desconocidos y queremos ver qué relación tiene con medidas tales como: la edad de la gestante, si es primeriza o no, etc. A todas gestantes del mundo se le llama en estadística La población y denotaremos por P. Pero como no podemos ir una a una y seguirla para saber el tiempo de gestación, porque es imposible para nosotros, entonces la damos como desconocida.

Para poder realizar el estudio, tomamos una cantidad S de gestantes. Estas gestantes se toman de manera aleatoria. Al grupo de estas gestantes se les denomina Muestra. Entonces el problema sería: predecir el tiempo de gestación que llamaremos y, a partir de un conjunto de rasgos, que llamaremos x tomados de una muestra que llamamos S.

$h_s : X \rightarrow y$ Entonces, se trata de encontrar un mapeo o función que relacione la variable y o tiempo de gestación, con los rasgos X, que son la edad, si fuma, hemoglobina, entre otros. Todos los datos que se recopilan con estas gestantes se tabulan y se conforma un dataset. Que lo usaremos para entrenar nuestros modelos.

Matemáticamente, este mapeo toma la forma siguiente:

$$h_s : X \rightarrow y$$

Es a esta hs, es a la que le llamamos función hipótesis y se escoge de entre varias disponibles para un método determinado, a esta clase de hipótesis la llamamos H. Revisemos las asunciones iniciales que hemos hecho hasta aquí:

- Hemos supuesto que las chicas que escogimos todas son normales.
- Que las hemos escogido de manera aleatoria.
- Que todas provengan de la misma población.
- Asunciones sobre los rasgos:
- Que los rasgos son representativos de lo que queremos medir.

© Universidad Internacional de La Rioja (UNIR)

- Que los casos de la muestra representan bien a la población.

Con todos estos elementos formulemos el problema:

La tarea de es aprender a predecir la salida $y \in Y$ a partir de un conjunto de pares (x,y) que pertenecen a una distribución conjunta D. Para nosotros desconocida. Este problema consiste en aprender una función o mapeo. A partir de un dataset de entrenamiento, finito S, compuestos por m muestras de D, estas m muestras deben estar Independiente e idénticamente distribuidas.

A $h_s \in H$, la función aprendida, también se le conoce como hipótesis, la cual es elegida del conjunto de hipótesis o funciones que soporta el modelo. Para evaluar la calidad de lo aprendido por medio de la hipótesis $h_s \in H$ se una una función llamada normalmente Riesgo o perdida.

La función de esta función de pérdida o riesgo es cuantificar de alguna forma el error entre lo aprendido por la función o mapeo y el proceso real que generó el dato. En nuestro caso, la función de pérdida nos diría el error esperado entre los valores del tiempo de gestación de las gestantes y el valor real si conociéramos el proceso natural que decide este tiempo. Pero este proceso natural no lo conocemos. Por tanto, la única cosa que podemos hacer es usar el conjunto de entrenamiento con el fin de evaluar dicho error.

El objetivo, entonces, de los métodos de aprendizaje supervisado es minimizar el riesgo: $minR(h_s) = E(x,y) \sim l(h(x),y)$.

Esto conduce a lo que se conoce como minimización del riesgo empírico donde se pretende aprender la hipótesis h minimizando R(h). Ahora bien, no basta que la función aprendida h, prediga correctamente los casos que se usaron para derivarla, es necesario que esta hipótesis sea capaz de predecir correctamente casos que no estaban dentro de conjunto de entrenamiento. Es decir, si tomamos cualquier gestante que no esté dentro del grupo que elegimos con el fin de calcular la h, ¿esta

© Universidad Internacional de La Rioja (UNIR)

función sería capaz de predecir el tiempo de gestación de forma correcta?

Es a esta cualidad de h a la que se le denomina capacidad de generalización. Pero ¿qué puede ir mal cuando minimizamos el riesgo basado en el conjunto de entrenamiento en vez de usar el conjunto de todas las gestantes del mundo?

Aquí pueden pasar dos cosas, una que la h aprenda y memorice tan detalladamente los casos del conjunto de entrenamiento que luego no pueda generalizar de forma correcta y por otro lado puede pasar que h no aprenda muy bien y por tanto tampoco pueda.

Veamos esto en más detalles, veamos las imágenes siguientes:

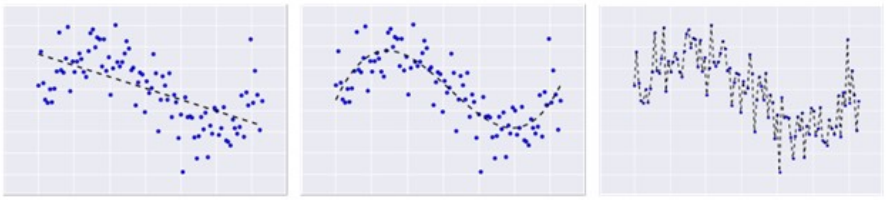


Figura 3. En la izquierda la función h no se ajusta a los datos, en el centro se nota algún ajuste, en la derecha la función se sobreajusta a los datos. Imágenes tomadas de On the Bias-Variance Tradeoff: Textbooks Need an Update. Brady Neal.

En el caso de la imagen de la izquierda la complejidad de la función h, no permite que se ajuste a la complejidad de los datos y esto provoca que cometa muchos errores tanto en entrenamiento como en generalización. A esto se le conoce como: Bajo Ajuste, desajuste o UnderFitting.

En el caso de la imagen del centro, la función h, al parecer se ajusta a los datos, arrojando una baja cantidad de errores en el conjunto de entrenamiento y relativamente bajos errores en un conjunto de prueba de la generalización. A esto se le conoce como ajuste o se dice que la hipótesis se ajusta a los datos.

En la imagen de la derecha la hipótesis h, se ajusta demasiado a los datos, incluso a

© Universidad Internacional de La Rioja (UNIR)

los ruidos, provocando que prácticamente arroje cero errores en los conjuntos de entrenamiento, pero un elevado número de errores en los conjuntos de prueba de generalización. A esta situación se le llama Overfitting o sobreajuste de los datos.

Entonces podemos definir ahora que se entiende por bias y varianza:

Se define como Bias o sesgo, a la diferencia entre el valor esperado de una predicción hecha por nuestra hipótesis h de cualquier valor x, no visto anteriormente por ella y la verdadera función generadora de los casos.

Matemáticamente la representaremos así:

$$Bias = E\left(h_s\left(X\right)\right) - f\left(X\right)$$

Donde, hs es la hipótesis que aprendimos usando el conjunto de entrenamiento s, E es el valor esperado y f(x) es la función que obtendríamos si entrenáramos nuestros modelos con la población completa, cosa que es imposible.

