Universidad Europea de Madrid

Escuela de Arquitectura, Ingeniería y Diseño

PROYECTO GRUPAL

OPEN DATA II

Profesor: RAFAEL MUÑOZ GIL

Integrantes:

MARÍA FERNÁNDEZ MORÍN
CRISTINA AMAIA RODRÍGUEZ VIRTO

ÍNDICE

1. EXPLICACIÓN DEL DATASET:

El dataset que vamos a estudiar contiene información sobre 303 distintos pacientes pertenecientes a Cleveland Clinic Foundation. La fecha de la tabla data de julio de 1988. De dichos pacientes se muestran 13 distintas características llegando hasta la columna objetivo de la base de datos: el paciente desarrolla o no una enfermedad de corazón. En total conforman 14 columnas.

El dataset nos proporciona la siguiente información:

1. Edad: edad en años
2. Género: género del paciente
3. MT: tipo de molestia torácica
4. PArterial: presión arterial en reposo en mm/Hg
5. Colesterol: colesterol en suero en mg/dl
6. Glucemia: azúcar en sangre del paciente en ayunas
7. EcgReposo: electrocardiograma en reposo
8. FCM: frecuencia cardíaca máxima
9. AIE: angina inducida por el ejercicio
10. STDep: depresión del ST inducida por el ejercicio en relación al descanso
11. Pendiente: la pendiente del ejercicio máximo del segmento ST
12. Vasos: número de vasos principales
13. TipoDefecto: tipo de anomalía que presenta el paciente
14. Enfermedad: indica si el paciente desarrolla o no una patología de corazón

Encontramos ciertos valores de diversos atributos que están indicados en números y no se deduce su significado sin una información que procedemos a explicar a continuación:

- Género - 1: masculino; 2:femenino
- MT - 0: asintomático; 1: angina atípica; 2: ninguna angina detectada; 3: angina común
- Glucemia - 1: mayor a 120 mg/dl; 0: menor a esa cifra
- EcgReposo - 0: que muestra hipertrofia en el ventrículo izquierdo; 1: normal; 2: anormalidad de la onda T
- AIE - 1: sí; 0: no
- Pendiente - 0: en bajada; 1: plana; 2: subida
- TipoDefecto - 0: no claro; 1: anomalía solucionada; 2: normal; 3: defecto reversible

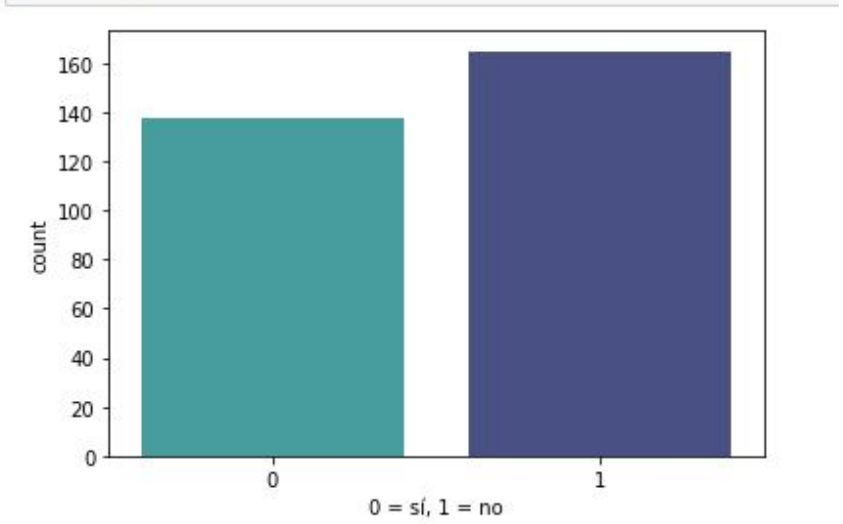
2. BALANCEADO:

Nuestra columna objetivo es enfermedad. Al analizar si los datos están balanceados vemos si la distribución de los datos, en nuestro caso si el paciente está enfermo o no, es homogénea y tenemos proporciones parecidas para los posibles resultados.
En problemas de clasificación como el nuestro puede ocurrir que uno de los grupos de datos sea minoritario, es decir, que se encuentre en menor medida respecto al grupo complementario. Esto provoca un desbalance de los datos que nos sirven para el uso de los algoritmos y de su correspondiente entrenamiento. Este desbarajuste puede ser causante de overfitting y poca

precisión en los resultados, el grupo minoritario no se vería correctamente representado con el modelo obtenido.

