

Aprendizaje Supervisado II: Clasificación

27th March 2019

Outline

- 1 De la Regresión Lineal a Logistic Regression
- 2 Support Vector Machine
- 3 Árboles de decisión
- 4 Random Forest
- 5 Lazy algorithms
- 6 Métricas de clasificación

Aprendizaje supervisado?

El aprendizaje supervisado no es más que, dado un input de *features* $\{x_1, x_2, \dots, x_m\}$, el ajuste de un modelo que aproxime la función de $f(x_1, x_2, \dots, x_m)$ mediante el aporte de los valores de la función conocida, que se conocen como *labels o targets* $y = f(x_1, x_2, \dots, x_m)$. Es aquí donde entra la supervisión. Una vez ajustada la función a nivel óptimo, lo que se pretende es la predicción del target de nuevos inputs.

Clasificación

La clasificación es el aprendizaje supervisado donde lo que se pretende predecir es una variable **categorica**.



IJC
International Journal of Cancer

Vitronectin and dermcidin serum levels predict the metastatic progression of AJCC I-II early-stage melanoma

Idola Ortega-Martinez¹, Jesús Gardeazabal^{2,3}, Asier Erramuzpe³, Ana Sanchez-Diez^{2,3}, Jesús Cortés^{1,3,4}, María D. García-Vázquez², Gorka Pérez-Yarza^{1,3}, Rosa Izu^{2,3}, Jose Luis Díaz-Ramón^{2,3}, Ildefonso M. de la Fuente⁵, Aintzane Asumendi^{1,3} and María D. Boyano^{1,3}

¹ Department of Cell Biology and Histology, Faculty of Medicine and Dentistry, University of the Basque Country (UPV/EHU), Leiza, Bizkaia, Spain

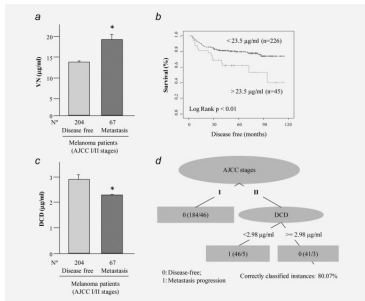
² Department of Dermatology, Ophthalmology and Otorhinolaryngology, Faculty of Medicine and Dentistry, University of the Basque Country (UPV/EHU), Leiza, Bizkaia, Spain

³ BioCrucis Health Research Institute, Plaza De Cruces S/N, Barakaldo, Bizkaia, Spain

⁴ Iberbasque: The Basque Foundation for Science, Bilbao, Spain

⁵ Institute of Parasitology and Biomedicine Lopez-Neyra, Parque Tecnológico Ciencias De La Salud, Avenida Del Conocimiento S/N, Armilla, Granada, Spain

Like many cancers, an early diagnosis of melanoma is fundamental to ensure a good prognosis, although an important proportion of stage I-II patients may still develop metastasis during follow-up. The aim of this work was to discover serum biomarkers in patients diagnosed with primary melanoma that identify those at a high risk of developing metastasis during the follow-up period. Proteomic and mass spectrophotometry analysis was performed on serum obtained from patients who developed metastasis during the first years after surgery for primary tumors and compared with that from patients who remained disease-free for more than 10 years after surgery. Five proteins were selected for validation as prognostic factors in 348 melanoma patients and 100 controls by ELISA: serum amyloid A and clusterin; immune system proteins; the cell adhesion molecules plakoglobin and vitronectin and the antimicrobial protein dermcidin. Compared to healthy controls, melanoma patients have high serum levels of these proteins at the moment of melanoma diagnosis, although the specific values were not related to the histopathological stage of the tumors. However, an analysis based on classification together with multivariate statistics showed that tumor stage, vitronectin and dermcidin levels were associated with the metastatic progression of patients with early-stage melanoma. Although melanoma patients have increased serum dermcidin levels, the REPTree classifier showed that levels of dermcidin <math>< 2.98 \mu\text{g/ml}</math> predict metastasis in AJCC stage II patients. These data suggest that vitronectin and dermcidin are potent biomarkers of prognosis which may help to increase the personalized medical care of melanoma patients and their survival.



Clasificación

La clasificación es el aprendizaje supervisado donde lo que se pretende predecir es una variable **categorica**.



