UNIVERSITAT DE BARCELONA

Aplicación web para comparar algoritmos de Machine Learning Supervisado

Proyecto final de curso

DATA SCIENCE (CIENCIA DE LOS DATOS): APLICACIONES A LA BIOLOGÍA Y A LA MEDICINA CON PYTHON Y R

Javier Jarque Valentín

Xavier Abarca Garcia

30 de julio del 2018

Índice

Introducción

"*La ciencia de datos es un campo interdisciplinario que involucra métodos científicos, procesos y sistemas para extraer conocimiento o un mejor entendimiento de datos en sus diferentes formas, ya sea estructurados o no estructurados, lo cual es una continuación de algunos campos de análisis de datos como la estadística, la minería de datos, el aprendizaje automático y la analítica predictiva.*"

Así es como wikipedia define la ciencia de datos. El mundo que nos rodea es un mundo cada vez más digitalizado, en el que se generan muchísimos datos. Todo está parametrizado, sensorizado... y el gran volumen de datos de los que se dispone puede suponer información muy valiosa para el quien sepa extraer conclusiones de su estudio. La ciencia de datos precisamente provee de una serie de tecnologías y metodologías para convertir todo ese caudal de información no estructurada en información útil, entendible y estructurada, básica para un montón de campos diferentes: biología, medicina, negocios online, domótica, seguridad y un sinfín de posibles aplicaciones, muchas de ellas aún por descubrir.

La ciencia de datos surge gracias a la necesidad de estudio de la gran cantidad de datos existentes, y apoyada en la potencia de cáculo de los equipos informáticos actuales permite implementar muchas técnicas estadístico-matemáticas conocidas desde hace tiempo (imposibles de llevar a cabo en la práctica por la limitada potencia de cálculo existente). Y a raíz de esta necesidad creciente, se han desarrollado y perfeccionado muchas otras. La estadística, las matemáticas y la informática son disciplinas tecno-científicas que proporcionan transversalmente apoyo a muchas otras disciplinas, pero en la ciencia de datos son la base fundamental sobre el que se construye su base.

El curso universitario "*DATA SCIENCE (CIENCIA DE LOS DATOS): APLICACIONES A LA BIOLOGÍA Y A LA MEDICINA CON PYTHON Y R*" es una de tantas iniciativas para formar a profesionales con experiencia en distintas disciplinas en este apasionante y nuevo mundo.

Objetivos

El objetivo de este trabajo es poner en práctica todo lo aprendido en el curso, y no es tarea fácil en tanto en cuanto la ciencia de datos (*data science*) se nutre de campos tan diversos y amplios como la Informática, la Estadística, las Matemáticas o la Biología. El curso ha tratado de reflejar esta gran diversidad de herramientas disponibles y el abanico de posibilidades era grande a la hora de afrontar este trabajo. Pero hemos creído oportuno centrarnos en las dos herramientas básicas que un científico de datos debe al menos conocer, si no ya dominar: *R* y *python*. Y con estas herramientas, adentrarnos en uno de los campos que más nos interesan: el *machine learning*.

En concreto hemos desarrollado una aplicación web que evalua diferentes modelos de *machine learning supervisado* dado un conjunto de datos cualquiera (dataset) y permite establecer cual de esos modelos seleccionados (de entre un total de 10 posibles) obtiene mejores resultados. La aplicación está hecha en R (usando su módulo de publicación web *shiny*) y usa los clasificadores de machine learning (ML) supervisado de la librería *scikit-learn* de python.

Nuestra aplicación pretende servir de ayuda a quien necesite de un desarrollo de machine learning supervisado, ya que permite fácilmente observar los datos, la distribución de clases a clasificar (cuantos tipos hay y en qué proporción aparecen) y qué algoritmos de ML dan a priori (en su configuración por defecto) una mayor efectividad (accuracy) de predicción.