Para nuestros datos, vemos que está balanceado ya que tenemos una proporción 45,54% el paciente si está enfermo y 54,46% el paciente no está enfermo.

Enfermedad: 0= sí (45,54%) y 1= no (54,46%)



3. SQL:

SQL Spark es un módulo Spark que sirve para procesar datos estructurados y resulta de utilidad como motor SQL distribuido. Es importante recalcar que cuando se habla de datos estructurados se refiere a la información que suele encontrarse en la mayoría de bases de datos, información etiquetada y controlada que puede encontrarse en filas o en columnas.
Con SQL Spark, podemos consultar datos en bases y otros conjuntos de datos.
El uso de este módulo consiste en ejecutar consultas SQL
Su uso tiene las siguientes ventajas: (documento)

- Unificar información distribuida en diferentes entornos o sistemas de almacenamiento.
- Explorar y analizar datos con un alto nivel de abstracción mediante lenguaje SQL, permitiendo su uso a personas no especializadas.
- Elimina restricciones impuestas por las fuentes de datos de origen. Por ejemplo, Cassandra no permite realizar querys filtrando por campos no incluidos en la Primary Key, sin embargo, Spark SQL al mantener toda la información en memoria permite realizar consultas sobre cualquier tipo de dato.
- Operaciones optimizadas. La API ofrece multitud de métodos de transformación que abstraen de la complejidad a bajo nivel, garantizando un proceso óptimo.
- Trabajar con una plataforma de código abierto en continuo desarrollo, con una gran comunidad activa.

4. ESTIMADOR:

Ya que nuestro target es una variable discreta vamos a utilizar algoritmos de clasificación, en nuestro caso hemos elegido **random forest y logistic regression** ya que con estos algoritmos se pueden emplear para determinar la enfermedad de una persona de acuerdo a los síntomas que presenta.
Para empezar, aplicamos nuestros algoritmos estableciendo unos parámetros iniciales maxIter y regPram para Regresión Logística y numTrees y maxDepth para Random Fores. Más adelante evaluaremos qué valores son mejores para estos parámetros y así conseguir un resultado más óptimo.

5. TRANSFORMERS:

El proceso que realizan los transformers nos sirve para escalar, convertir o modificar características. Lo primero que vamos a hacer es crear el esquema con los tipos de las variables que vamos a utilizar.
Creamos un transformer del tipo Vector Assembler que guarda todas nuestras variables menos la variable objetivo (Enfermedad) en un vector que llamaremos **features.**

6. PIPELINE:

A continuación, creamos un pipeline que nos sirve para encapsular el estimador y los transformers para ejecutarlos. Haremos dos uno para random forest y otro para logistic regression.

7. CONJUNTOS TRAIN Y TEST:

Dividimos los datos en un 80% para el conjunto train y un 20% para el test, para después entrenar al modelo en logistic regression.
En random forest, dividimos los datos en 70% para el conjunto train y un 30% para el test.
Podemos ir variando las cantidades de los conjuntos y ver cómo influye en el modelo.

8. EVALUACIÓN:

En este apartado evaluamos cómo se comporta el algoritmo, si conseguimos resultados óptimos o si por el contrario tenemos que variar los parámetros para mejorar su funcionamiento. Para ello haremos dos evaluaciones: **el área bajo la curva ROC y el área bajo la curva PR.**

Para entender lo que realmente estamos evaluando con estas mediciones vamos a analizar unos conceptos:

Precisión: es el ratio o porcentaje de clasificaciones correctas de nuestro clasificador.

$$precision = \frac{TP}{TP + FP}$$

TP: verdaderos positivos
FP: falsos negativos

Recall o sensibilidad: de nuestro modelo es el ratio de positivos detectado en el dataset por nuestro clasificador. En otras palabras, de todos los positivos reales de nuestro dataset, detectados o no (TP+FN), cual es el ratio de positivos detectados.

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

TP: verdaderos positivos
FN: falsos negativos

Aumentar la precisión, disminuye el recall (la sensibilidad). Por lo tanto dependiendo de si queremos un modelo más preciso o con mayor sensibilidad, tendremos que ajustar nuestro modelo para encontrar el equilibrio.

- **Alta precisión y alto recall**: el modelo maneja perfectamente esa clase
- **Alta precisión y bajo recall**: el modelo no detecta la clase muy bien, pero cuando lo hace es altamente confiable.
- **Baja precisión y alto recall**: La clase detecta bien la clase pero también incluye muestras de otras clases.
- **Baja precisión y bajo recall:** El modelo no logra clasificar la clase correctamente.