RESEARCH ARTICLE

Predicting functional networks from region connectivity profiles in task-based versus resting-state fMRI data

Javier Raserio^{1,5}, Hannelore Aerts², Marisa Ontivero Ortega^{2,3}, Jesus M. Cortes^{1,4}, Sebastiano Stramaglia^{5,6*}, Daniele Marinazzo²

1 BioCrucis Health Research Institute, Hospital Universitario de Cruces, E-48903, Barakaldo, Spain, **2** Faculty of Psychology and Educational Sciences, Department of Data Analysis, Ghent University, Henri Dunantlaan 2, B-9000 Ghent, Belgium, **3** Neuroinformatics Department, Cuban Center for Neuroscience (CENEC), La Habana, Cuba, **4** Herbasque, The Basque Foundation for Science, E-48011, Bilbao, Spain, **5** Dipartimento di Fisica, Università degli Studi "Aldo Moro" Bari, Italy, **6** Istituto Nazionale di Fisica Nucleare, Sezione di Bari, Bari, Italy

* sebastiano.stramaglia@ba.infn.it



OPEN ACCESS

Citation: Raserio J, Aerts H, Ontivero Ortega M, Cortes JM, Stramaglia S, Marinazzo D (2018) Predicting functional networks from region connectivity profiles in task-based versus resting-state fMRI data. PLoS ONE 13(11): e0207385. <https://doi.org/10.1371/journal.pone.0207385>

Editor: Claudio Gallucci, Università degli Studi di Pisa, ITALY

Received: May 17, 2018

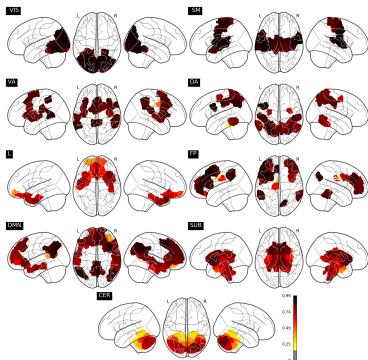
Accepted: October 29, 2018

Published: November 12, 2018

Copyright: © 2018 Raserio et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract

Intrinsic Connectivity Networks, patterns of correlated activity emerging from "resting-state" BOLD time series, are increasingly being associated with cognitive, clinical, and behavioral aspects, and compared with patterns of activity elicited by specific tasks. We study the reconfiguration of brain networks between task and resting-state conditions by a machine learning approach, to highlight the Intrinsic Connectivity Networks (ICNs) which are more affected by the change of network configurations in task vs. rest. To this end, we use a large cohort of publicly available data in both resting and task-based fMRI paradigms. By applying a battery of different supervised classifiers relying only on task-based measurements, we show that the highest accuracy to predict ICNs is reached with a simple neural network of one hidden layer. In addition, when testing the fitted model on resting state measurements, such architecture yields a performance close to 90% for areas connected to the task performed, which mainly involve the visual and sensorimotor cortex, whilst a relevant decrease of the performance is observed in the other ICNs. On one hand, our results confirm the cor-



Clasificación

- ¿Qué pasa ahora si nuestro target en vez de ser continuo (regresión) es discreto (clasificación), por ejemplo $y = \{0, 1\}$ después de codificar nuestros labels?
- Si ajustáramos un modelo de regresión, el output predicho sería continuo. Necesitamos discretizar dicho output imponiendo un *threshold* de asignación a clase, que nos daría una *regla de decisión*:

$$C_1 \text{ si } g(x) > \textit{threshold} \quad (1)$$

$$C_2 \text{ si } g(x) \leq \textit{threshold} \quad (2)$$

- La condición que separa ambas clases se conoce como *Decision boundary*. En esta región el label es ambiguo

$$B = \{x : g(x) = 0\} \quad (3)$$

Funciones discriminantes

- $g(x)$ se conoce como *función discriminante* y es la mejor manera de describir un clasificador.
- Cada clasificador implica una serie en particular de funciones discriminantes $g_y(x)$ para los diferentes labels $y = 1, 2, \dots, n_c$.
- $g_y(x)$ puede verse como una medida de probabilidad a la clase y por parte de x .
- De esta forma, la clase de x es asignada a aquella con mayor valor $g_y(x)$

$$C^{pred} = \operatorname{argmax} g_y(x) \quad (4)$$

Funciones discriminantes

Algunos ejemplos:

