Maestría en Explotación de Datos
y Descubrimiento del Conocimiento
Universidad de Buenos Aires

Maestría en Explotación de Datos y Descubrimiento del Conocimiento

Aprendizaje Automático

Trabajo Práctico Nro. 2

Detección de Enfermedades Cardíacas vía Ensambles

Lucas Tomasini

Comisión 1

26 de julio de 2019

Resumen

En este artículo se utilizan cuatro métodos de ensamble distintos (Bagging, Random Forest, AdaBoost y XGBoost) para predecir enfermedades cardiovasculares a partir de mediciones de factores de riesgo de pacientes admitidos en hospitales. Se utilizaron 920 casos clínicos con 13 factores de riesgo. Existen datos faltantes en algunos atributos de ciertos casos, con lo cual se realizó una imputación múltiple antes de correr los modelos. Se buscaron los hiperparámetros óptimos de cada uno mediante *grid search* y utilizando *5-fold cross validation* sobre un conjunto de desarrollo. Finalmente, se compararon los diferentes ensambles evaluando su performance sobre nuevos pacientes utilizados como datos de prueba.

Si bien los cuatro métodos lograron buenos resultados en general (*precision* y *recall* superiores al 85%), Bagging resultó ser el que mejor clasifica de todos, especialmente el que menor cantidad de falsos negativos obtuvo, que es el indicador más importante en este estudio.

El tipo de dolor de pecho (anginoso, no anginoso o asintomático) y los resultados del escaneo de corazón mediante Talio son los atributos que más ayudan al diagnóstico de enfermedades cardiovasculares en un paciente dado. También el nivel de colesterol y la depresión de la onda ST inducida por ejercicio son factores importantes a la hora de clasificar. Por el contrario, los niveles de azúcar en sangre son los que menos influyen en la decisión. Le siguen los resultados electrocardiográficos y la presión arterial en reposo, que también son factores de importancia relativamente baja.

Introducción

El presente trabajo tiene como objetivo evaluar y comparar diferentes modelos de clasificadores basados en ensambles para predecir la aparición de enfermedades cardíacas a partir de una serie de factores de riesgo cardíaco tales como sexo, edad, nivel de colesterol, presión sanguínea, niveles de azúcar en sangre, indicadores electrocardiográficos, entre otros. Existen numerosos trabajos de *Machine Learning* al respecto (ver [1]), pero todos ellos se basan únicamente en el dataset construido a partir de la base de datos de la Clínica de Cleveland. En cambio, en este trabajo se utilizan, además de Cleveland, otros tres conjuntos de datos tomados de instituciones médicas ubicadas en Budapest, Long Beach y Zurich, triplicando así el número de casos disponibles para alimentar los algoritmos de aprendizaje.

Los métodos de ensamble que se analizan son: *Bagging*, *Random Forest*, *AdaBoost* y *XGBoost*, todos utilizando Árboles de Decisión como predictores de base. Para cada método se realiza una búsqueda exhaustiva de aquellos hiperparámetros que maximicen el F1-score sobre el conjunto de validación. Finalmente se compara cada modelo evaluando diferentes métricas de performance sobre los datos de testeo, separados desde un principio. Las métricas a considerar son: *accuracy*, *precisión*, *recall*, *F1*, *TPR*, *FPR*, *TNT* y *FNR*.

El trabajo se encuentra dividido en varias partes, donde se explican los diferentes pasos que se tuvieron en cuenta para la construcción de los modelos. En la sección *Datos* se describen los diferentes conjuntos de datos y sus atributos, así como las diferentes transformaciones que se realizaron para poder utilizarlos. En la sección *Metodología* se detallan los algoritmos y los diferentes hiperparámetros utilizados para cada uno de los métodos de ensamble. Por último, en la sección *Resultados* se muestran los resultados obtenidos en cada caso junto con una comparación de la performance de cada modelo.