Habrá en el mercado aplicaciones, proyectos colaborativos y aplicaciones web gratuitas

que hagan algo parecido. Pero el marco de un proyecto fin de curso universitario en el que estamos, nos permite alcanzar diferentes objetivos:

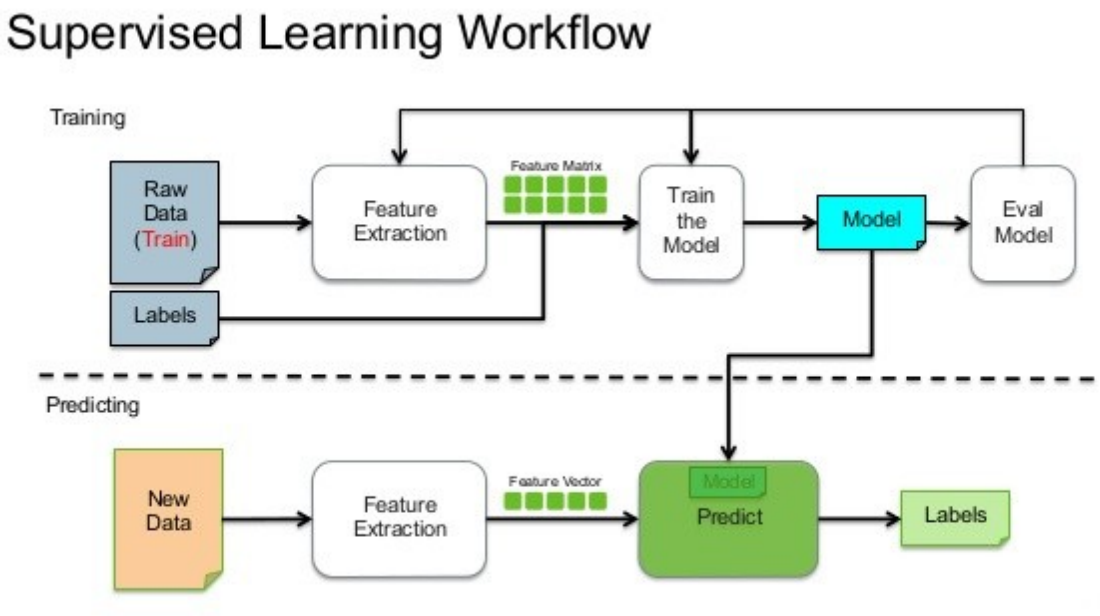
- Permite explorar los entresijos de varias herramientas data science básicas y muy interesantes, consolidando los puntos aprendidos durante el curso (R, python, shiny o scikit-learn) que al final es el objetivo más importante

- Nuestra aplicación trabaja con cualquier tipo de dataset, siempre que tenga una columna final con la clase a clasificar. Este es un punto fuerte, pero a su vez hay que comentar que quizá es una flaqueza también, ya que al generalizar tanto hay análisis que podríamos haber hecho para un dataset concreto que es difícil montar para uno cualquiera en general. Un análisis de datos necesita entrar en profundidad en lo que los datos representan, pero hemos decidido seguir este camino.

- La aplicación web resultante está desarrollada con herramientas totalmente gratuitas y libres. El código no sólo es público sino que está subido en un servidor accesible a todo el mundo y listo para ser descargado, modificado y mejorado por quien quiera hacerlo.

Metodología

El propósito de nuestra aplicación web es la de comparar diferentes algoritmos de Machine Learning supervisado. La aplicación es capaz de leer un dataset, conjunto de datos en formato CSV con campos separados por punto y coma ";". El dataset ha de estar compuesto por una serie de campos (o características, *features*, variables) y un campo final con la clase (o *target*, *label*).