- Clasificador Lineal:

$$g_y(\mathbf{x}) = \beta_0 + \sum_{j=1}^m \beta_j x_j$$

- K-nn:

$$g_y(\mathbf{x}) = \sum_{i=1}^K I[(y_{i(k)} == y)]$$

- Nearest centroid:

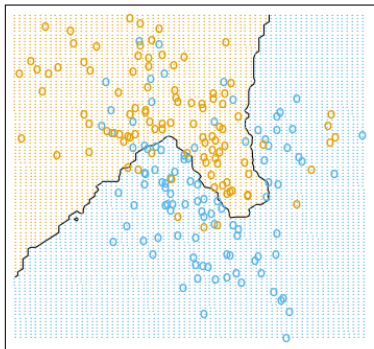
$$g_y(\mathbf{x}) = d(\mathbf{x}, \mu_y)$$

- Naive Bayes:

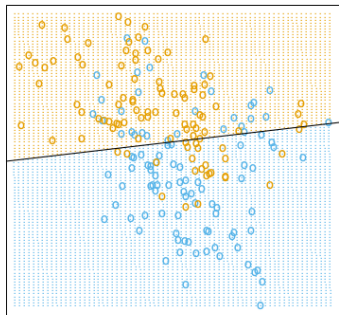
$$g_y(\mathbf{x}) = p(y|\mathbf{x})$$

Funciones discriminantes

15-Nearest Neighbor Classifier



Linear Regression of 0/1 Response



Caso lineal

Supongamos clasificación binaria, de tal forma que $y = \{0, 1\}$

- Definimos una sola función discriminante como

$$g(x) = g_{(y==0)} - g_{(y==1)} .$$

- Para el caso **lineal**

$$g(x) = \beta_0 + \sum_{j=1}^m \beta_j x_j \quad (5)$$

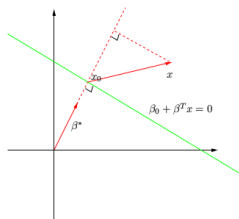
- Para este caso, la regla de decisión

$$y = 0 \text{ si } g(x) \geq 0 \quad (6)$$

$$y = 1 \text{ si } g(x) \leq 0 \quad (7)$$

Caso lineal: geometría de la decision boundary

- Para dos puntos x_a y x_b sobre esta boundary, se cumple que $\sum_{i=1}^m (x_a - x_b)^T \beta = 0$, luego los coeficientes $\beta \perp B$



- La distancia normal a B desde el origen, como cualquier x sobre B se cumple $g(x) = 0$

$$\rho = -\frac{\beta_0}{|\beta|}, \quad (8)$$

para $\rho = |x|$ con x sobre B.

- La distancia r de cualquier punto a la boundary B será además

$$r = \frac{g(x)}{|\beta|} \propto \frac{1}{|\beta|} \quad (9)$$

De la Regresión Lineal a Logistic Regression

Clasificación de la regresión lineal

Si recordamos del tema anterior, para la regresión lineal:

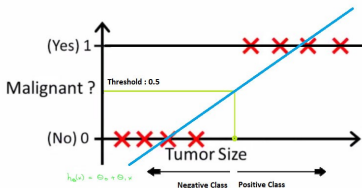
$$\begin{aligned}RSS &= \sum_{i=1}^N (y_i - f(x_{i1}, x_{i2} \dots, x_{im}))^2 \\ &= \sum_{i=1}^N (y_i - \beta_0 - \sum_{j=1}^m x_{ij}\beta_j)^2\end{aligned}$$

que minimizando daba

$$2X^T(X\beta - Y) = 0 \tag{10}$$

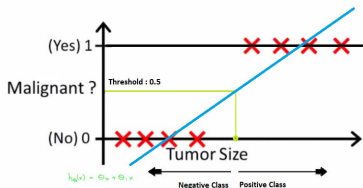
De linear Regression a Logistic Regression

- Ajustando usando la regresión lineal e imponiendo un threshold de 0.5, más o menos lo haría bien

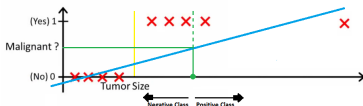


De linear Regression a Logistic Regression

- Ajustando usando la regresión lineal e imponiendo un threshold de 0.5, más o menos lo haría bien