Datos

Los conjuntos de datos originales se obtuvieron de un repositorio de la UCI creado por David W. Aha [2] en 1988. El mismo contiene 4 bases de datos con diagnósticos de enfermedades cardiovasculares recolectados de las siguientes instituciones médicas:

- Cleveland Clinic Foundation
- Hungarian Institute of Cardiology, Budapest
- V.A. Medical Center, Long Beach, CA
- University Hospital, Zurich, Switzerland

Cada conjunto de datos tiene los mismos 14 atributos, todos ellos representados por números. Los investigadores principales de cada institución responsables de la recopilación de los mismos se mencionan en [3]. El número de instancias de cada dataset es:

- Cleveland: 303
- Hungarian: 294
- Switzerland: 123
- Long Beach V.A.: 200

Se integraron las 4 fuentes de datos en un único *dataframe* de 920 casos que utilizaremos para entrenar los diferentes clasificadores. El atributo objetivo a predecir se refiere a la presencia de enfermedad cardíaca de un paciente determinado. Es un valor entero que va de 0 a 4 que binarizamos con el objetivo de distinguir presencia (valores 1, 2, 3 o 4) de ausencia de enfermedad (valor 0). Los 13 atributos restantes representan los *features* de entrada:

- **age:** edad del paciente en años
- **sex:** sexo del paciente (1: hombre, 0: mujer)
- **cp:** tipo de dolor de pecho (1: angina típica, 2: angina atípica, 3: dolor no anginoso, 4: asintomático)
- **trestbps:** presión arterial en reposo en mmHg en el momento de la admisión al hospital.
- **chol:** niveles de colesterol sérico en mg/dl
- **fbs:** niveles de azúcar en sangre en ayunas mayor a 120 mg/dl (1: *true*, 0: *false*)
- **restecg:** resultados electrocardiográficos en reposo
    - 0: normal
    - 1: anomalía en la onda ST-T (inversión de la onda T y/o elevación o depresión mayor a 0.05 mV de la onda ST)
    - 2: probable o definitiva hipertrofia ventricular izquierda de acuerdo a los criterios de Romhilt-Estes
- **thalach:** frecuencia cardíaca máxima alcanzada en *rpm*
- **exang:** angina inducida por ejercicio (1: si, 0: no)
- **oldpeak:** depresión de la onda ST inducida por ejercicio relativa al reposo.
- **slop:** pendiente del segmento ST durante el ejercicio pico (1: ascendente, 2: plana, 3: descendente)
- **ca:** número de vasos principales coloreados por fluoroscopía (de 0 a 3)
- **thal:** escaneo del corazón utilizando Talio
    - 3: normal (no se observan puntos fríos)
    - 6: defecto fijo (puntos fríos durante el reposo y el ejercicio)
    - 7: defecto reversible (puntos fríos sólo aparecen durante el ejercicio)

Nótese que los atributos *age*, *trestbps*, *chol*, *thalach* y *oldpeak* son atributos numéricos (continuos o discretos) mientras que el resto son del tipo categóricos nominales. También existe una cantidad importante de valores faltantes en diferentes celdas (sobre todo en los atributos *slop*, *ca* y *thal*), los cuales imputamos de forma iterativa mediante técnicas de regresiones múltiples, resultando en un dataframe final de 920 filas y 14 columnas, sin valores faltantes.

Metodología

Como se mencionó en la *Introducción*, en este trabajo se compara la performance de 4 métodos de ensamble: *Bagging*, *Random Forest*, *AdaBoost* y *XGBoost*. En cada caso trabajamos con una partición del dataset aplicando 5-fold cross-validation sobre el conjunto de desarrollo para evaluar diferentes hiperparámetros mediante el F1-score como métrica de performance. El objetivo es minimizar los falsos negativos y, por lo tanto, maximizar el *recall*, ya que hay un gran costo en diagnosticar a un paciente como sano cuando en realidad no lo está. Pero utilizando únicamente el *recall* como métrica de performance se corre el riesgo de encontrar modelos con un *precision* muy bajo, con lo cual es necesario un *score* que tenga en cuenta ambos factores. El *F2-score* es una alternativa adecuada en este caso, ya que tiene en cuenta el *precision* pero le da un mayor peso al *recall*. De todas formas, durante el desarrollo de este trabajo se comprobó que tanto utilizando *F1* como *F2* se obtiene un recall similar, siendo que con F1 se logra mejorar el *precision*. Por ese motivo, *F1-score* es la métrica utilizada para evaluar la performance de los modelos.