En esencia el Bias de un modelo nos habla de cuan desviado anda el modelo de la realidad. El valor esperado de una variable aleatoria, por otro lado, es el promedio los valores que admite y su forma matemática es la que sigue:

$$E\left(x\right) = \sum_{i=1}^{n} x_i P\left(x_i\right) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Si la variable es continua se sustituye la sumatoria por una integral. P(xi) es la probabilidad de xi.

La Varianza de una hipótesis h, es la medida que caracteriza la variación alrededor de la tendencia central de la hipótesis.

La forma matemática que toma es la siguiente:

$$Var\left(h_s\right) = E_s\left[\left(h_s\left(x\right) - E_s\left(h_s\left(x\right)\right)\right)^2\right]$$

© Universidad Internacional de La Rioja (UNIR)

Hay un gráfico que puede que nos aclare de manera visual y nos permita desarrollar un tanto la intuición detrás de estos conceptos:

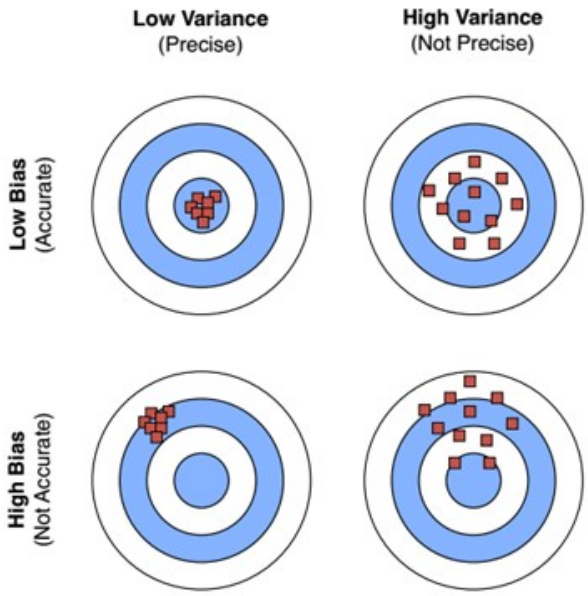


Figura 4. Gráfico para ayudar entender los conceptos de Bias y Varianza. Imágen tomada de STAT 479: Machine Learning Lecture Notes. Sebastian Raschka.

Los conceptos de Bias y varianza de los modelos de aprendizaje automático están también ligados a los conceptos de sobreajuste o overfitting y al de bajoajuste, desajuste o underfitting.

También están estrechamente ligado al concepto de complejidad de la hipótesis que escogemos para aprender y el de error. Históricamente se ha dicho que en la medida que escojamos hipótesis más complejas para un determinado problema corremos el riesgo de que el modelo incurra en overfitting y viceversa, si la complejidad de este no es suficiente pues corremos el riesgo de que incurra en underfitting. Esto nos lleva directamente a lo que se conoce como el compromiso Bias/Varianza.

Para ilustrar este tipo de compromiso, por favor vean la imagen en la Figura 5.

© Universidad Internacional de La Rioja (UNIR)

No vamos a profundizar aún más en estos elementos y creemos que se han aclarado los conceptos Bias, Varianza, over y under fitting, así como el de compromiso Bias-Varianza.

Lo práctico a sacar de aquí, grosso modo, es, ¿cómo sé que mi modelo adolece de overfitting?, bueno probablemente este arrojándo un muy bajo valor de error en el conjunto de entrenamiento y un muy alto con el de prueba. ¿cómo sé que está haciendo underfitting? Lo normal en underfitting es que los valores de error sean grandes en ambos conjuntos.

Cuando el modelo genera valores de error relativamente bajos en ambos hay que pensar que vamos por el camino correcto.

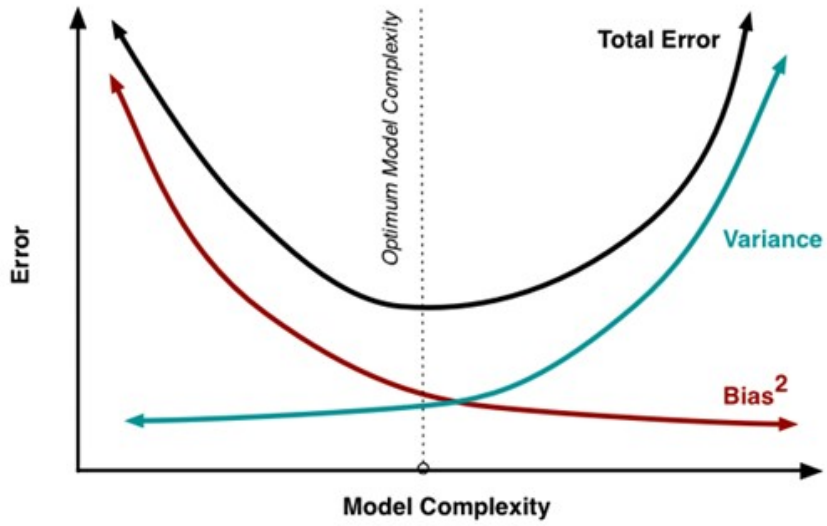


Figura 5. Compromiso Bias/Varianza. Imágenen tomadas de On the Bias-Variance Tradeoff :Textbooks Need an Update. Brady Neal.

Con el fin de poder probar la capacidad de generalización de nuestros modelos, lo común, es que el conjunto de datos se divida en dos, una para el entrenamiento mientras que el otro se use para pruebas; pero es suficiente?

Antes de ver las métricas que usamos para evaluar cada uno de nuestros modelos

© Universidad Internacional de La Rioja (UNIR)

es necesario que aclaremos lo referente a la Validación Cruzada. ¿Qué es? Y ¿para qué se usa?

© Universidad Internacional de La Rioja (UNIR)

Tema 3. Algoritmos de aprendizaje automático supervisado

Validación cruzada

El objetivo que se persigue al entrenar los algoritmos de aprendizaje automático es obtener modelos que sean capaces de predecir de forma correcta casos que no estén incluidos en los que se usaron para su entrenamiento.

El primer instinto, a partir de lo que hemos ido aprendiendo, es dividir el dataset en dos, una parte, normalmente el 75% o 80% de los datos para entrenar y el resto para probar la capacidad de generalización del modelo. Aunque esto es correcto, tiene un par de inconvenientes: Al dejar una parte de los datos fuera perdemos información que puede ser vital para la construcción del modelo y como entrenamos y validamos sobre conjuntos fijos tendemos a sobre valorar la calidad de nuestros modelos, pues los valores de errores dependen de los conjuntos que elijamos. Si, separamos de nuevo los conjuntos y entrenamos de nuevo, probablemente obtendremos otros valores que puede que sean mejores o peores.