- El problema viene cuando empieza a haber puntos muy diferentes.



from <https://towardsdatascience.com/understanding-logistic-regression-9b02c2aec102>

Logistic Regression

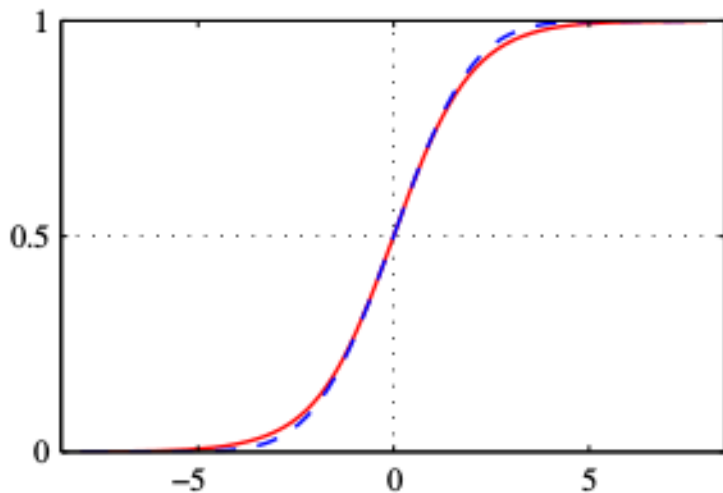
- Es un caso lineal $g(x) = \beta_0 + \sum_{j=1}^m \beta_j x_j$
- Discretizamos el output mediante una *función de activación* para que asintóticamente tienda a nuestro target:

$$y = f(g(x)) \quad (11)$$

- En logistic regression esto se realiza mediante el uso de la función sigmoide

$$f(x) = \frac{1}{1 + e^{-g(x)}} \quad (12)$$

Función logística



- De esta forma

$$g(x) > 0 \Rightarrow 1 > f(x) > 0.5 \text{ para } y = 1 \quad (13)$$

$$g(x) < 0 \Rightarrow 0 < f(x) < 0.5 \text{ para } y = 0 \quad (14)$$

- Por tanto, el output va a estar entre cero y uno, se puede interpretar como una probabilidad

$$p(y|x) = \frac{1}{1 + \exp(-(\beta_0 + \sum_j \beta_j x_j))} \quad (15)$$

Ajuste en Logistic Regression

- Estamos viendo el output como probabilidades. Podemos por tanto adoptar un maximum (log) likelihood estimation (MLE) para ajustar el modelo

$$L(\beta) = \sum_{i=1}^N \log(p_i(\mathbf{x}; \beta)) \quad (16)$$

- Por tanto, para el caso de logistic regression con dos clases

$$L(\beta) = \sum_{i=1}^N y_i \log(p(y_i|x)) - (1 - y_i) \log(1 - p(y_i|x)) \quad (17)$$

- Como en regresión lineal, los coeficientes β se calculan maximizando $L(\beta)$ (o minimizando $-L(\beta)$)

- **(NOTA):** el ajuste de mínimos cuadrados en regresión era un caso particular de MLE cuando se suponía que seguían una distribución normal.
- Si cada observacion $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ entonces $p_i \propto e^{-(y_i - \beta_0 - \sum_{j=1}^m \beta_j x_j)^2}$
- El likelihood es el producto de N gaussianas

$$L(\beta) = \prod_i p_i(x; \beta) \propto e^{-\sum_{i=1}^N (y_i - \beta_0 - \sum_{j=1}^m \beta_j x_j)^2}$$

$$\propto e^{-RSS}$$

maximizar $\mathcal{L}(\beta) \Leftrightarrow$ minimizar RSS
--

Regularización en Logistic Regression

Asimismo, también se puede añadir un regulador de los features

$$L(\beta) \rightarrow L(\beta) + F(\beta), \quad (18)$$

con la forma de F definiendo diferentes tipos de regularización;

- tipo L2 (Ridge) : $F(\beta) = C^{-1}\beta^T\beta$
- tipo L1 (Lasso) : $F(\beta) = C^{-1}|\beta|$