1. *Imputación de datos faltantes*

La mayoría de los atributos contienen valores faltantes, especialmente *slop*, *ca* y *thal*. Utilizamos una técnica de imputación múltiple que modela a cada variable con valor faltante como una función del resto de las variables, y usa como imputación el valor estimado a partir de dicha función. Lo hace en forma iterativa: en cada paso, se designa una columna como variable de salida *y* y el resto como entradas *X*. Se calcula una regresión en *(X, y)*, para *y* conocido y se utiliza esa regresión para predecir los valores faltantes de *y*. Se realiza este proceso para cada *feature* con valores faltantes de forma iterativa y se repite un máximo de veces especificado (1000 en este caso).

2. *Visualización de atributos y sus relaciones*

Luego de la imputación graficamos los diferentes atributos para ver cómo se distribuyen sus valores: utilizamos *box-plots* para atributos continuos y gráficos de barra con frecuencias absolutas para atributos discretos. Para atributos continuos también mostramos los histogramas y los gráficos de dispersión de a pares, pero agrupando de acuerdo al target de salida (0 ó 1), para poder visualizar cómo difieren las distribuciones y las relaciones entre atributos según la clase.

3. *Partición de datos*

El primer paso para poder entrenar y testear los modelos con dos conjuntos diferentes fue partir los datos en un conjunto de desarrollo y un conjunto de prueba, de forma aleatoria y en una proporción 80/20. El conjunto de prueba se deja aparte para ser utilizado al final, durante la etapa de comparación de los diferentes métodos de ensamble. Con el conjunto de desarrollo realizamos una búsqueda de los hiperparámetros óptimos de cada modelo mediante grid-search (o randomized-search según corresponda) utilizando 5-fold cross-validation. Para cada método de ensamble se informan los valores óptimos de accuracy, precision, recall, F1, TPR, FPR, TNR y FNR sobre los conjuntos de validación (valores promedios calculados sobre cada fold) y sobre el conjunto de prueba (valores finales sobre los cuales se comparan los diferentes modelos).

4. *Bagging o Boostrap Aggregating*

Este es el primer método de ensamble que implementamos. Los hiperparámetros a optimizar, junto con el rango de valores posibles que van a ser evaluados mediante *grid-search* son:

- **max_samples** *(entre 0.1 y 1 en pasos de 0.05)*: la proporción de muestras a tomar del conjunto de entrenamiento para entrenar cada estimador base.
- **max_features** *(de 2 a 13)*: el número de *features* a tomar del conjunto de entrenamiento para entrenar cada estimador base.
- **bootstrap** *(True/False)*: determina si las muestras se toman con reemplazo.
- **bootstrap_features** *(True/False)*: determina si los *features* se toman con reemplazo.

5. *Random Forest*

En este segundo método los hiperparámetros a optimizar junto con su rango de valores son:

- **criterion** *("gini" o "entropy")*: función que mide la calidad de un split.
- **max_depth** *(de 2 a 13)*: altura máxima del árbol
- **max_features** *("sqrt", "log2", None)*: el número de *features* a tomar del conjunto de entrenamiento para entrenar cada estimador base. *None* significa tomar todos los features.
- **bootstrap** *(True/False)*: determina si las muestras se toman con reemplazo.
- **min_sample_split** *(de 2 a 5)*: Mínimo número de muestras requerido para separar un nodo interno.
- **min_sample_leaf** *(de 1 a 5)*: Mínimo número de muestras requerido para poder ser un nodo hoja.

6. *AdaBoost*

En este tercer método los hiperparámetros a optimizar junto con su rango de valores son:

- **base_estimator** *(árbol de profundidad 1 o 2)*: el estimador base utilizado.
- **n_estimators** *(de 50 a 400 en pasos de 10)*: el número máximo de estimadores base a utilizar (cantidad de iteraciones)
- **learning_rate** *(0.01 a 0.1 en pasos de 0.005)*: tasa de aprendizaje que controla la contribución de cada estimador.