Para aliviar ese problema se desarrolló la técnica de validación cruzada. En este tipo de esquema el conjunto completo se usa para todo, es decir para entrenar, validar y probar. Aunque existen varios esquemas, solo explicaremos en llamado K-Fold.

Validación Cruzada usando K-Fold

En este esquema, el parámetro k define la subdivisión del conjunto de datos en entrenamiento y prueba y la cantidad de veces que se entrenará y probará el modelo. Esto quiere decir que este esquema divide el conjunto de entrenamiento en k partes y se entrenará y probara k veces nuestro modelo usando K-1 subdivisión del conjunto para entrenar y una para probar y al final los valores de errores obtenidos se promedian.

© Universidad Internacional de La Rioja (UNIR)

Figura 6. Validación cruzada con k=5.

En la imagen se denota con el color azul, aquellos subconjuntos que se usarán como conjuntos de prueba. Todos los errores de cada ciclo de entrenamiento/prueba se promedian y se dan como un solo valor que caracteriza la calidad del modelo.

Con estos conceptos a mano, podemos entender las métricas que se usan para medir la calidad de los modelos generados por los algoritmos de aprendizaje automático.

A continuación veremos las métricas correspondientes a los algoritmos de regresión y luego a los de clasificación.

Métricas para la evaluación de los algoritmos de regresión

Existen diferentes métricas de error que se pueden aplicar en un problema de regresión para obtener la función f(x) ideal. Las más comunes o habituales y sus definiciones matemáticas son:

- **Error medio cuadrático o *mean square error* (MSE):** se define como la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y_i}\right)^2$$

© Universidad Internacional de La Rioja (UNIR)

- **Error medio absoluto, *mean absolute error* (MAE):** se define como la diferencia en valor absoluto entre el valor real y el valor predicho.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{y_i}|$$

- **Raíz del error cuadrático medio o *root mean square Error* (RMSE):** se define como la raíz cuadrada de la media de la diferencia entre el valor real y el valor predicho o estimado al cuadrado.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(y_i - \widehat{y_i}\right)^2}$$

Esta métrica comparada con el error absoluto medio (MAE) amplifica y penaliza los errores grandes. Por otro lado, en MAE cada error contribuye al total del error en función de su valor absoluto.

- **Logaritmo de la raíz del error cuadrático medio, *root mean logarithmic square error* (RMLSE):**

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} \left(\log\left(y_i + 1\right) - \log\left(\widehat{y_i} + 1\right)\right)^2}$$

Una de las mejores formas de evaluar un modelo de regresión es utilizando gráficos. Uno de los gráficos que más información proporcionan visualmente es un diagrama de dispersión de dos dimensiones donde el eje x son los valores reales y el eje y los valores predichos o estimados (o viceversa).

En el siguiente gráfico se muestra el gráfico de dispersión del ejemplo de la producción de trigo y precio de la harina.

© Universidad Internacional de La Rioja (UNIR)

Cada uno de los puntos corresponde a una de las estimaciones o predicciones realizadas. En el caso de que las predicciones o estimaciones fueran perfectas, caerían directamente sobre la diagonal del gráfico. Por otro lado, los puntos que caen por encima de la diagonal son sobre estimaciones de las predicciones, mientras que los puntos que caen por debajo de la diagonal son predicciones que se quedan cortas.

Una métrica común que se utiliza para medir la dispersión en este tipo de diagramas es el coeficiente de correlación, el cual cuantifica la relación lineal entre dos variables. Este coeficiente toma valores entre -1 y 1, donde 1 implica correlación lineal positiva de forma completa, -1 correlación lineal negativa de forma completa; y cuanto más cercano de 0 sea el valor menor correlación entre las dos variables.

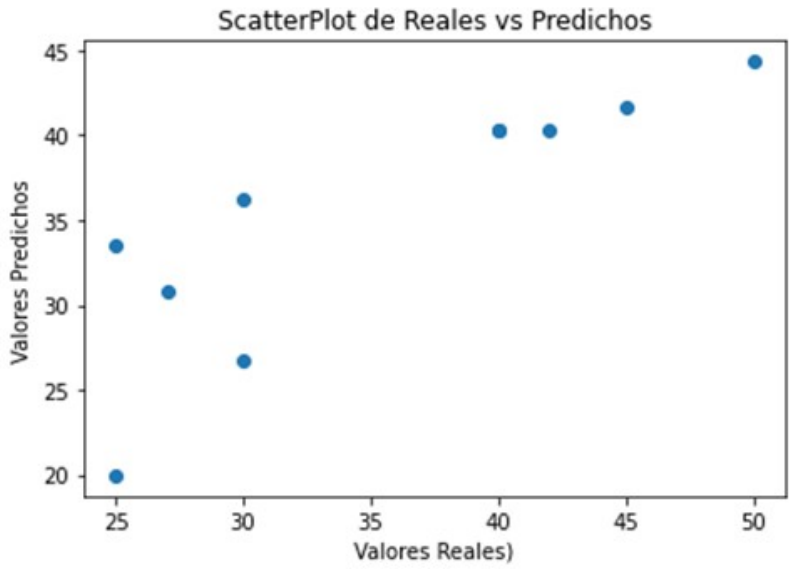


Figura 7. Gráfica de dispersión (ScatterPlot) de los valores predichos vs reales el precio de la harina en el ejemplo inicial de esta lección.

Una correlación lineal positiva implica que cuando el valor en el eje x crece, el valor en el eje y lo hace en la misma proporción. Por el contrario, una correlación lineal negativa implica que cuando el valor en el eje x crece el valor en y decrece de forma proporcional. Esta relación se define matemáticamente con la siguiente función.

© Universidad Internacional de La Rioja (UNIR)

Esta bondad de ajuste también se puede medir utilizando el coeficiente de determinación o R-Cuadrado. Este coeficiente indica la fracción de la variación explicada por la recta de regresión respecto a la variación total. Coincide con el cuadrado del coeficiente de correlación y puede tomar valores entre 0 y 1, siendo 1 el mejor ajuste y 0 el peor. En el ejemplo que hemos estado usando, este toma valor de 0.7176465181664973.

Métricas de evaluación de los algoritmos de clasificación

Dentro del aprendizaje supervisado, a continuación de los algoritmos de regresión, el siguiente gran grupo de algoritmos son los de clasificación.

A diferencia de los algoritmos de regresión cuyo objetivo es predecir un valor numérico, los algoritmos de clasificación tienen como objetivo obtener la clase más probable para cada una de las instancias.