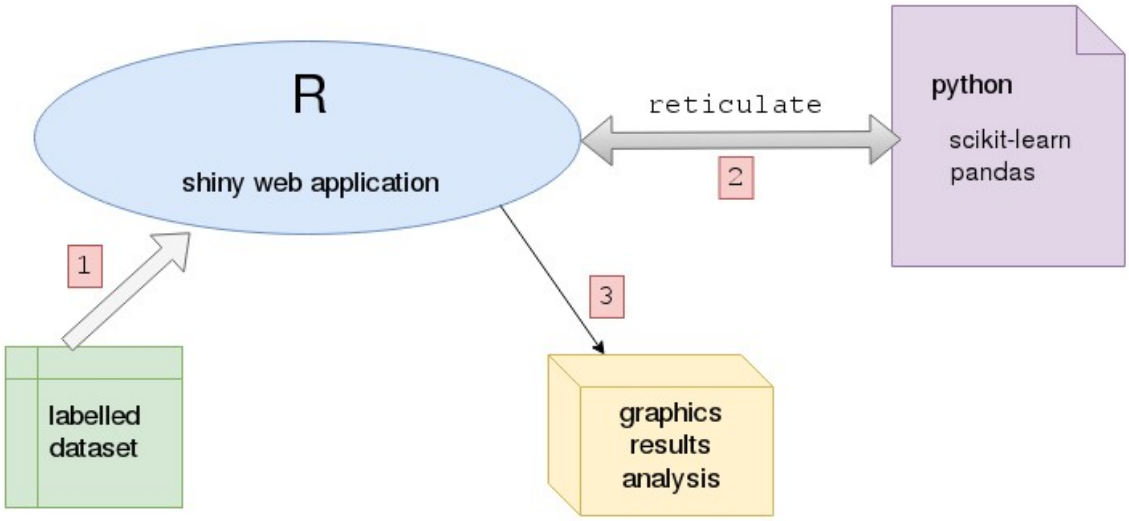
El Machine Learning Supervisado es la técnica que permite entrenar un algoritmo para que sea capaz de determinar si una observación concreta es de una clase u otra, en base a un aprendizaje previo. Es decir, si subministramos suficientes datos (individuos o observaciones conformados por una serie de variables) y estas están etiquetadas tienen marcadas previamente a qué clase pertenecen, el algoritmo puede se capaz de clasificar futuras observaciones, con una fiabilidad suficiente (en algunos casos superior al 90%).

En nuestra aplicación se permite al usuario proporcionar un dataset al programa, para que luego éste proporcione la siguiente información:

- histograma de cada una de las variables del dataset

- barplot para considerar la distribución de las clases o labels
- estadísticas de cada variable (media, diaviación estándard, mediana, cuartiles…)
- posibilidad de escoger uno o varios modelos o algoritmos de Machine Learning para ver cómo se comportan en cuanto a rendimiento (tanto en modo gráfico como visualitzando los datos numéricos obtenidos)

Para su realización se han utilizado una serie de herramientas informáticas:



shiny / R

Para la realización de la aplicación en sí, con su navegación, los diferentes gráficos dinámicos que se incluyen, la presentación de resultados, la interfaz para que se puedan seleccionar qué algoritmos se usarán… se ha usado Shiny, que es un paquete que proporciona RStudio para realizar desarollos online sencillos pero bastante aparentes. Al

ser un paquete de R (correr sobre R) permite usar toda la potencia de R para graficar usando la potente librería ggplot2.

python / scikit-learn

Se ha creído oportuno, por cuestiones didácticas, y también por rendimiento que la realización de las tareas de computación se lleven a cabo en python. Python dispone de muchas librerías para el procesado de datos y para entrenamiento y testeo de modelos/algoritmos de machine learning. Pandas y numpy son las elegidas para lo primero (data-wrangling y procesado general) mientras que la potente y completísima skicit-learn es la elegida para machine learning.

reticulate

La conexión entre R y python se realiza mediante un package de R llamado reticulate. Permite que código python se ejecute en R, pudiendo añadir toda la potencia de las librerías de machine learning y procesados de data science (pandas, scipy, numpy) en R. Ha sido clave para la realización de este proyecto el encontrarla y hacerla funcionar. En el *Anexo de instalación* se comentan posibles problemas que pueden surgir y cómo solucionarlos.