En scikit, se puede encontrar en **linear_model.LogisticRegression**

Support Vector Machine

Support Vector Machines

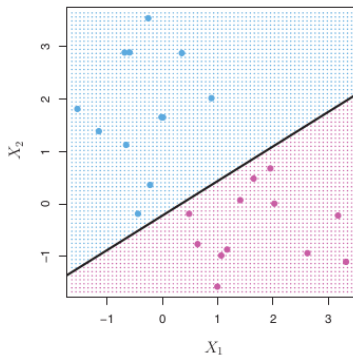
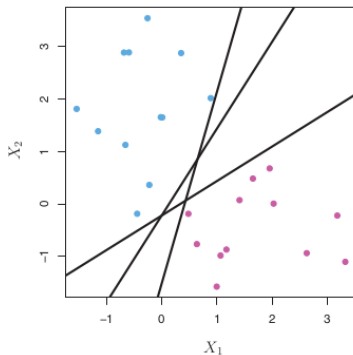
- Support Vector Machine permite, a diferencia de Logistic Regression, aprender funciones no lineales y más complejas
- Partamos de la función que minimizábamos en Logistic Regression:

$$L(\beta) = \sum_{i=1}^N y_i \log(p(y_i|x)) - (1 - y_i) \log(1 - p(y_i|x)) \quad (19)$$

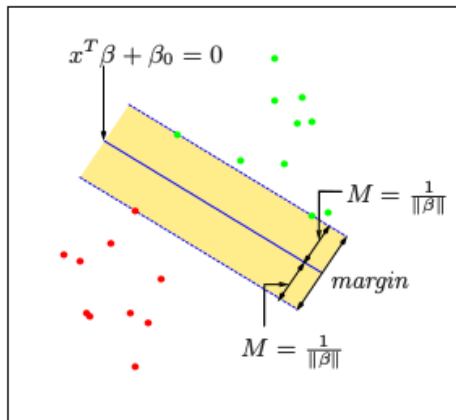
De aquí se ve que:

- Tanto cuando $y = 1$ y $x^T \beta \gg 0$ e $y = 0$ y $x^T \beta \ll 0$, el resultado es bastante fiable.
- En el resto de los casos, cerca de la decision boundary, no ocurre lo mismo, contribuyendo a ésta de manera importante

Lo interesante entonces sería que el resultado encontrado sea lo más fiable posible, o lo que es lo mismo, que podamos encontrar la boundary que separe en mayor medida las clases.



En support vector machine, se toman solo los puntos más cercanos a la boundary decision (support vectors) y se busca que la **distancia o margen** a esta boundary sea máxima



$\min \beta^T \beta$ sujeto a:

$$x^t \beta > 1 \text{ si } y = 1$$

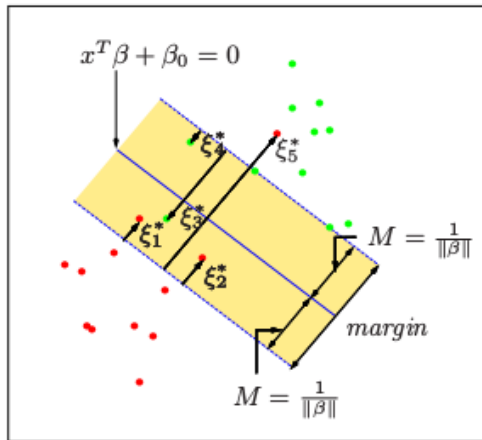
$$x^t \beta < -1 \text{ si } y = 0$$

clases no separables

- Esto que hemos presentado antes es para el caso en el que ambas clases son separables de manera clara.
- Podemos flexibilizar esta condición añadiendo algún penalti a la función de minimización:

$$C \sum_i \xi_i \quad (20)$$

- C actúa como parámetro de regularización que controla el balance entre la minimización con el ajuste y la complejidad del problema.



$$\min \beta^T \beta + C \sum_i \xi_i$$

sujeto a:

$$\xi > 0$$

$$x^t \beta > 1 - \xi \text{ si } y = 1$$

$$x^t \beta < \xi - 1 \text{ si } y = 0$$

SVM como clasificador no lineal

- Una propiedad importante y que convierte a SVM en un clasificador muy potente es que es capaz de crear boundaries de clasificación complejas.
- Para ello, en vez de ajustar x , lo hacemos en una función determinada $\phi(x)$. Es decir:

$$g(x) = \beta_0 + \sum_j \beta_j x_j \rightarrow \beta_0 + \sum_j \beta_j \phi(x_j)$$