Este tipo de técnicas pueden utilizarse para predecir o estimar la probabilidad de pertenencia a una clase de entre dos posibles, lo que se conoce como clasificación binaria. Por otro lado, también se pueden utilizar para predecir o estimar la probabilidad de pertenencia de una clase de entre varias posibles (más de dos), lo cual se conoce como clasificación multiclase.

A continuación, veremos las métricas de evaluación que se utilizan centrándonos en la matriz de confusión, precisión/recall, accuracy/sensitivity y f-measure.

Posteriormente, veremos las curvas ROC y la métrica de área bajo la curva.

Las métricas clásicas de evaluación de los clasificadores se extraen de la llamada Matriz de confusión. Veamos en detalle en que consiste esta matriz y para eso nos apoyaremos en la siguiente imagen:

© Universidad Internacional de La Rioja (UNIR)

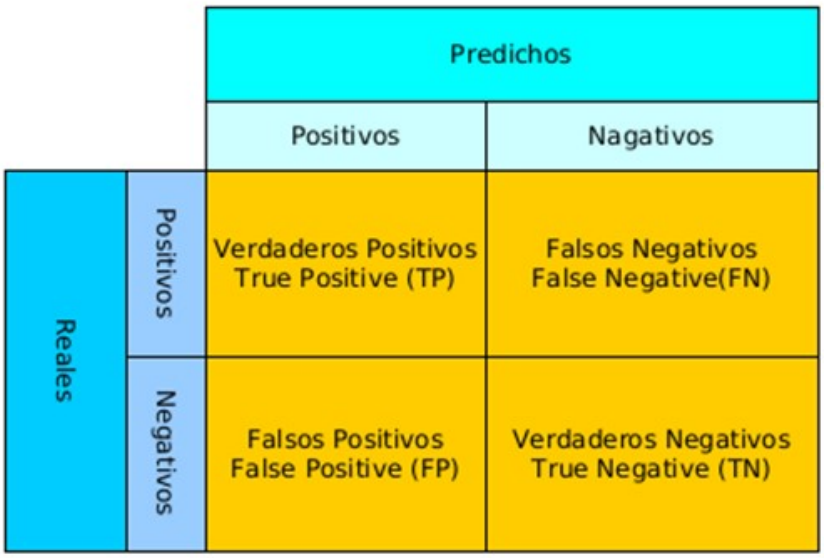


Figura 8. Matriz de Confusión.

Que significa cada uno de estos cuadrantes:

- True Positive(TP): Son los casos que en realidad son positivos y que el clasificador los clasificó como tal.
- True Negative (TN): Son los casos que en realidad son negativos y que el clasificador los clasificó de forma correcta.
- False Negative (FN): Son los casos que son positivos y el clasificador los clasifica como negativos.
- False Positive (FP): Son casos que en realidad son negativos y el clasificador los clasifica como positivos.

Veamos ahora las métricas que se derivan de esta matriz.

La primera de las métricas en la llamada Precisión o Accuracy está interpretado como el porcentaje total de casos clasificados correctamente. Esta medida toma valores en 0 y 1 y entre más alta es mejor.

© Universidad Internacional de La Rioja (UNIR)

Su interpretación matemática es la siguiente:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Es decir, esta medida son todos los clasificados correctamente contra el total. Aunque esta es una medida muy intuitiva y de las más fáciles de usar hay que tener cuidado con su uso.

Por ejemplo, si estamos tratando de predecir una enfermedad y las muestras de cada una de las clases está desbalanceada, esta medida nos puede arrojar una accuracy muy alta aunque sea completamente inútil. El problema es que esta métrica siempre favorece a la clase más poblada. En estos casos es aconsejable usar otras métricas.

Recall, también llamada sensibilidad:

$$Recall = \frac{TP}{TP + FN}$$

Precisión:

$$Precisión = \frac{TP}{TP + FP}$$

Especificidad:

$$Especificidad = \frac{TN}{TN + FP}$$

Medida F1:

$$F_1 Score = 2\frac{precision * recall}{precision + recall}$$

Tasa de falsos positivos *(false positive rate):*

$$FalsePositiveRate\,(FPR) = \frac{FP}{TN + FP}$$

© Universidad Internacional de La Rioja (UNIR)

Tasa de verdaderos positivos *(true positive rate):*

$$TruePositiveRate\,(TPR) = \frac{TP}{TP + FN}$$

ROC: ROC es la curva Característica Operativa del Receptor (en inglés, Receiver Operating Curve), también conocida como ROC, es una representación gráfica de la sensibilidad frente a la especificidad según se varía el umbral. Es una curva que en en eje x se plotean los valores de False Positive Rate, mientras que en él y se plotean los valores correspondientes a True Positive Rate.

En la siguiente imagen se muestra un ejemplo de una curva ROC:

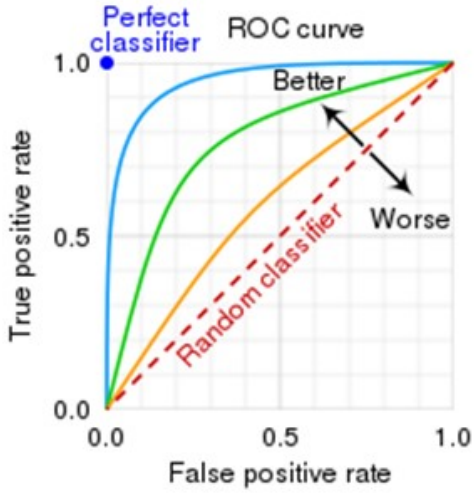


Figura 9. Forma de interpretar una gráfica ROC. Imagen tomada de Wikipedia.

Algoritmos de regresión

Como ya hemos comentado a lo largo de las lecciones anteriores la regresión es la tarea que pretende explicar el efecto que tienen las variables, llamadas independientes sobre la variable dependiente. Y también, se ha explicado que se da, cuando necesitamos predecir una variable que es continua en su naturaleza.

Si la dependencia de la variable a predecir u Outcome, es de una sola variable explicativa o independiente, entonces podemos hablar de una regresión lineal simple.

© Universidad Internacional de La Rioja (UNIR)

Si por el contrario, la variable dependiente, está relacionada con varias variables independientes o explicativas, nos podemos referir a este tipo de regresión como Regresión Lineal Múltiple.

Si la relación que se espera entre las variables dependientes e independientes es lineal, entonces la regresión se llama Lineal; pero si el tipo de relación que se espera no es de naturaleza lineal, entonces la regresión puede ser polinomial o multinomial.