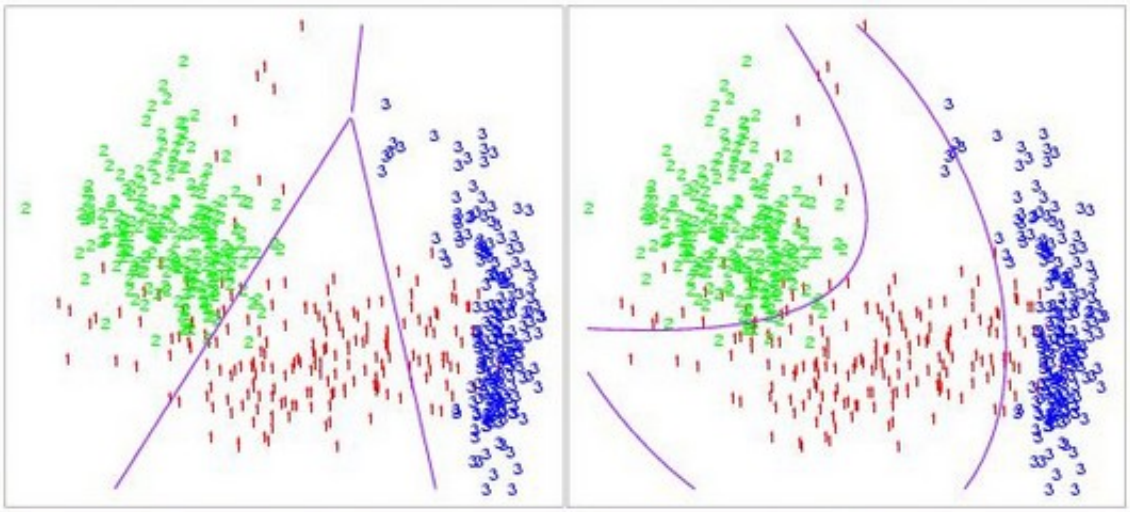
A continuación se realiza una breve descripción de los algoritmos disponibles para ser probados en la aplicación:

Linear discriminant Analysis (LDA) y Quadratic discriminant analysis (QDA)

El Linear Discriminant Analysis es un método de regresión lineal que incorpora una separación en clases. Este método consiste en agrupar los conjuntos de parátros de entrada que determinan una variable de salida parecidos entre sí, creando clases (estos grupos de parámetros similares). Para cada clase, la variable de salida se aproxima usando una regresión lineal:

$$y = \sum_i \alpha_i x_i + \alpha_0$$

Dado un nuevo conjunto de parámetros de entrada, la variable de salida se aproxima en dos pasos. Primero se determina la clase de la cual forma parte y después se usa la recta de regresión específica para aquel conjunto para determinar la variable de salida.



