIFICATION AND REGRESSION.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

WHAT IS THE RANDOM FOREST ALGORITHM?



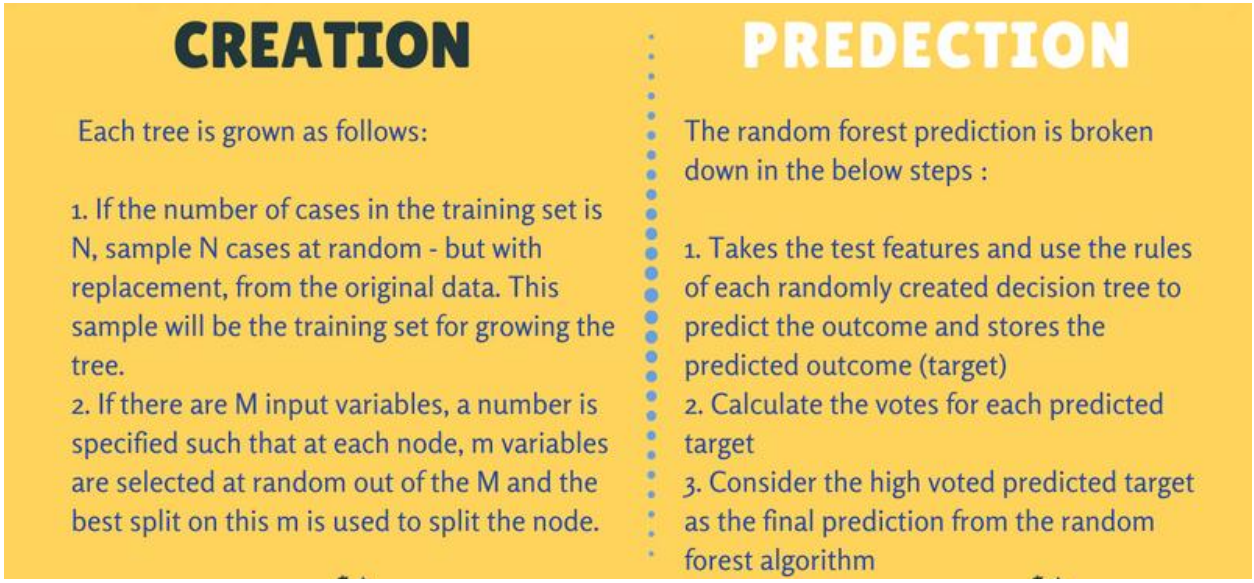
Ensemble learning models aggregate multiple machine learning models, allowing for overall better performance.

The logic behind this is that each of the models used is weak when employed on its own, but strong when put together in an ensemble. In the case of Random Forests, a large number of Decision Trees, acting as the "weak" factors, are used and their outputs are aggregated, with the result representing the "strong" ensemble.

There are two steps in the Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first step.

THE DIFFERENCE BETWEEN THE RANDOM FOREST ALGORITHM AND THE DECISION TREE ALGORITHM IS THAT IN RANDOM FOREST, THE PROCESSES OF FINDING THE ROOT NODE AND SPLITTING THE FEATURE NODES WILL RUN RANDOMLY.

HOW DOES IT WORK?


CREATION
Each tree is grown as follows:
1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number is specified such that at each node, m variables are selected at random out of the M and the best split on this m is used to split the node.
PREDECTION
The random forest prediction is broken down in the below steps :
1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the votes for each predicted target
3. Consider the high voted predicted target as the final prediction from the random forest algorithm

SVM

SUPPORT VECTOR MACHINES

What is SVM?

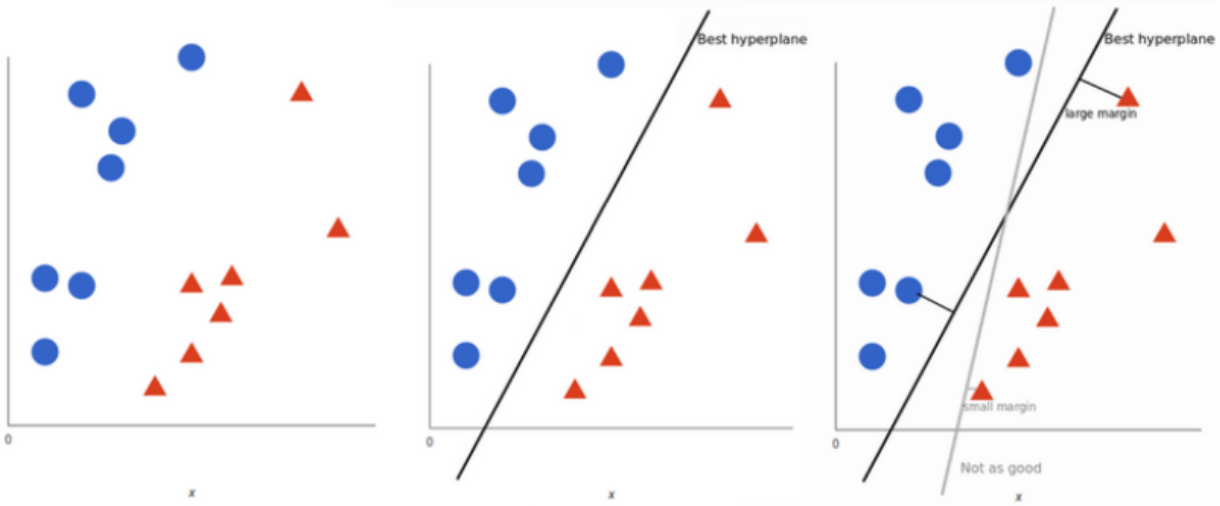
Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression. However, it is mostly used in classification problems.
In this algorithm, we plot each data item as a point in n-dimensional space (where n is the number of features) with the value of each feature being the value of a particular coordinate.