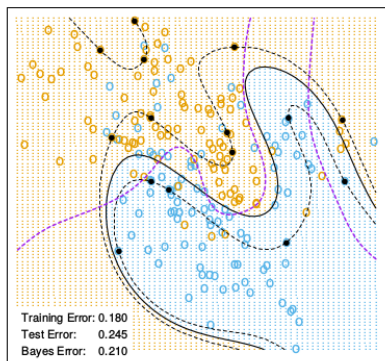
- De esta manera, la dimensión del espacio aumenta, haciendo que un clasificador no lineal se convierta en lineal

- La elección de $\phi(x)$ puede ser compleja
- Por suerte, todas las relaciones que involucran $\phi(x)$ lo hacen en forma de productos
- De esta forma, sólo es necesario especificar lo que se conoce como función *kernel*

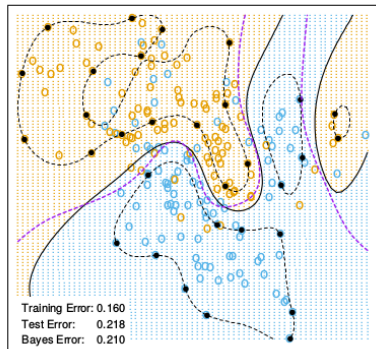
$$K(x, x') = h(x)^T \cdot h(x') \quad (21)$$

- Existen diferentes elecciones del kernel:
 - polinomio de grado d $K(x, x') = (1 + x^T x')^d$
 - Radial $K(x, x') = \exp(-\gamma(x - x')^T(x - x'))$
 - Neural Network $K(x, x') = \tanh(\kappa_1 \cdot x^T x' + \kappa_2)$

SVM - Degree-4 Polynomial in Feature Space



SVM - Radial Kernel in Feature Space



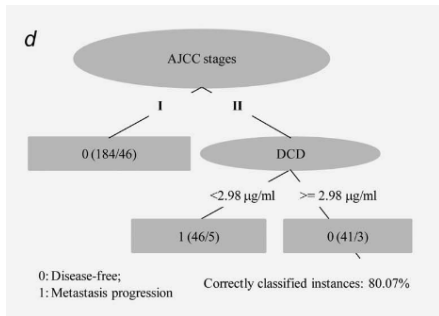
En scikit se puede encontrar todo esto en `svm.SVC`, `svm.LinearSVC`, `svm.NuSVC`

Árboles de decisión

Árboles de decisión

- Los árboles de decisión son un método de machine learning de tipo no paramétrico.
- Se basan en la partición del espacio de features en rectángulos y ajustar un modelo sencillo en cada uno.
- Dos métodos muy conocidos: C4.5 y CART. *scikit* usa una versión optimizada de este último, con el nombre **tree.DecisionTreeClassifier**

Árboles de decisión



Los elementos del árbol de decisión son los siguientes:

- Nodos internos que se van dividiendo
- Nodos hoja, que se encuentran al final del árbol y que determinan la clase
- Para cada nodo interno se asocia una función $Q_t(x)$
- Un condición de división

Árboles de decisión

Un árbol de decisión viene especificado por:

- La forma del node impurity $Q_t(x)$
- El criterio de subdivisión de cada nodo
- El criterio de parar de dividir
- El valor predicho en cada hoja nodo

critorio de subdivisión

- Dados la matrix de featur X y el vector de targets y
- Cada nodo t representa una region R .
- Para cada feature j y punto de partición s , se divide los datos en

$$R_{left}(j, s) = X|X_j < s$$

$$R_{right}(j, s) = X|X_j > s$$

- Para cada par (j, s) se calcula la impuridad total:

$$G(R, j, s) = \frac{n_{left}}{N_t} Q(R_{left}(j, s)) + \frac{n_{right}}{N_t} Q(R_{right}(j, s)) \quad (22)$$