7. *XGBoost*

En el método de XGBoost hay una gran cantidad de hiperparámetros para buscar, con lo cual hacer un grid-search es impracticable. Por lo tanto realizamos una búsqueda aleatoria con un gran número de iteraciones (50.000) para asegurarnos de encontrar valores óptimos o cuasi-óptimos. Los hiperparámetros a optimizar junto con su rango de valores son:

- **n_estimators** *(de 10 a 200 en pasos de 10)*: el número máximo de estimadores base a utilizar (cantidad de iteraciones)
- **learning_rate** *(0.01 a 0.3 en pasos de 0.005)*: tasa de aprendizaje que controla la contribución de cada estimador.
- **subsample** *(0.5 a 1 en pasos de 0.1)*: la proporción de muestras a tomar del conjunto de entrenamiento para entrenar cada estimador base.
- **colsample_bytree** *(4/13, 7/13, 10/13 y 1)*: la proporción de *features* a tomar del conjunto de entrenamiento cada vez que se construye un árbol.
- **colsample_bylevel** *(1/4, 1/2, 3/4, 1)*: la proporción de *features* (respecto de los ya elegidos para cada árbol vía *colsample_bytree*) a utilizar en cada nivel del árbol.
- **colsample_bynode** *(1/4, 1/2, 3/4, 1)*: la proporción de features (respecto de los ya elegidos para un dado nivel vía *colsample_bylevel*) a utilizar en cada nodo cada vez que se evalúa un nuevo split.
- **reg_alpha** *(0.1 a 1 en pasos de 0.1)*: parámetro de regularización L1 sobre los pesos.
- **reg_lambda** *(0.1 a 1 en pasos de 0.1)*: parámetro de regularización L2 sobre los pesos.

Resultados

1. *Visualización de los datos*

Se trata de un dataset cuyas clases están bastante equilibradas. De los 920 casos, hay un 55.33% (509 casos) de pacientes con afecciones cardíacas y un 44.67% sanos (411 casos). En la *Figura 1* mostramos histogramas (en la diagonal) y gráficos de dispersión de los atributos continuos, separados por clase. Los gráficos de dispersión muestran la relación entre cada uno de los atributos tomados de a dos. Se observa a simple vista que ninguno de estos atributos por sí solos o de a pares logran separar claramente los pacientes enfermos de los sanos. En la Figura 2 y 3 graficamos respectivamente los box-plots de las variables continuas y los gráficos de barra de las variables discretas para ver su distribución.