Por tanto, hagamos una especie de clasificación de los métodos de regresión con el fin de entender su taxonomía. Si tenemos en cuenta la cantidad de variables relacionadas con la dependiente, entonces la regresión se puede clasificar como:

- Regresión lineal simple.
- Regresión lineal múltiple.

Si la relación que se espera entre variables dependientes e independientes es lineal o no, se pueden clasificar en:

- Regresión Lineal.
- Regresión Polinomial.

Hay muchas otras formas de clasificar los métodos de regresión si tomamos en cuenta uno u otro aspecto de sus características, en realidad hay muchos métodos de regresión y no es nuestro interés hacer una taxonomía cerrada de todos ellos. Solo destacar las principales ramas y las que acabamos de dar son las más notables.

A continuación haremos un listado de algunos de los métodos de regresión y una breve explicación de sus fundamentos, tampoco es nuestro interés hacer un listado exhaustivo de todos los métodos.

© Universidad Internacional de La Rioja (UNIR)

Algunos métodos de Regresión:

- Regresión Lineal.
- Ridge (Ridge Regression).
- Lasso.
- Elastic-Net.
- LAR (Least Angle Regression).
- OMP (Orthogonal Matching Pursuit).
- Bayesian Regression.
- Automatic Relevance Determination – ARD.
- Logistic regression.
- Generalized Linear Regression.
- Stochastic Gradient Descent – SGD.
- Passive Aggressive Algorithms.
- RANSAC: RANdom SAmple Consensus.
- Quantile Regression.
- Polynomial regression.

Los métodos que acabamos de listar, son modelos que implementados en el módulo linear_model de scikit-learn, Lo hemos hecho con toda intención, pues además de ser un paquete muy popular, les permite aplicarlo de manera inmediata a sus propios proyectos o simplemente coquetear con ellos para ganar en comprensión.

© Universidad Internacional de La Rioja (UNIR)

De manera general, los métodos de regresión esperan que la variable objetivo o dependiente sea explicada o dependa de una combinación

lineal de las variables independientes o rasgos. La notación matemática sería la siguiente:

$$\widehat{y}\left(\beta,x\right)=\beta_0+\beta_1x_1+...\beta_nx_n$$

A los $\beta_0,\beta_1,...\beta_n$ se les llama coeficientes del modelo y son desconocidos. El método clásico de estimar estos coeficientes se llama Mínimos Cuadrados. La mayoría de las variantes de regresión se basan en variantes o métodos diferente de estimar estos coeficientes.

La idea detrás de los mínimos cuadrados es encontrar una combinación de estos coeficientes que minimice la Suma de los residuos cuadrados.

$$RSS=\sum_{i=1}^{m}\left(\widehat{y}-y\right)^2$$

Regresión lineal

Este es el tipo de regresión con la comenzamos esta lección. Específicamente con la Regresión lineal simple, en la que la variable dependiente y está relacionada con una sola variable explicativa y se supone que la naturaleza de esta relación sea lineal.

La función de costo de este tipo de regresión es llamada Mínimos Cuadrados Ordinarios y la estimación de β_0,β_1 es como sigue:

$$\beta_1=\frac{Cov\left(x,y\right)}{{S_x}^2}=\frac{\sum_{i=1}^{n}\left(x_i-\bar{x}\right)\left(y_i-\bar{y}\right)}{\sum_{i=1}^{n}\left(x_i-\bar{x}\right)}$$

Mientras que la intersección se calcula de la siguiente forma:

$$\beta_0=\bar{y}-\widehat{\beta_1}\bar{x}$$

© Universidad Internacional de La Rioja (UNIR)

En la Figura 1, se muestra un ejemplo de implementación de de una regresión linear simple en Python.

La diferencia entre este tipo de regresión y la regresión lineal múltiple es la cantidad de predictores que se relacionan con la variable dependiente. Por tanto no la explicaremos.

Asunciones del modelo:

- Ausencia de multicolinealidad o autocorrelación entre las variables explicativas o rasgos.
- Los casos deben ser independientes e idénticamente distribuidos.
- No pueden haber Outliers.
- Entre otros.

Ridge regression

Este tipo de regresión resuelve alguno de los problemas de la regresión lineal y regresión lineal múltiple. Específicamente los de los mínimos cuadrados ordinarios. Especialmente en escenarios donde la variables están altamente correlacionadas.

En este tipo de regresión se modifica el método de los mínimos cuadrados ordinarios añadiendo un término de regularización llamado norma L2. Esto hace que se penalicen los coeficientes en dependencia de su tamaño.

La función de costo o pérdida de este tipo de regresión es:

$$Loss_{Ridge} = \sum_{i=1}^{N} (\hat{y} - y)^2 + \alpha \sum_{j=1}^{m} (\beta_j)^2$$

El parámetro a tener en cuenta en este tipo de regresión es el alfa.

© Universidad Internacional de La Rioja (UNIR)

Mientras mayor es el parámetro de complejidad alfa, el método se hace más robusto a la colinealidad. Si este alfa se hace igual a cero, el método se hace equivalente a una regresión lineal típica.

La forma de escoger el alfa que recomendamos es por medio de Validación Cruzada.

```
>>> import numpy as np
>>> from sklearn import linear_model
>>> reg = linear_model.RidgeCV(alphas=np.logspace(-6, 6, 13))
>>> reg.fit([[0, 0], [0, 0], [1, 1]], [0, .1, 1])
RidgeCV(alphas=array([1.e-06, 1.e-05, 1.e-04, 1.e-03, 1.e-02, 1.e-01, 1.e+00, 1.e+01,
      1.e+02, 1.e+03, 1.e+04, 1.e+05, 1.e+06]))
>>> reg.alpha_
0.01
```

Ejemplo de Ridge Regression en Python.

Lasso regression

Lasso significa **Least absolute shrinkage and selection operator** y fue introducido para mejorar la calidad de la predicción e interpretabilidad de los modelos de regresión. Al igual que Ridge introduce un término de regularización en su función de pérdida; pero en este caso en forma de norma L1.

$$Loss_{Lasso} = \sum_{i=1}^{N} (\hat{y} - y)^2 + \alpha \sum_{j=1}^{m} |\beta_j|$$

Al igual que Ridge, el cuándo alfa es cero, el modelo se comporta como lo hace los mínimos cuadrados ordinarios. En cuanto crece alfa, se van haciendo cero algunos coeficientes y teóricamente con alfa infinito todos los coeficientes se harían cero.

```
>>> from sklearn import linear_model
>>> reg = linear_model.Lasso(alpha=0.1)
>>> reg.fit([[0, 0], [1, 1]], [0, 1])
Lasso(alpha=0.1)
>>> reg.predict([[1, 1]])
array([0.8])
```

Ejemplo de Lasso Regression.