Accuracy: número total de predicciones correctas dividido por el número total de predicciones.

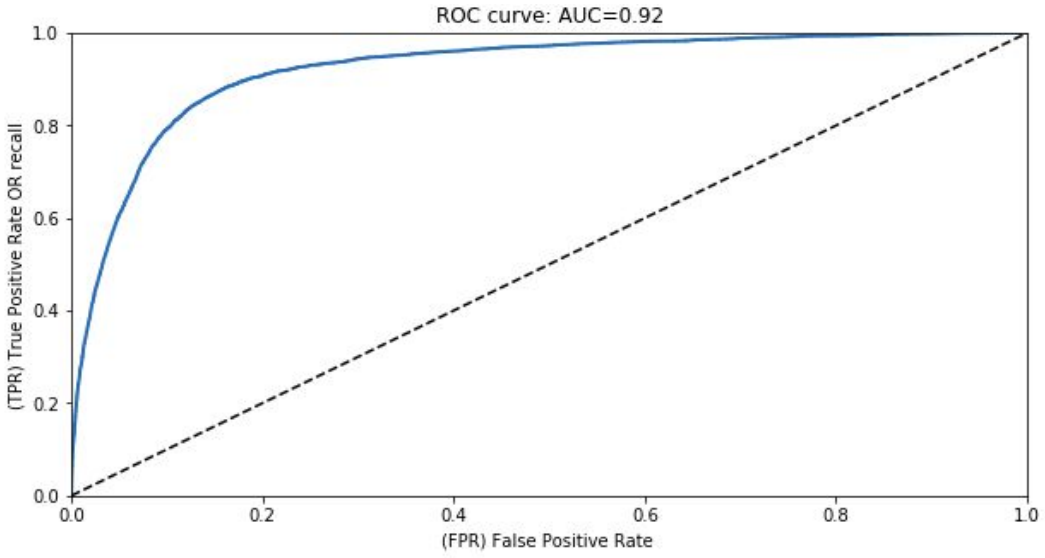
$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{All Samples}}$$

Las **curvas PR y ROC** nos ayudan a ver el **equilibrio entre precisión y sensibilidad** de un modelo.

La **curva PR** es el resultado de dibujar la gráfica entre el precision y el recall. Esta gráfica nos permite ver a partir de qué recall tenemos una degradación de la precisión y viceversa.

El **Average precision (AP)** y es una manera de calcular el área bajo la curva PR. Nos sirve para evaluar y comparar el rendimiento de modelos. Cuanto más se acerque su valor a 1(mayor curva), mejor será nuestro modelo (alta precisión y alto recall).

La **curva Roc** relaciona la sensibilidad de nuestro modelo con los fallos optimistas (clasificar los negativos como positivos). Si aumentamos el recall, nuestro modelo tenderá a ser más optimista e introducirá más falsos positivos en la clasificación. AUC, nos interesa que su valor se acerque lo máximo posible a 1.



9. HYPERTUNING DE LOS PARÁMETROS:

El hyper-tunning de los parámetros consiste en mejorar el modelo estudiando cual resulta ser la mejor combinación de parámetros a introducir en el algoritmo.
Al dar con una combinación optimizada, a su vez mejoraremos los resultados del modelo.
El proceso de ajuste optimiza los parámetros del modelo para que éste se ajuste a los datos de entrenamiento tan bien como pueda.

Creamos un **grid** donde establecemos diferentes valores para los parámetros de su algoritmo correspondiente: maxIter y regPram para Regresión Logística y numTrees y maxDepth para Random Fores. Es decir, haremos el mismo método para ambos algoritmos empleados pero por separado.
El método evalúa todas las combinaciones posibles de los distintos parámetros que hemos definido en el grid y elige los mejores. Los que nos proporcionan un resultado más óptimo.

10. CROSS VALIDATION CON K FOLD:

La validación cruzada es una técnica de partición de datos, que como el método anterior pretende conseguir los resultados más óptimos para el algoritmo pero de una forma distinta. Introducimos el pipeline, los parámetros del grid más óptimos que ya hemos conseguido mediante el método de hypertuning y un clasificador binario que nos evalúa la columna objetivo. Mediante el método de cross validation obtenemos las predicciones y evaluamos cómo de certeras han sido con los métodos empleados anteriormente: curvas Roc y PR.