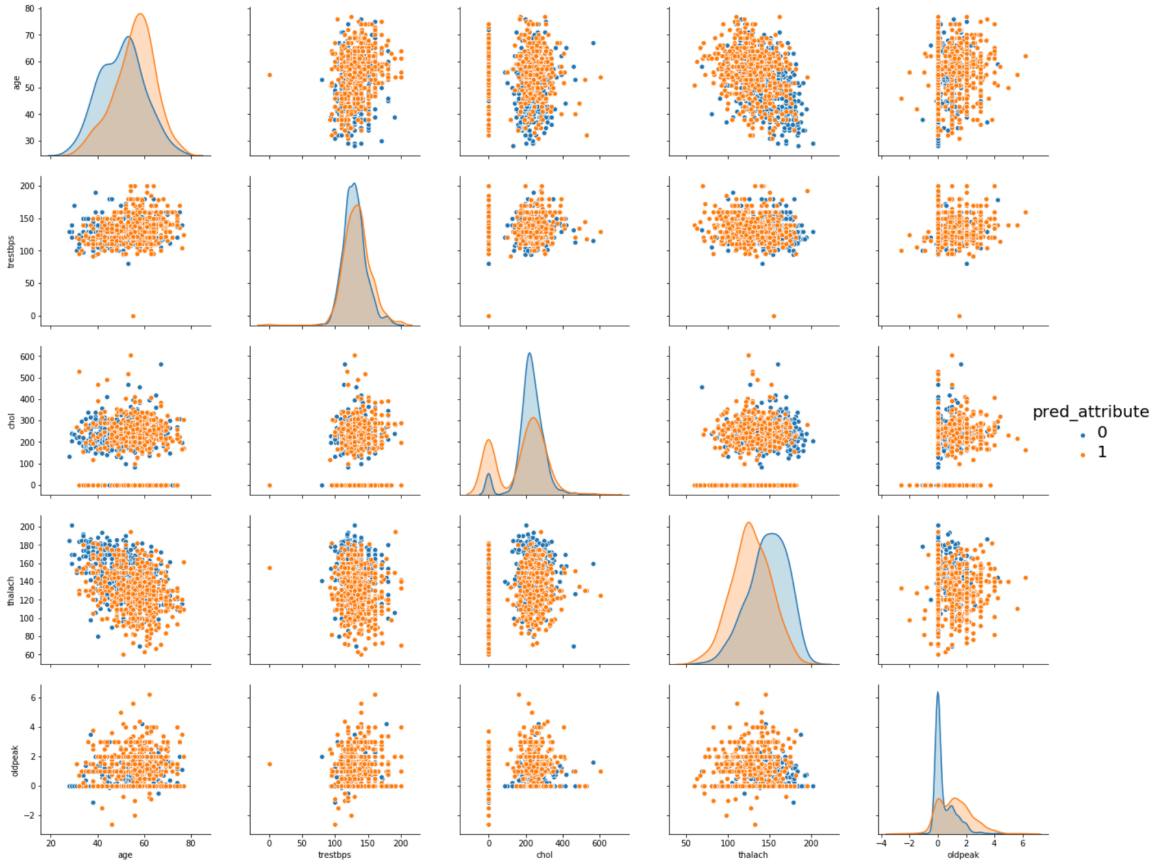

Figura 1 – Histogramas y gráficos de dispersión de los atributos continuos

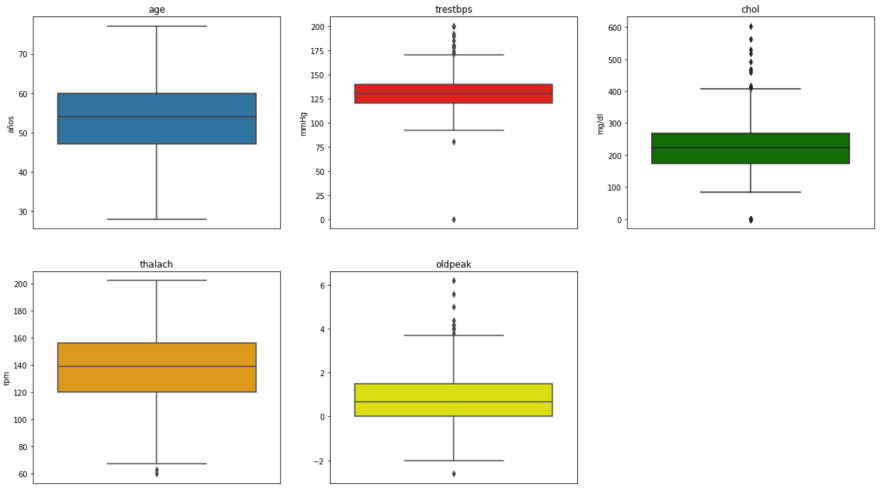

Figura 2 – Box-plots de los atributos continuos

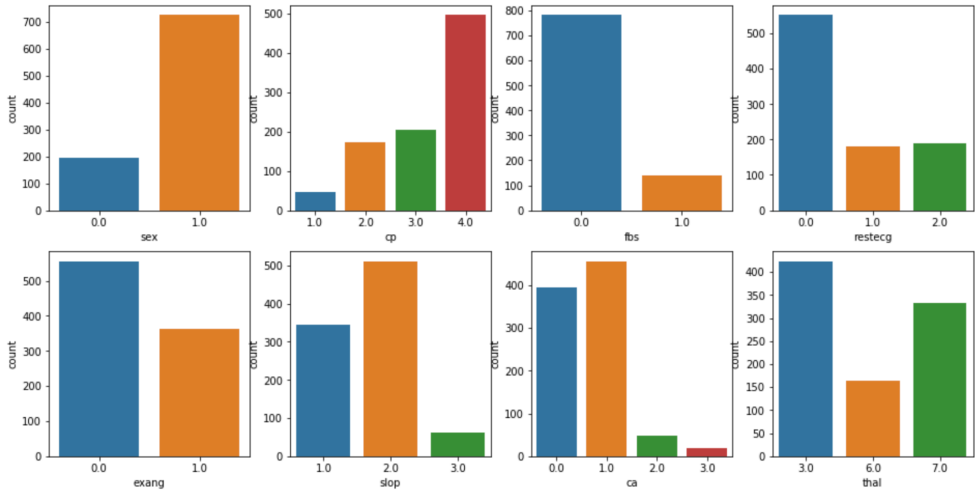

Figura 3 – Gráficos de barra para atributos discretos

2. *Hiperparámetros óptimos y CV scores para cada ensamble*

Bagging: fijamos el número de iteraciones en 200, un número lo suficientemente grande como para minimizar la varianza. El resto de los hiperparámetros encontrados vía grid-search son:

- *max_samples = 0.3*
- *max_features = 10*
- *bootstrap = False*
- *bootstrap_features = True*

CV Scores (%, promedios sobre los 5 folds)							
Accuracy	**Precision**	**Recall**	**F1**	**TPR**	**FPR**	**TNR**	**FNR**
84.65 ± 1.30	84.45 ± 2.32	88.70 ± 2.45	86.47 ± 1.07	49.29	9.07	35.36	6.28

Random Forest: igual que en el caso de Bagging. Los hiperparámetros óptimos son:

- *criterion = gini*
- *max_depth = 5*
- *max_features = 'sqrt'*
- *bootstrap = True*
- *min_sample_leaf = 2*
- *min_sample_leaf = 2*

CV Scores (%, promedios sobre los 5 folds)							
Accuracy	**Precision**	**Recall**	**F1**	**TPR**	**FPR**	**TNR**	**FNR**
83.97 ± 1.07	84.24 ± 2.06	87.47 ± 1.75	85.79 ± 0.79	48.54	9.08	35.43	6.95

AdaBoost: los hiperparámetros óptimos son:

- *base_estimator = árbol de profundidad 2*
- *n_estimators = 230*
- *learning_rate = 0.02*

CV Scores (%, promedios sobre los 5 folds)							
Accuracy	**Precision**	**Recall**	**F1**	**TPR**	**FPR**	**TNR**	**FNR**
84.65 ± 1.80	85.23 ± 1.97	87.47 ± 3.45	86.28 ± 1.74	48.50	8.40	36.14	6.95

XGBoost: en este caso la altura máxima de los árboles la fijamos en 10 y el parámetro *scale_pos_weight*, que representa el balance de pesos positivos y negativos, lo fijamos en 0.8733, que es el cociente entre cantidad de instancias negativas y positivas (valor recomendado en la literatura para este hiperparámetro). El resto los obtenemos mediante búsqueda aleatoria:

- *n_estimators = 20*
- *learning_rate = 0.01*
- *subsample = 0.75*
- *colsample_bytree = 0.8*
- *colsample_bylevel = 0.25*
- *colsample_bynode =0.5*
- *reg_alpha = 0*
- *reg_lambda = 0.3333*

CV Scores (%, promedios sobre los 5 folds)							
Accuracy	**Precision**	**Recall**	**F1**	**TPR**	**FPR**	**TNR**	**FNR**
85.05 ± 1.01	85.49 ± 1.23	87.96 ± 2.89	86.66 ± 1.12	48.74	8.28	36.31	6.67

3. *Evaluación de los modelos*

En la Tabla 1 se muestra las 8 métricas para cada método de ensamble, todas calculadas sobre el conjunto de prueba. A su vez, y revelando información análoga, en la Figura 4 se visualizan las matrices de confusión de cada modelo.