© Universidad Internacional de La Rioja (UNIR)

Elastic-net Regression

En este tipo de regresión se incluye dos términos de regularización, la norma L1 y la norma L2.

La función de pérdida queda de la siguiente forma:

$$Loss_{Elasticnet} = \frac{1}{2n}\sum_{i=1}^{N}(\hat{y}-y)^2 + \alpha\rho\sum_{j=1}^{m}|\beta_j| + \frac{\alpha(1-\rho)}{2}\sum_{j=1}^{m}(\beta_j)^2$$

El grado en que influye cada una de las penalizaciones está controlado por el hiperparámetro l1_ratio. Su valor está comprendido en el intervalo [0,1]. Cuando l1_ratio=0, se aplica ridge y cuando l1_ratio=1 se aplica lasso. La combinación de ambas penalizaciones suele dar lugar a buenos resultados. Una estrategia frecuentemente utilizada es asignarle casi todo el peso a la penalización l1 (l1_ratio muy próximo a 1) para conseguir seleccionar predictores y un poco a la l2 para dar cierta estabilidad en el caso de que algunos predictores estén correlacionados.

© Universidad Internacional de La Rioja (UNIR)

Tema 3. Algoritmos de aprendizaje automático supervisado

Conclusiones

Hasta este punto hemos visto los conceptos más importantes a la hora de evaluar, tanto los modelos de aprendizaje automático supervisados de regresión como los de clasificación y hemos visto un ejemplo de regresión lineal que a pesar de ser muy simple transmite la idea de cómo se aplican a problemas reales. Vimos, aunque de forma superficial, la idea detrás de los métodos para evaluar los modelos, sin embargo nos queda por ver los algoritmos de clasificación que veremos en la próxima lección.

Hemos tratado de escoger el material de las lecciones siempre pensando en la disponibilidad de documentación y la posibilidad de su rápida aplicación. No hemos querido irnos a detalles demasiado teóricos y listar la mayor cantidad de nombre de algoritmos y métodos que luego si es necesario se puedo buscar de forma fácil en Internet o literatura técnica.

© Universidad Internacional de La Rioja (UNIR)

Tema 3. Algoritmos de aprendizaje automático supervisado

3.3. Algoritmos de aprendizaje automático supervisado II

Objetivos

El objetivo de esta lección es continuar donde nos quedamos desde la anterior. En la anterior tocamos lo referente a conceptos como Bias y Varianza de un modelo, así como revisamos el concepto de validación cruzada. También vimos las métricas de evaluación de los algoritmos de regresión y clasificación y terminamos revisando algunos de los algoritmos de regresión. En esta lección revisaremos los algoritmos más conocidos de clasificación.

Cuando termine esta lección estará usted equipado con los conocimientos necesarios para poner el práctica soluciones que incluyan clasificación y evaluar sus resultados aplicando las métricas estudiadas en la lección pasada.

Introducción

En las lecciones anteriores hemos estudiado que dentro de las tareas de aprendizaje supervisado se encuentra la clasificación y que a diferencia de la regresión que trataba de predecir variables cuantitativas, la clasificación trata de métodos para predecir variables cualitativas.

En esencia de lo que tratan los métodos de aprendizaje automático para clasificación es de aprender una regla que permita decidir cuándo un caso pertenece a una clase o no a partir de los rasgos. Por ejemplo, decidir si un caso pertenece a un paciente diabético o no en función de rasgos o variables como la glusemia, la edad, o la presión. Para ello, la regla se deriva, por algún método a partir de un conjunto de casos de ejemplo, en este caso concreto puede ser a partir del Pima Indian Diebetes dataset u otro conjunto de casos.

Las tareas de clasificación pueden dividirse según el número de clases en las que pueden dividirse los casos en:

- **Clasificación binaria (Dos clases):** La variable Outcome solo toma dos valores.
- **Clasificación Multiclase:** La variable Outcome toma varios valores.

De manera general el problema de la clasificación podemos definirlo de la siguiente forma:

Dado un conjunto de casos y un conjunto de etiquetas, encontrar una función que aprenda el mapeo entre casos y etiquetas, y que luego sea capaz de asignar la etiqueta correcta a casos no vistos en el conjunto inicial.

© Universidad Internacional de La Rioja (UNIR)

Aunque esta es una definición informal del problema de clasificación, nos sirve perfectamente para comenzar la introducción de los principales métodos o algoritmos de aprendizaje supervisado y poner en práctica este tipo de tarea.

La calidad de la clasificación de un modelo de aprendizaje automático depende en gran medida de la separabilidad entre las clases. Una clase es un grupo de casos que comparten características comunes y que por tanto en el dataset con el que pretendemos entrenar, deben tener una etiqueta común. Sin embargo la diferencia entre las propiedades, en algunos casos, puede que no sean lo suficientemente diferenciables.

Eso nos lleva a examinar el concepto de separabilidad entre clases. Cuando la distancia entre las clases, hablando en términos de un espacio de rasgos, es suficiente, la clasificación no resulta problemática; pero cuando esta distancia no es suficiente y los casos que pertenecen a una clase u otra están muy cercas entonces ya no es tan sencilla la clasificación.

Si suponemos un espacio de rasgos, de dos dimensiones y se puede encontrar una linea recta, de forma tal que los casos de las dos clases se ubiquen a cada lado de esta recta se dice que las clases son linealmente separables.

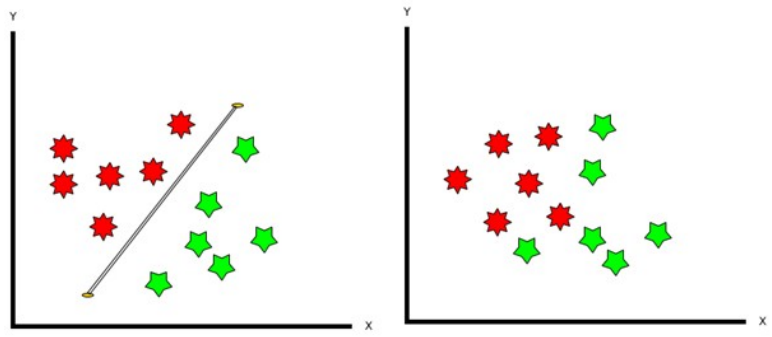


Figura 10. Figura de la Izquierda: dos clases linealmente separables. Figura de la derecha: Clases que no son linealmente separables.

A los clasificadores que solo pueden lidiar con clases linealmente separables se les denominan clasificadores lineales, mientras a los que tratan con clases que no lo son, se les llaman clasificadores no lineales.