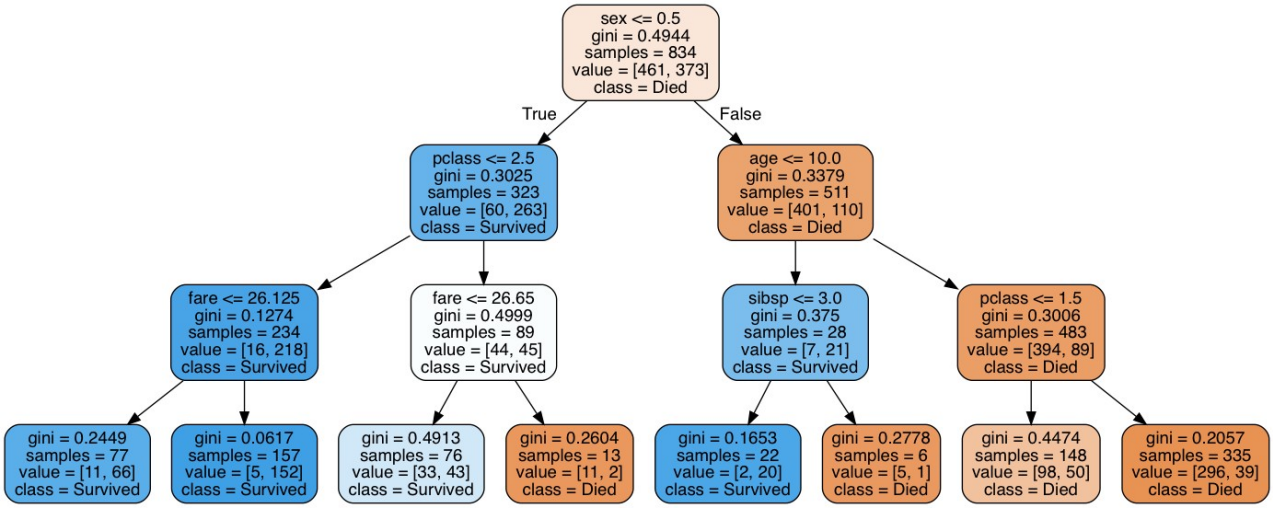
El método cuadrático (Quadratic Discriminant Analysis) discrimina no linealmente sino

mediante una superfície de decisión cuadrática.

Decision tree

Un árbol de decisión (decition tree) és una grafo que indica las acciones a realizar en función del valor de una o varias variables. Es una representación en forma de árbol cuyas ramas se bifurcan en función de los valores tomados por las variables y que terminan en una acción concreta.

En Machine Learning, lo que se hace es separar binariamente los resultados en función de una variable y evaluar cuan buena ha sido dicha separación, en función de un coste (definido matemáticamente). Esta evaluación nos pemite realizar diferentes bifurcaciones binarias recursivamente y tratar de quedarnos con las que más separen nuestros datos.



Es una algoritmo que permite gráficamente de forma muy intuitiva cómo se ha llegado a la conclusión de clasificar de una determinada forma, y funciona bien con pocas variables y muchos datos, para no hacer inmeso dicho árbol.

Random Forest

Este algoritmo es básicamente un Decition Tree mejorado, en el que se generan diferentes árboles de decisión y se hace un promedio de los resultados.

La ventaja del Random Forest es que agrega aleatoriedad adicional al modelo, mientras

crecen los árboles. En lugar de buscar la característica más importante mientras se divide un nodo, busca la mejor característica entre un subconjunto aleatorio de características. Esto da como resultado una amplia diversidad que generalmente da como resultado un mejor modelo.

Por lo tanto, en Random Forest, solo un subconjunto aleatorio de las características es tomado en consideración por el algoritmo para dividir un nodo. Incluso puede hacer que los árboles sean más aleatorios, al usar adicionalmente umbrales aleatorios para cada característica en lugar de buscar los mejores umbrales posibles (como lo hace un árbol de decisión normal).

Extra Trees

El método Extras Trees (que representa árboles extremadamente aleatorios) nació con el objetivo principal de aleatorizar aún más la construcción de árboles en el contexto de las funciones de entrada numérica, donde la elección del punto de corte óptimo es responsable en gran proporción de la varianza del árbol resultante.

Con respecto a los Random Forest, el método descarta la idea de utilizar copias de arranque de la muestra de aprendizaje, y en lugar de tratar de encontrar un punto de corte óptimo para cada una de las K variables elegidas al azar en cada nodo, selecciona un punto de corte al azar. Esta idea es bastante productiva en el contexto de muchos problemas caracterizados por un gran número de características numéricas que varían más o menos continuamente: a menudo conduce a una mayor precisión gracias a su suavizado y, al mismo tiempo, reduce significativamente las cargas computacionales vinculadas a la determinación de óptimos puntos de corte en árboles estándar y en bosques aleatorios.