11. ESTANDARIZAR:

Creamos este método para normalizar variables. Esto ayuda a que el modelo comprenda mejor la relación entre las variables para sacar los coeficientes y mejorar la eficiencia computacional, es decir el tiempo de respuesta del modelo. En nuestro caso hemos seleccionado la columna Colesterol ya que tiene valores que se mueven en un rango más extremo.

12. EXTRACCIÓN DE CARACTERÍSTICAS:

Con este método extraemos las variables que tienen más poder predictivo, es decir las que tienen más influencia en la salida del modelo. Para ello quitamos las variables que no aportan información y evitamos sobrecargar el modelo, mejorando el tiempo de ejecución y el resultado del algoritmo.

13. PCA:

Se define como análisis de componentes principales. Empleamos este método para reducir la dimensionalidad de los datos del dataset para luego emplearlos en el algoritmo, de esta manera nos quedamos con las componentes principales.

14. DISCRETIZACIÓN DE VARIABLES CONTINUAS:

Este método es una transformación de variables continuas a variables discretas ya que ciertos algoritmos, como por ejemplo el random forest, trabajan mejor con variables discretas. En nuestro caso no hemos tenido que emplearlo ya que nuestras variables ya son discretas y por lo tanto nuestros algoritmos funcionan bien con nuestros datos de base.

15. CONCLUSIONES:

A modo de conclusiones, tanto del trabajo realizado este semestre como del propio proyecto que acabamos de realizar, destaca sobre el resto el aprendizaje y entendimiento de Apache Spark, motor utilizado para el procesamiento en memoria de grandes cantidades de datos. En específico, hemos utilizado su la API que proporciona en Python, el mismo lenguaje utilizado para la primera parte de la asignatura. Spark es una plataforma de código abierto, una herramienta rápida y unificada. En añadido, cabe destacar que, a pesar de ser una plataforma de código abierto, su velocidad se debe a su tolerancia a fallos y al hecho de que trabaja en memoria y no en disco, entre otros factores. El almacenamiento de los datos en la memoria hace que la iteración de los algoritmos de machine learning con los datos sea más eficiente.
Una de las características más notables de Apache Spark es que podríamos decir que es una plataforma de plataformas, lo cual agiliza mucho el funcionamiento y el mantenimiento de sus soluciones. Entre ellas, a modo resumen, las que hemos utilizado han sido:
- Spark SQL: permite la consulta de datos estructurados utilizando lenguaje SQL o una API, que nosotros hemos utilizado con Python.
- MLlib (Machine Learning): esta herramienta contiene algoritmos de muchas utilidades, como la regresión logística; modelos de árbol de regresión bayesiana; modelos de mezclas gaussianas; análisis de conglomerados de K medias; regresión lineal…

Todo ello, añadido al estudio anterior que realizamos sobre algoritmos de machine learning, supervisados y no supervisados, es lo que nos ha llevado a la ejecución de este proyecto. Como hemos ido explicando, nuestro objetivo era averiguar según qué características el paciente desarrolla una patología de corazón y ese es el camino que hemos seguido. Hemos utilizado los algoritmos y mejoras anteriormente utilizadas para todo esto y finalmente hemos llegado a la conclusión de que realizando el k fold en Logistic Regression o en Random Forest tras haber optimizado la combinación de parámetros es la manera en la que mejores resultados se obtienen. No obstante, la precisión que obtenemos en PCA tampoco se queda atrás, también son resultados bastante interesantes.

Finalmente, con todas estas cartas sobre la mesa, ambas integrantes del grupo nos damos cuenta de que el trabajo realizado en esta asignatura se va acercando cada vez más a los objetivos de la propia docencia. En añadido, los conocimientos que hemos adquirido a lo largo de estos últimos meses nos están ayudando en el desarrollo del resto de asignaturas, lo que clarifica nuestras ideas haciendo que nos demos cuenta de que poco a poco estamos formando una buena base de cara al mundo laboral.

16. BIBLIOGRAFÍA:

- Curvas ROC: Elección de puntos de corte y área bajo la curva (AUC)
- Curvas PR y ROC - bluekiri
- Clasificación con datos desbalanceados
- ML Tuning - Spark 2.4.5 Documentation
- Exploración y Preprocesamiento de datos usando PySpark — tarjeta de crédito
- Apache Spark: las ventajas de usar al nuevo 'rey' de Big Data
- Muñoz, R. (2020) *Tema 2 Hadoop y Spark.*
- Apache Spark: Introducción a Spark Sql | | Future Bites