© Universidad Internacional de La Rioja (UNIR)

Tema 3. Algoritmos de aprendizaje automático supervisado

Estrategias de clasificación

Como mencionamos al inicio, el problema de la clasificación puede dividirse en binaria o multiclase. Esto hace que se hayan diseñado clasificadores que son capaces de enfrentar ambos tipos, sin embargo otros solo pueden hacer clasificación binaria.

Para poder usar estos clasificadores binarios por excelencia en clasificación multiclase se han desarrollado un par de estrategias: Uno contra Uno y Uno contra Todos.

Uno contra Todos. One-versus-all (OvA o OvR):

Supongamos que necesitamos clasificar los casos del Iris dataset en las tres categorías que ya conocemos: setosa, versicolor y virginica. En esta estrategia de uno-contra-todos crearíamos tres clasificadores: uno que distinguiese entre setosa y las otras dos. Otro que distinguiese entre versicolor y las otras, y un tercero que distinguiese entre virginica y el resto. A la hora de clasificar un caso, se ejecutarían los tres clasificadores y se escogería la especie que recibiese el mayor voto. En general, si tenemos N clases distintas, harán falta N clasificadores.

Uno contra Uno. One-vs-One (OvO):

En esta segunda estrategia entrenaríamos un clasificador para cada combinación posible de especie de Iris: uno para setosa-versicolor, otro para setosa-virginica y otro para versicolor-virginica. A la hora de clasificar una caso, se ejecutarían todos los clasificadores para ver qué clase "gana" la mayor parte de las confrontaciones (si el caso pertenece a la clase setosa, uno esperaría que fuese así detectada por el clasificador setosa-versicolor y por el setosa-virginica).

En esta estrategia, si tenemos N clases, harán falta N x (N-1) / 2 clasificadores.

El problema de la clasificación ha sido bien estudiado en el marco de la estadística y del aprendizaje automático, se han desarrollado varios enfoques y cientos de algoritmos. Un par de libros serían insuficientes. En esta lección cubriremos algunos de los enfoques principales y trataremos de mantener la matemática al mínimo haciendo énfasis en la aplicabilidad.

En esta lección usaremos el dataset Pima Indian Diabetes, que de alguna manera lo hemos usado en las primeras lecciones. Este dataset pertenece a un problema de clasificación binaria. Sin embargo para demostrar el problema de clasificación multiclase, usaremos el dataset iris, que contiene 150 casos, pertenecientes a tres clases diferentes.

© Universidad Internacional de La Rioja (UNIR)

Tema 3. Algoritmos de aprendizaje automático supervisado

El dataset Pima Indian diabetes, consta con 9 variables, 8 variables son independientes y uno la variable de clase o outcome que toma solo dos valores : 1 para los casos con diabetes confirmada y 0 para aquellos que no.

Algoritmos de clasificación

K-Nearest Neighbors (KNN):

Quizás el algoritmo de clasificación más simple de aplicar y de entender es el llamado KNN. Estas siglas significan K Nearest Neighbors o los K vecinos más cercanos y comenzaremos esta lección explicando cómo funciona y la manera de aplicarlo a un dataset.

La idea detrás de este algoritmo es muy sencilla pero potente y consiste en encontrar los k casos que más se parecen al nuevo caso que se quiere clasificar y asignarle la clase que más se repite o más frecuente entre todos los vecinos. Esta clase se elige por voto simple. Este número k puede ser definido por el usuario.

Como todos los algoritmos de aprendizaje automático tiene sus pro y sus contras, vemos:

Pros

- Es muy fácil de usar.
- No hace ninguna asunción sobre los datos. (No paramétrico)
- Muy rápido.

Contras

- Es necesario encontrar el k óptimo.
- Como no genera un modelo interno de los datos, los memoriza y luego en cada paso de clasificación los usa todos para tomar una decisión.

Veamos cómo funciona:

En la fase de entrenamiento, toma todos los datos de dataset y los almacena en una estructura interna, donde le permita, luego, hacer las comparaciones. En las implementaciones de scikit-learn usan principalmente dos, KD-Tree y Ball-Tree. Este proceso es relativamente rápido.

Luego, en la fase de test o clasificación, cuando se le presenta un caso a clasificar, busca, usando determinada estrategia, ya sea fuerza bruta u otra, los k casos más cercanos al que se desea clasificar.

© Universidad Internacional de La Rioja (UNIR)

Para determinar esta cercanía se usa una medida de distancia.

Una de las distancias que se usa muy frecuentemente con este algoritmo es la llamada distancia Euclideana.

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Sin embargo la distancia por omisión de la implementación en scikit-learn es la Minkowski.

$$d(p,q) = \left(\sum_{i=1}^{n} |p_i - q_i{}^p| \right)^{\frac{1}{p}}$$

Aunque se pueden usar otras métricas de distancias.

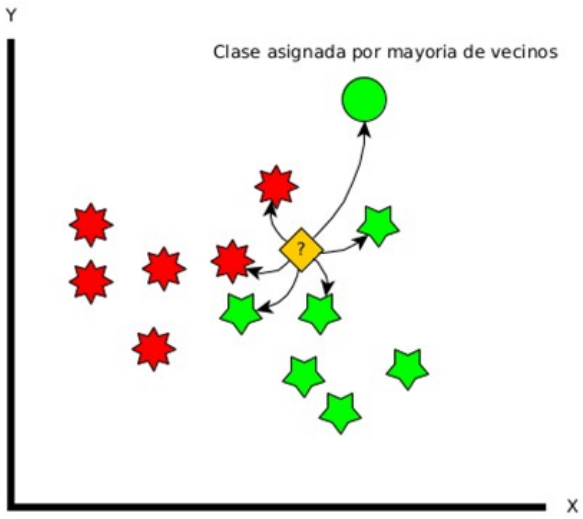
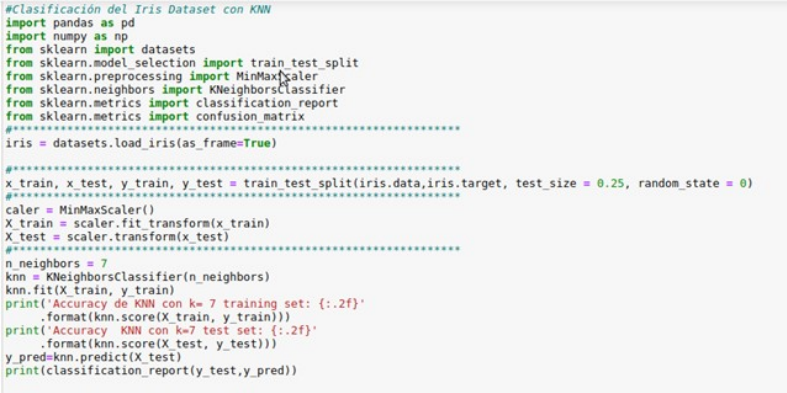


Figura 12. Clase asignada a un caso, por mayoría de votos. K=5.