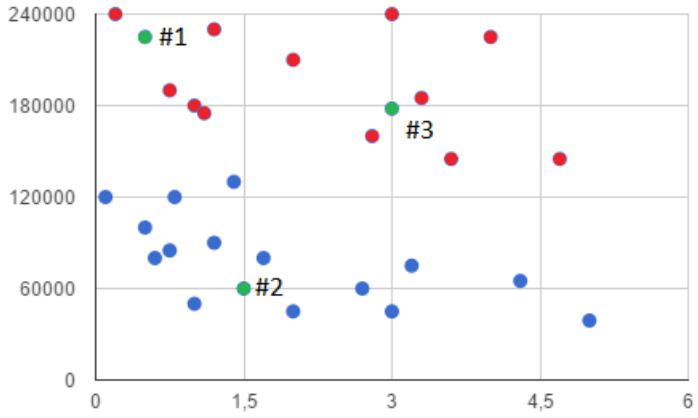
Desde un punto de vista estadístico, descartar la idea de arranque lleva a una ventaja en términos de sesgo, mientras que la aleatorización de punto de corte a menudo tiene un excelente efecto de reducción de varianza.

K-Neighbors

El métode de clasificación K-Neighbors (o K-Nearest Neighbors) es un algoritmo que intenta agrupar las individuos que tienen unas características semejantes en el espacio de parámetros. A diferencia del clasificador por árboles, la decisión se torma en base a criterios de proximidad en base a la definición de una función de distancia. El algoritmo intenta clasificar agrupando mediante una minimización de las distancias entre los diferentes individuos entre sí.

Tiene un parámetro principal (la k), que define en cuantas clases se quiere clasificar tus datos. El algoritmo define unos centoides (puntos centrales de cada uno de los k-conjuntos) y lo que se minimiza es la distancia de cada elemento a ese centoide para situarlos en el espacio de parámetros.

Es uno de los algoritmos más sencillos e intuitivos, ya que el concepto de distancia (euclídea o no, la implementación de los alforitmos permiten decidir entre diferentes pre-establecidas) es visible al graficar en 2-D.



Podemos comentar finalmente algunas de sus características:

- El procesamiento de la muestra de prueba consume mucho tiempo, ya que para cada muestra de prueba, el modelo debe recorrer todo el conjunto de datos para encontrar los vecinos más cercanos.

- Los datos de entrenamiento deben ser de naturaleza numérica, ya que la clasificación es conducida por distancias euclidianas.
- El clasificador kNN no funciona muy bien en datos desequilibrados. P.ej. considere dos clases, A y B. Si la mayoría de los datos de entrenamiento, digamos 70-80% de ella se clasifica como A, entonces el modelo finalmente dará mucha preferencia a A, sin importar qué tan pequeño sea k. Esto puede ocasionar que muchos datos sean clasificados erróneamente.