%	Accuracy	Precision	Recall	F1	TPR	FPR	TNR	FNR
Bagging	88.04	86.36	93.14	89.62	51.63	8.15	36.41	3.80
RandomForest	85.87	85.19	90.20	87.62	50.00	8.70	35.87	5.43
AdaBoost	85.33	85.71	88.24	86.96	48.91	8.15	36.41	6.52
XGBoost	85.87	85.85	89.22	87.50	49.46	8.15	36.41	5.98

Tabla 1 – Comparación de los diferentes métodos

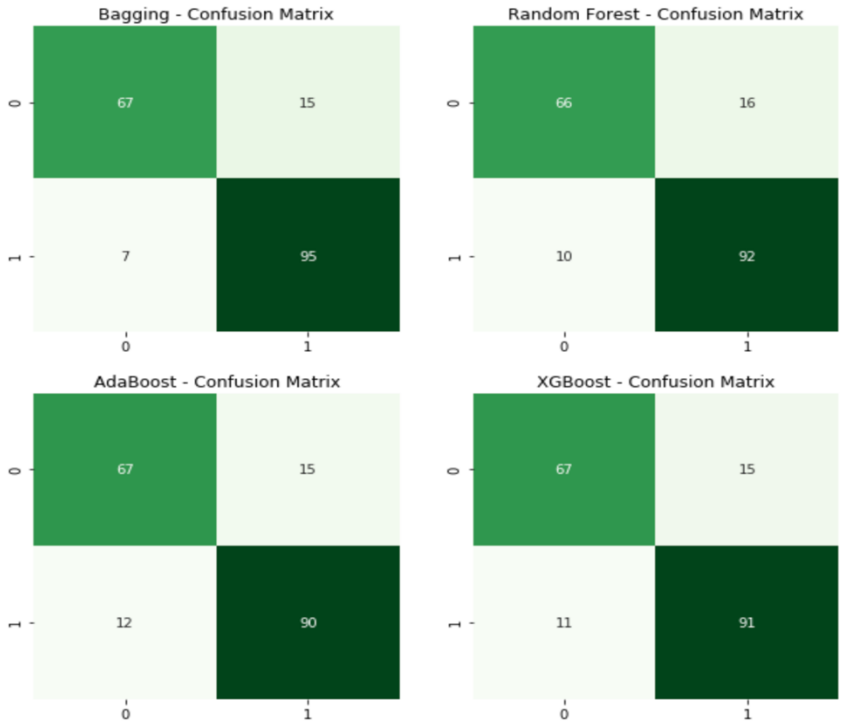

Figura 4 – Matrices de confusión

En las tres figuras siguientes se muestra, para los métodos de Random Forest, AdaBoost y XGBoost respectivamente, la importancia de cada atributo, que provee una medida de cuán útil o valioso es un atributo en la construcción de los árboles de decisión dentro de los ensambles. Cuanto más se utilice un determinado *feature* para tomar decisiones a la hora de dividir un nodo, mayor será la importancia relativa de ese *feature.*