How is the data classified?

We perform classification by finding the hyperplane that differentiates the two classes very well. In other words the algorithm outputs an optimal hyperplane which categorizes new examples.

What is a optimal Hyper-Plane?

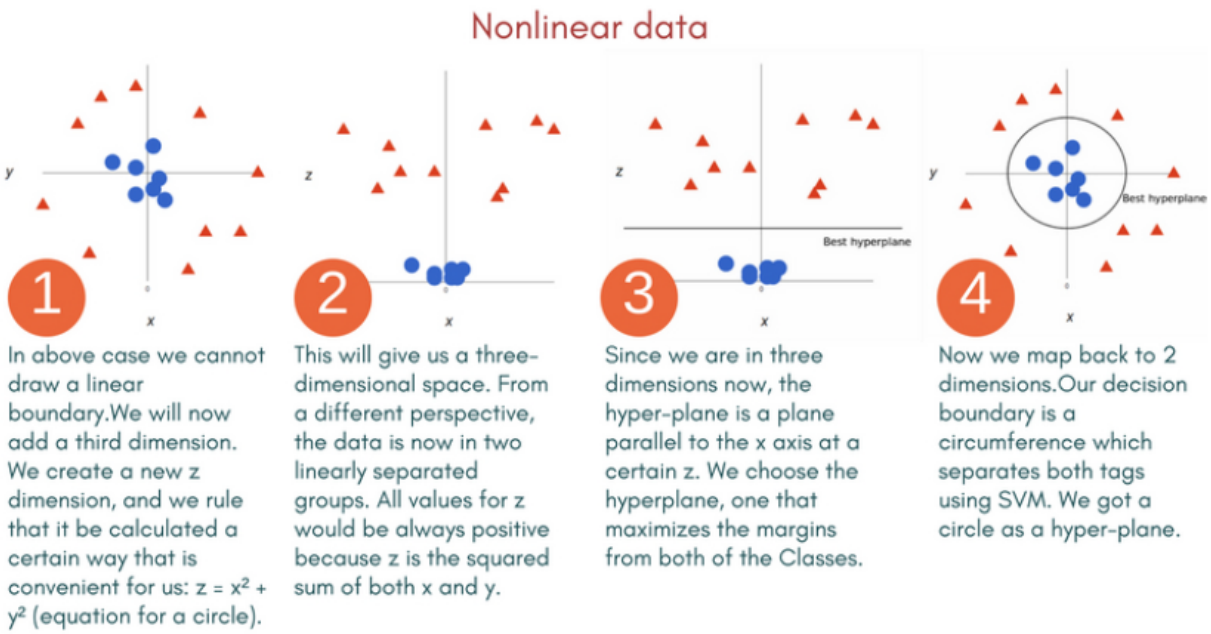
For SVM, it's the one that maximizes the margins from both tags. In other words: the hyperplane whose distance to the nearest element of each tag is the largest.



Our labeled data

In 2D, the best hyper-plane is simply a line

Not all hyper-planes are created equal



In above case we cannot draw a linear boundary.We will now add a third dimension. We create a new z dimension, and we rule that it be calculated a certain way that is convenient for us: $z = x^2 + y^2$ (equation for a circle).

This will give us a three-dimensional space. From a different perspective, the data is now in two linearly separated groups. All values for z would be always positive because z is the squared sum of both x and y.

Since we are in three dimensions now, the hyper-plane is a plane parallel to the x axis at a certain z. We choose the hyperplane, one that maximizes the margins from both of the Classes.

Now we map back to 2 dimensions.Our decision boundary is a circumference which separates both tags using SVM. We got a circle as a hyper-plane.

TUNING PARAMETERS

KERNEL

The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. Polynomial and exponential kernels calculates separation line in higher dimension. This is called kernel trick.

REGULARIZATION

For large values of this parameter, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of it will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

GAMMA

The gamma parameter defines how far the influence of a single training set reaches. With low gamma, points far away from the possible separation line are considered in calculation for the separation line. Where as high gamma means the points close to possible line are considered in calculation.

MARGIN

A margin is a separation of line to the closest class points.
A good margin is one where this separation is larger for both the classes. A good margin allows the points to be in their respective classes without crossing to other class.

WEBGRAFÍA

- Wikipedia
- [100 Days of ML Code]