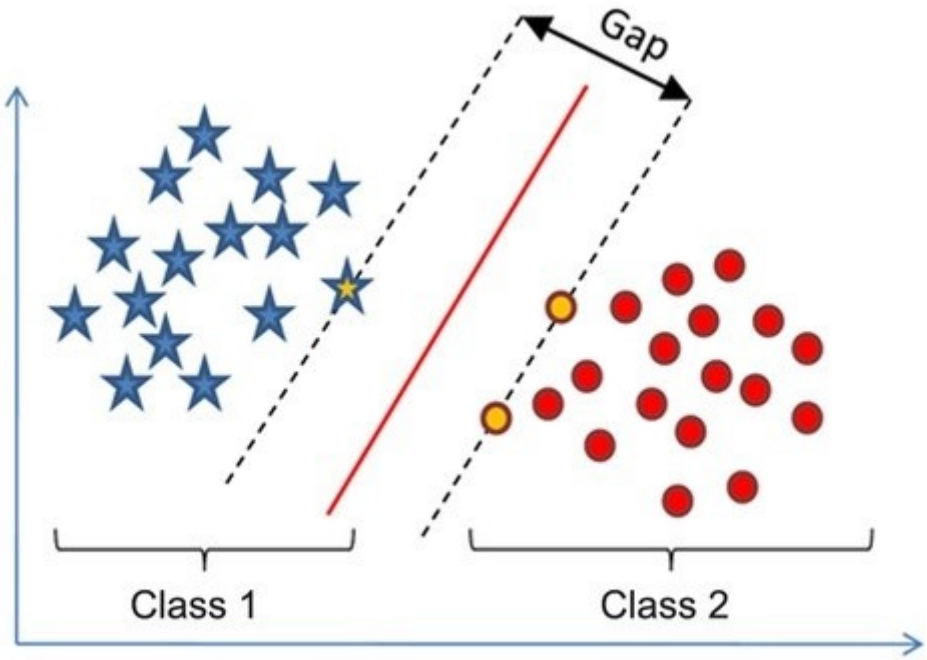
Gaussian Naive Bayes

Es un predictor basado en el teorema de Bayes y en la hipótesis de que los parámetros predictores son independientes entre sí. Esta hipótesis define su nombre (naive=ingenuo).

Lo que se hace es separar primero los conjuntos de parámetros predictores en clases diferentes. Para cada clase suponemos que los diferentes parámetros de entrada siguen una distribución gausiana y calculamos su media y desviación estándard. Conociendo estos dos datos se puede determinar en cuál de las clases colocaríamos cada nuevo individuo.

Suport Vector Machine (SVM)

Dado un conjunto de ejemplos de entrenamiento (de muestras) podemos etiquetar las clases y entrenar una SVM para construir un modelo que prediga la clase de una nueva muestra. Intuitivamente, una SVM es un modelo que representa a los puntos de muestra en el espacio, separando las clases a 2 espacios lo más amplios posibles mediante un hiperplano de separación definido como el vector entre los 2 puntos, de las 2 clases, más cercanos al que se llama vector soporte. Cuando las nuevas muestras se ponen en correspondencia con dicho modelo, en función de los espacios a los que pertenezcan, pueden ser clasificadas a una o la otra clase.



Más formalmente, una SVM construye un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta.

Bootstrap y bagging (Adaboost y RF-Bagging)

Bootstrap y Baggins son dos técnicas llamadas de *Ensemble* (agrupación). Este es un concepto de Machine Learning en el cual la idea es entrenar múltiples modelos usando el mismo algoritmo de aprendizaje. Los conjuntos participan en un grupo más grande de métodos, llamados multiclasificadores, donde un conjunto de cientos o miles de entrenadores con un objetivo común se fusionan para resolver el problema.

Las principales causas de error en el aprendizaje se deben al ruido, el sesgo y la varianza. Las técnicas de Ensemble ayuda a minimizar estos factores. Estos métodos están diseñados para mejorar la estabilidad y la precisión de los algoritmos de Machine Learning. Las combinaciones de clasificadores múltiples disminuyen la varianza, especialmente en el caso de los clasificadores inestables, y pueden producir una clasificación más confiable que un clasificador único.

Bootstrap y Bagging genera N entrenadores mediante la generación de datos adicionales en la etapa de aprendizaje. Los N nuevos conjuntos de datos de entrenamiento se producen mediante muestreo aleatorio con reemplazo del conjunto original. Al muestrear con reemplazo, algunas observaciones pueden repetirse en cada nuevo conjunto de datos de entrenamiento.

En el caso del Bagging, cualquier elemento tiene la misma probabilidad de aparecer en un nuevo conjunto de datos. Sin embargo, para Boosting las observaciones son ponderadas y, por lo tanto, algunas de ellas tomarán parte en los nuevos conjuntos con mayor frecuencia.