- Se seleccionan aquellos parámetros que minimizan la función de arriba

$$(j, s)^* = \operatorname{argmin}_{j,s} G(R, j, s) \quad (23)$$

Valor predicho por nodo

- Dado un nodo t que contiene N_t observaciones

$$\hat{p}_{t,c} = \frac{1}{N_t} \sum I(y_i = c), \quad (24)$$

siendo c las diferentes clases que existen

- Las observaciones en ese nodo entonces se clasifican a la clase

$$c(t) = \operatorname{argmax}_c \hat{p}_{t,c}$$

Node impurity

Existen varias medidas $Q_t(x)$ de la impuridad del nodo:

- Gini index = $\sum_{c=1}^{n_c} \hat{p}_{t,c}(1 - \hat{p}_{t,c})$.
- Cross-entropy = $-\sum_{c=1}^{n_c} \hat{p}_{t,c} \log \hat{p}_{t,c}$.
- Misclassification error = $1 - \max(\hat{p}_{t,c})$

Scikit usa Gini index por defecto, aunque se puede cambiar.

Algunas ventajas:

- Simples de entender e interpretar
- requieren poco preprocesado de los datos
- Es computacionalmente escable (complejidad logarítmica)
- pueden manejar tanto datos categóricos como numéricos
- white box

Algunas desventajas:

- Suelen crear árboles muy complejos que no generalizan bien. Overfitting suele ocurrir.
- Suelen ser inestables ante pequeños cambios
- Pueden crear árboles biased hacia la clase dominante (class weight='balanced')

Random Forest

Random Forest

- El algoritmo Random Forest pertenece al grupo de tipo *ensemble*, en el cual se combinan algoritmos para mejorar la generabilidad y robusteza de un solo clasificador
- Dentro de este grupo se puede distinguir a su vez:
 - Métodos tipo Boosting, en el cual los clasificadores se van construyendo de manera secuencial, reduciendo el bias, combinando modelos débiles en uno más potente. En scikit: **ensemble.AdaBoostClassifier**, **ensemble.GradientBoostingClassifier**
 - Métodos de tipo averaging, en el que varios clasificadores se construyen individualmente y sus predicciones se promedian. De esa forma se reduce la varianza. En scikit: **ensemble.BaggingClassifier**
- Random forest pertenece a este último grupo, ya que como su nombre indica, se trata de un bosque de árboles de decisión. En scikit: **ensemble.RandomForestClassifier**

Random Forest

Random Forest funciona de la siguiente forma. Para cada árbol del bosque de N_{trees}

- 1 Toma de la data completa un bootstrap sample de cierto tamaño
- 2 Construcción el árbol de decisión sobre este sample.
- 3 Para evitar árboles correlacionados, en este proceso se suele tomar también un subset aleatorio de los features para dividir los nodos.
- 4 Output del conjunto de árboles de decisión.
- 5 Predicción de la clase mediante el promedio de la probablilidad predicha.

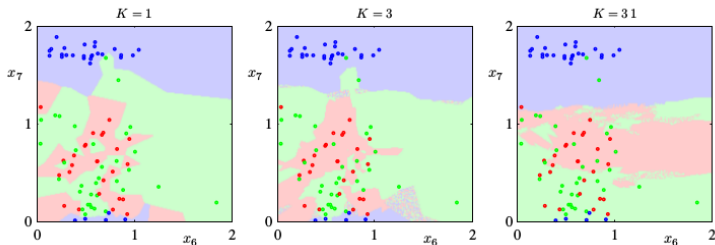
Lazy algorithms

Lazy algorithms

- Los algoritmos de tipo lazy, como su nombre indica, son aquéllos que no intentan construir una función general que discrimine entre clases, sino que sólo almacena los datos y predice en base a ellos.
- Como no modelizan una función, es difícil entender las relaciones entre los features y el target. Actúan como un *black box*.
- Pueden ser aún así muy buenos predictores

k-Nearest Neighbours

- Este algoritmo simplemente almacena las observaciones de los datos en el ajuste.
- La clasificación se calcula simplemente por la clase mayoritaria de entre sus k vecinos.