El valor óptimo de K se puede ajustar por medio de validación cruzada.

En la imagen siguiente se muestra un ejemplo de KNN en Python usado para hacer predicciones sobre el iris Dataset.

© Universidad Internacional de La Rioja (UNIR)



```python
#Clasificación del Iris Dataset con KNN
import pandas as pd
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
#*****************************************************************
iris = datasets.load_iris(as_frame=True)

#*****************************************************************
x_train, x_test, y_train, y_test = train_test_split(iris.data,iris.target, test_size = 0.25, random_state = 0)
#*****************************************************************
caler = MinMaxScaler()
X_train = scaler.fit_transform(x_train)
X_test = scaler.transform(x_test)
#*****************************************************************
n_neighbors = 7
knn = KNeighborsClassifier(n_neighbors)
knn.fit(X_train, y_train)
print('Accuracy de KNN con k= 7 training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy  KNN con k=7 test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
y_pred=knn.predict(X_test)
print(classification_report(y_test,y_pred))
```

Figura 13. Ejemplo de Implementación de KNN con Python.

En este ejemplo de cómo aplicar KNN a la clasificación de Iris, el código se divide en 5 partes. La primera donde se importa las librerías que contienen los métodos que necesitamos para la tarea a mano. Las partes se separan por líneas de asteriscos.

La segunda parte de este código es donde se importan el conjunto de datos. En la tercera etapa se separa el conjunto de datos en dos, uno con el fin de entrenar los métodos de clasificación y el otro con vistas a evaluar su calidad.

En la cuarta etapa, normalizamos los datos usando el MinMaxScaler.

Y en la quinta etapa entrenamos y evaluamos el modelo. En este caso de KNN hemos entrenado y probado el modelo con K=7.

Como en KNN la clase de un nuevo caso se asigna usando la mayoría de las clases de los K vecinos más cercanos, aconsejamos, siempre usar un K impar, para evitar que haya algún empate. También sugerimos que el valor optimo se encuentre usando Validación Cruzada.

El resultado de este ejemplo es el que sigue:

```
Accuracy de KNN con k= 7 training set: 0.97

Accuracy KNN con k=7 test set: 0.97

precision recall f1-score support



0 1.00 1.00 1.00 13
```


© Universidad Internacional de La Rioja (UNIR)

```
1 1.00 0.94 0.97 16

2 0.90 1.00 0.95 9


accuracy 0.97 38

macro avg 0.97 0.98 0.97 38

weighted avg 0.98 0.97 0.97 38
```

El valor por default de k, en la clase KneighborsClassifier es 5. Los parámetros que esta clase admite son los siguientes:

- n_neighbors: Es un valor de tipo entero. Es el K al que nos referimos, el número de vecinos cercanos y el valor por omisión es 5.
- weights: Este parámetro permite añadir pesos a los vecinos más cercanos por varias estrategias diferentes, el valor por omisión es 'uniform':
  - Uniforme (uniform): Todos los vecinos tienen la misma importancia o peso.
  - Basado en distancia (distance): A los vecinos de un nuevo caso a clasificar se les asigna un peso que disminuye con la distancia. Eso significa que de los k vecinos más cercanos, los más cercas reciben una importancia mayor que los más lejos.
  - Función definida por el usuario(callable): Por medio de este parámetro se le pude pasar una función la cual acepte un arreglo de distancias y retorne un arreglo de la misma forma con los pesos.
  - Algorithm: Este parámetro decide que algoritmo se usa para calcular los k vecinos más cercanos. El valor por omisión es Auto.
  - ball_tree: Usará el algoritmo Ball Tree para el cálculo de los vecinos más cercanos.
  - kd_tree: Usará el algoritmo de KD Tree para el cálculo de los k vecinos más cercanos.
  - brute: Usará Fuerza bruta como mecanismo de busqueda de los k vecinos.
  - auto: Tratará de encontrar el mejor método basado en los parámetros pasados a fit().
- Metric: Este parámetro define la métrica que usará el algoritmo para determinar los vecinos más cercanos. Las medidas que se pueden usar son:

© Universidad Internacional de La Rioja (UNIR)

identifier	class name	args	distance function
“euclidean”	EuclideanDistance	•	sqrt(sum((x - y)^2))
“manhattan”	ManhattanDistance	•	sum(|x - y|)
“chebyshev”	ChebyshevDistance	•	max(|x - y|)
“minkowski”	MinkowskiDistance	p	sum(|x - y|^p)^(1/p)
“wminkowski”	WMinkowskiDistance	p, w	sum(|w * (x - y)|^p)^(1/p)
“seuclidean”	SEuclideanDistance	V	sqrt(sum((x - y)^2 / V))
“mahalanobis”	MahalanobisDistance	V or VI	sqrt((x - y)' V^-1 (x - y))

El resto de los argumentos son dependientes de las métricas y de los argumentos. Para el resto de los detalles, por favor, refiérase a la documentación de Scikit-learn.

Máquinas con vectores de apoyo (SVM)

Este es uno de los algoritmos más robustos de aprendizaje automático. Se puede usar tanto para regresión como para clasificación, aunque principalmente se usa en clasificación.

Este algoritmo fue desarrollado por Vladimir Vapnik y algunos de sus colegas basados en la teoría del aprendizaje estadístico o teoría de Vapnik-Chervonenkis. (VC Theory).

La gran contribución de Vapnik al campo del aprendizaje automático ha consistido, precisamente, en proponer uno de los primeros modelos de estimación/predicción que no está basado en ningún modelo probabilístico, lo cual es un gran cambio de mentalidad. Puesto que todos los modelos que se usaban hasta ese momento estaban basados en la teoría estadística y de probabilidad.

Sin embargo, las máquinas de vector de soporte (SVM) se pueden considerar el primer modelo que utiliza un enfoque completamente distinto y que está basado en un modelado geométrico y que se resuelve mediante un problema de optimización con restricciones.

Una de las principales ideas detrás de este método es la de tratar encontrar un hiperplano que pueda separar las clases y que además lo haga con el mayor margen posible entre los vectores o casos que se encuentran en los bordes de las clases. A estos casos se les llaman Vectores de apoyo o de soporte. En la imagen siguiente de encuentran destacados con el centro en color amarillo y en sobre la línea discontinua.

© Universidad Internacional de La Rioja (UNIR)