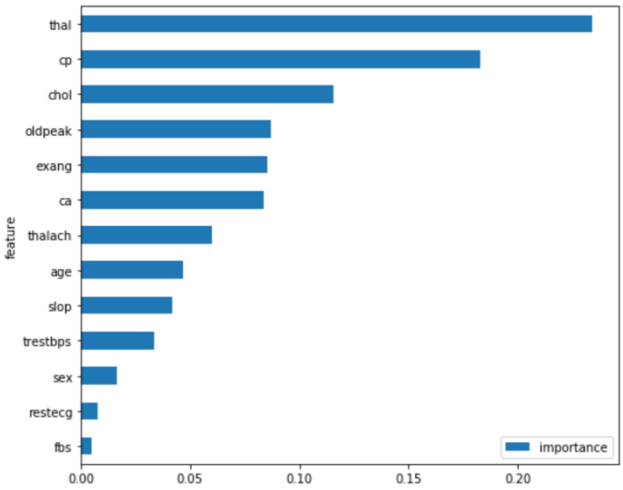

Figura 5 – Feature importance Random Forest

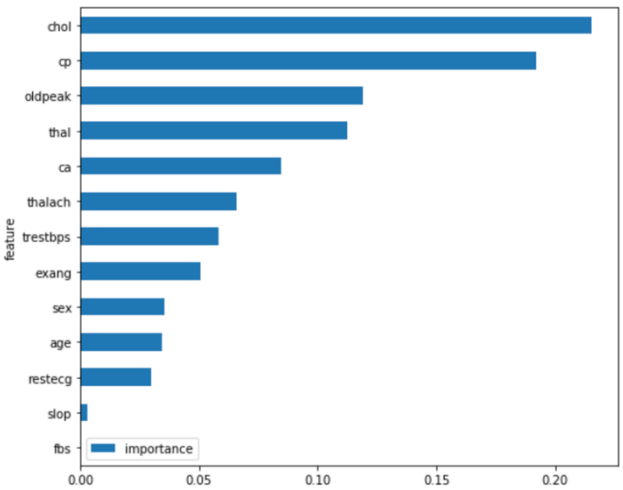

Figura 6 – Feature importance AdaBoost

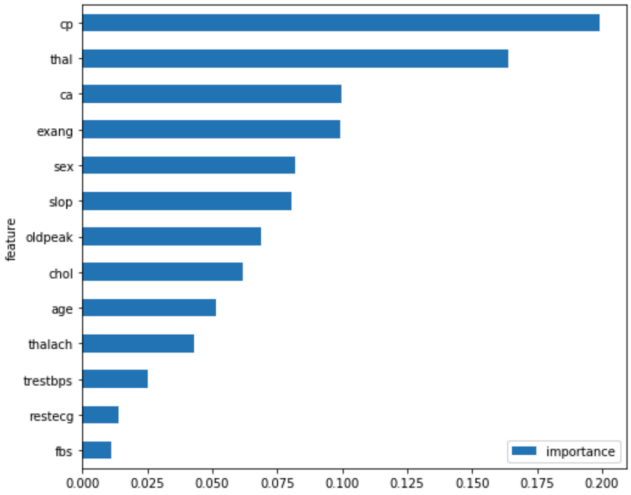

Figura 7 – Feature importance XGBoost

Conclusiones

Si bien los cuatro métodos dan resultados excelentes, el mejor de todos resultó ser el Bagging (obtuvo el mayor puntaje en todas sus métricas). En cuanto al *recall*, métrica más importante en este análisis (queremos minimizar los falsos negativos), el Bagging obtuvo un 93.14%, un 3% más que el Random Forest (90.20%), que es el segundo método más importante del ranking. Desde una perspectiva análoga, de la matrices de confusión se observa que hubo 10 falsos negativos de 184 ejemplos utilizando Random Forest, mientras que con Bagging solamente 7. Los métodos de *boosting* lograron un *recall* levemente inferior a los dos primeros (aunque bueno de todas formas): 88.24% y 89.22% para AdaBoost y XGBoost respectivamente, que equivale a 12 y 11 falsos negativos de 184 ejemplos.

De los gráficos de importancia de los *features*, vemos que todos concuerdan en que el tipo de dolor de pecho (*cp*) y el escaneo del corazón (*thal*) son de los atributos más importantes: *cp* está segundo en Random Forest y AdaBoost y primero en XGBoost, mientras que *tha*l primero, cuatro y segundo en el mismo orden. Para AdaBoost, el nivel de colesterol (*chol*) es el atributo más importante y en Random Forest está tercero. La depresión de la onda ST inducida por ejercicio (*oldpeak*) también está entre los primeros en general. Inversamente, los niveles de azúcar en sangre en ayunas (*fbs*) están últimos en importancia en los 3 gráficos. Los resultados electrocardiográficos en reposo (*restecg*) y la presión arterial en reposo (*trestbps*) también están entre los menos puntuados.

Referencias

1. https://www.kaggle.com/ronitf/heart-disease-uci/kernels
2. https://archive.ics.uci.edu/ml/datasets/Heart+Disease
3. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
   University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
   University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
   V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D.