En scikit: **`neighbors.KNeighborsClassifier`**

Métricas de clasificación

Métricas de clasificación

La métrica por excelencia y defecto es el accuracy, que mide el ratio de observaciones bien clasificadas

$$acc = \frac{1}{N} \sum_{i=1}^N I(y_i^{pred} = y_i^{true}) \quad (25)$$

En scikit: **metrics.accuracy_score**

Métricas de clasificación

- A veces, nos conviene ver en más detalle otras métricas viendo las relaciones entre las predicciones y los valores reales para las diferentes clases. Esto se consigue mediante la confusion matrix.
- Para el caso binario:

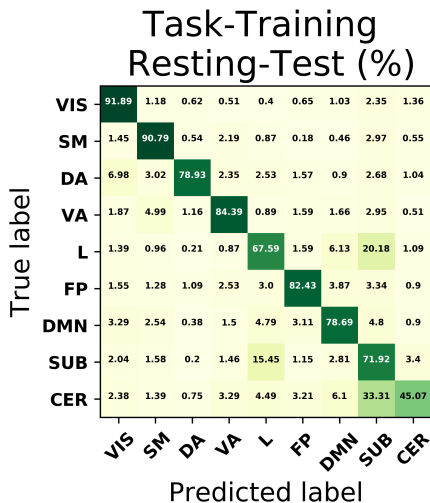
		Prediction	
		+	-
True class	+	TP (true positives)	FN (false negatives)
	-	FP (false positives)	TN (true negatives)

donde $P = TP + FN$ es el número total de observaciones de la clase positiva y $N = TN + FP$ el número total de observaciones de la clase negativa

En scikit: **`metrics.confusion_matrix`**

Métricas de clasificación

Es muy util para el caso multiclass



Métricas de clasificación

A partir de la anterior matriz es posible calcular otras métricas interesantes

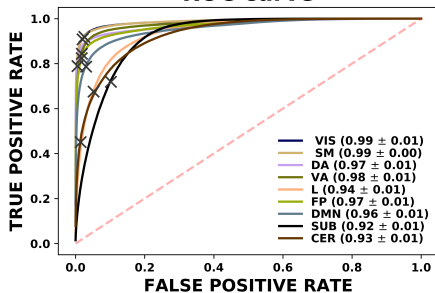
- False Positive Rate (FPR): FP/N
- True Positive Rate (TPR): TP/P
- Precision : $\frac{TP}{TP+FP}$. En scikit **metrics.precision_score**
- Recall : $\frac{TP}{P}$. En scikit **metrics.recall_score**
- F1: $2 \text{ Precision Recall} / (\text{Precision} + \text{Recall})$. En scikit **metrics.f1_score**

Métricas de clasificación

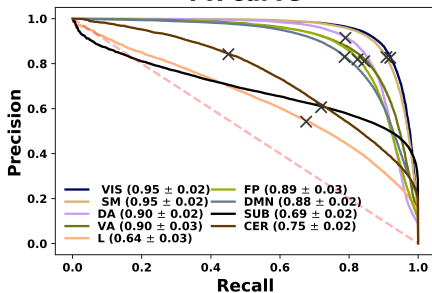
- Otra métrica muy utilizada son las ROC y PR Curves.
- A diferencia de las métricas anteriores, aquí se calcula el rendimiento del clasificador a medida que cambiamos el threshold de discriminación para la asignación de clase.
- ROC curve representa el TPR frente a FPR. En **`metrics.roc_curve`**
- PR curve representa la Precisión frente al Recall. En **`metrics.roc_curve`**
- En ambos casos, área bajo la curva mide el rendimiento del clasificador. En scikit **`metric.roc_auc_score`** y **`metric.average_precision_score`**

Métricas de clasificación

ROC curve



PR curve



Resumen: Clasificadores (algunos) en scikit

- módulo ***Linear_model***:
 - Logistic Regression **LinearRegression**
- módulo ***svm***:
 - SVC
 - LinearSVC
 - NuSVC
- módulo ***tree***:
 - DecisionTreeClassifier
- módulo ***ensemble***:
 - AdaBoostClassifier
 - GradientBosstingClassifier
 - RandomForestClassifier
- módulo ***neighbors***:
 - KNeighborsClassifier

Resumen: Métricas de clasificación en scikit

En el módulo *metrics*:

- Accuracy: **accuracy_score**
- Matriz de confusión : **confusion_matrix**
- Precisión: **precision_score**
- Recall (o Sensitividad): **recall_score**
- F1 score: **f1_score**
- Área bajo la curva ROC: **roc_auc_score**
- Área bajo la curva PR: **average_precision_score**