Parece que estos métodos hayan de ser siemre los mejores, pero dependerán de qué modelo o algoritmo haya por debajo (por defecto Random Forests) ya que Bootstrap/Bagging son técnicas de entreno más que modelos en sí.. Cuál de los dos produce mejores resultados? Dependerá de los datos, la simulación y las circunstancias. Veamos:

- Ambos reducen la varianza de su estimación individual ya que combinan varias estimaciones de diferentes modelos. Entonces, el resultado puede ser un modelo con mayor estabilidad.
- Si el problema es que el modelo individual obtiene un rendimiento muy bajo, Bagging rara vez tendrá un mejor sesgo. Sin embargo, Boosting podría generar un modelo combinado con menos errores, ya que optimiza las ventajas y reduce las trampas del modelo único.
- Por el contrario, si la dificultad del modelo individual es el over-fitting, el Bagging es la mejor opción. El impulso por su parte no ayuda a evitar el exceso de ajuste; de hecho, esta técnica se enfrenta con este problema en sí mismo. Por esta razón, el embolsado es efectivo más a menudo que el Bootstrap.

Conclusiones

Estamos satisfechos con el resultado obtenido. La aplicación resultante cumple con su cometido, está realizado con herramientas libres (gratuítas y de código abierto) y está disponible online para su descarga, instalación e incluso modificación si alguien se atreve con ello.

Consideramos que todo proyecto o aplicación puede mejorar y nuestro caso no es una excepción. Debido al tiempo limitado y al entorno educacional en el que ha sido desarrollado el mismo hay puntos que hubiésemos añadido/ampliado y no ha sido posible. Algunos de ellos son:

- La preparación de los datos para iniciar un análisis determinado, depende mucho del conjunto de datos usado. Pero hay técnicas de data-wrangling que podrían haberse usado y no se ha hecho (por tiempo, básicamente).
- Creemos que no sería complicado introducir una opción para hacer un estudio de Principal Component Analysis y hacer un gráfico 2D con la posible separación de las clases en función de las dos variables con más peso en el nuevo espacio PCA.
- Se han probado diversos modelos de machine learning supervisado, pero se han probado con los hiperparámetros por defecto. Podría haberse preparado una opción para que el usuario pudiese modificar dichos parámetros en tiempo real y ver cómo evolucionaba con ellos la predición de las clasificaciones.
- En datasets con muchas variables, una muy buena idea sería hacer feature selection. Se podría haber implementado una opción para ello, de forma más o menos sencilla dadas las herramientas (R & python) con las que contamos.

Aún sí, estamos satisfechos con el resultado y creemos que puede ser útil, siempre teniendo en cuenta en al ámbito en el que se ha desarrollado el proyecto y el alcance a priori del mismo.

Bibliografía

[1] International Society for Computational Biology, “START: Shiny Transcriptome Analysis Resource Tool Getting Started Input Data Group Plots Analysis Plots” - https://kcvi.shinyapps.io/START/

[2] Martí Català Sabaté, “Machine Learning per a l’optimització d’un model epidemiològic en tuberculosi”, Màster en Bioinformàtica i Bioestadística 2018, UOC

[3] Scikit-learn - Documentación oficial - http://scikit-learn.org/stable/index.html

[4] Pandas – Documentación oficial - https://pandas.pydata.org/pandas-docs/stable/

[5] Reticulate - Página oficial - https://github.com/rstudio/reticulate

[6] Igual, Laura, Seguí, Santi, “Introduction DataScience with python (curs UB)” - https://github.com/DataScienceUB/introduction-datascience-python-book

[7] Jeff Delaney, “10 Classifier Showdown in Scikit-Learn” - https://www.kaggle.com/jeffd23/10-classifier-showdown-in-scikit-learn

Anexos

Instalación entorno: instrucciones

python: miniconda + paquetes: numpy – scipy - pandas – scikit-learn – matplotlib – seaborn
/home/datascience/miniconda3/bin/conda install numpy scipy pandas scikit-learn matplotlib seaborn

R y R-Studio: paquetes extra: ggplot2, reticulate, ggcorrplot
install.packages("ggplot2")
install.packages("reticulate")
install.packages("ggcorrplot")

github proyecto

abrir Rstudio

cargar proyecto

Run !