TP3 Reconocimiento de Patrones

Corvalan César, Junqueras Juan, Suárez Juan Carlos

Julio 2021

Introducción

En el presente trabajo práctico se aplica Random forest sobre la base de datos de la South African heart disease, con el objetivo de clasificar a los individuos de acuerdo a la probabilidad de sufrir un infarto o no. Esta clasificación se realiza según el resto de los atributos incluídos en el dataset.

1. Marco teórico

La clasificación es uno de los principales objetivos de machine learning. Consiste en determinar ó predecir la pertenencia de objetos (observaciones, individuos) a una clase, sobre la base de variables cuantitativas y cualitativas. Esta posibilidad de clasificar de manera precisa en base a diversas observaciones o datos es fundamental en distintos ámbitos de la vida moderna. Por ejemplo, en instituciones financieras determinar la probabilidad de otorgar una tarjeta ó un crédito, en informatica determinar que es spam y que no, o en medicina determinar si un tumor es benigno o no lo es. Para entender de que manera machine learning realiza estas clasificaciones es necesario analizar el funcionamiento de árboles de decisión básicos, y luego ver cómo se combinan para crear un random forest

1.1. Árboles de decisión.

Un árbol de decisión es un mapa de los posibles resultados de una serie de decisiones relacionadas. Éstos árboles son fundamentales en la construcción de los algoritmos de random forest. Veremos su funcionamiento con un ejemplo.

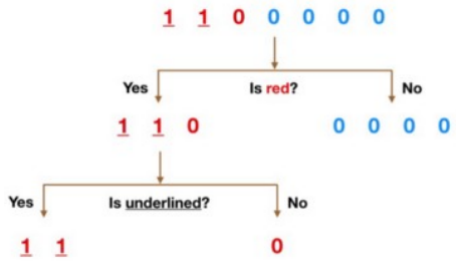

Figura 1: Árbol de decisión.

Imaginemos que nuestro dataset se compone de los números que aparecen arriba, en la raíz del árbol, en la figura 1. Tenemos dos unos y cinco ceros (unos y ceros son nuestras clases) y queremos separar las clases usando sus atributos (features). Los atributos son color (rojo y azul) y si la observación (un número en particular) está subrrayado o no.

En una primera instancia, parecería que el color es un buen atributo para separar las clases, pues todos menos uno de los ceros son azules. Por lo que podemos usar la pregunta "¿Es rojo?"para separar en nuestro primer nodo. El nodo de un árbol puede pensarse como el punto en el que el camino se divide en dos -las observaciones que cumplan con el criterio, irán a la rama "Sí" y los que no, a la "No".

La rama “No” (los azules) son todos ceros por lo que no debemos continuar haciendo preguntas, pero la rama “Sí” puede seguir separándose. Ahora usamos el segundo atributo y preguntamos “¿Está subrayado?” para hacer una segunda separación.

Los dos unos que están subrayados irán a la subrrama “Sí” y el cero que no está subrrayado irá a la subrrama derecha y terminamos. Nuestro árbol de desición fue capaz de separar los datos perfectamente.

Por supuesto, en la vida real, nuestra data no será tan limpia, pero la lógica que utiliza un árbol de desición sigue siendo la misma. En cada nodo preguntará: ¿Qué atributo me permitirá separar las observaciones de manera que los grupos resultantes sean lo más distinos entre sí posible (y los miembros de cada subgrupo resultante sean lo más similares entre sí que se pueda)?

1.2. El clasificador Random Forest.

Como su nombre lo sugiere, los Random Forest son un gran número de Árboles de Decisión que operan en ensamble. Cada árbol dentro el bosque devuelve predicciones de clasificación y la clase con más votos se convierte en la predicción del modelo (ver figura 2).

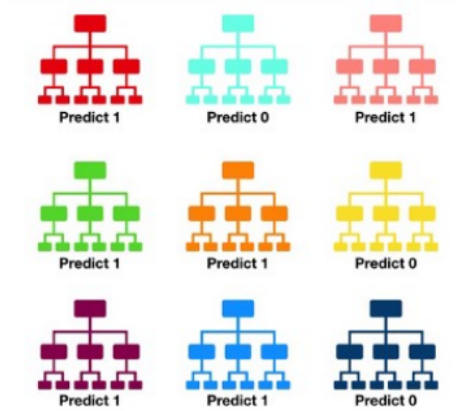

Figura 2: Random Forest.

En términos de data science: “Un gran número de modelos (árboles) relativamente no correlacionados operando en conjunto tendrá una mejor performance que cualquiera de los modelos constituyentes (del conjunto)”.

La razón de este efecto es que los árboles se “protegen” los unos a los otros de sus errores individuales, siempre y cuando no todos fallen constantemente en la misma dirección, por esto la baja correlación entre modelos es la clave. Cuando algunos árboles fallen, otros estarán en lo correcto, esto significa que como grupo los árboles serán capaces de moverse en la dirección correcta. Entonces los prerrequisitos para que los Random Forest funcionen bien son:

- Es necesario que haya señales en nuestros features de manera que los modelos construidos utilizando estos features sean mejores que adivinar.
- Las predicciones (y por lo tanto los errores) hechas por los distintos árboles deben tener una baja correlación entre sí.

1.3. South African Heart Disease dataset.

La base de datos utilizada en este trabajo es un subconjunto del estudio de referencia Coronary Risk-Factor Study (CORIS), llevado a cabo en tres áreas rurales de Western Cape, Sudáfrica (Rousseauw et al., 1983). El objetivo de este estudio fue establecer la intensidad de los factores de riesgo, listados en el cuadro de la figura 3, de enfermedad isquémica del corazón en una región de alta incidencia. Los datos representan a hombres (exclusivamente) caucásicos de entre 15 y 64 años, a los cuales se les consulta por la presencia o ausencia de infarto de miocardio (MI) al momento de la encuesta.

La siguiente tabla muestra las variables de estudio con las cuales se intenta concluir la relación entre ciertos hábitos y enfermedades con la probabilidad de sufrir un infarto de miocardio.

sbp	systolic blood pressure
tobacco	cumulative tobacco (kg)
ldl	low densiity lipoprotein cholesterol
adiposity	
famhist	family history of heart disease (Present, Absent)
typea	type-A behavior
obesity	
alcohol	current alcohol consumption
age	age at onset
chd	response, coronary heart disease

Figura 3: Variables.

2. Desarrollo

Uno de los primeros pasos a la hora de realizar un proyecto que involucre el análisis de datos es explorar y visualizar los datos. El objetivo principal es obtener información sobre el contenido de los datos, ayudar a enmarcar las preguntas que haremos y detectar posibles vías para avanzar en las respuestas a estas preguntas.

2.1. Análisis del dataset

Como se explicó en la sección 1.3, se trabajó con el conjunto de datos llamado South African Heart Disease. El método $info()$ nos permite obtener información útil (una descripción rápida) de los datos, en particular el número total de filas, el tipo de cada atributo y el número de valores no nulos, cómo se puede apreciar en la figura 4.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 462 entries, 1 to 463
Data columns (total 10 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   sbp        462 non-null    int64
 1   tobacco    462 non-null    float64
 2   ldl        462 non-null    float64
 3   adiposity  462 non-null    float64
 4   famhist    462 non-null    object
 5   typea      462 non-null    int64
 6   obesity    462 non-null    float64
 7   alcohol    462 non-null    float64
 8   age        462 non-null    int64
 9   chd        462 non-null    int64
dtypes: float64(5), int64(4), object(1)
memory usage: 39.7+ KB
```

Figura 4: Método $info()$ aplicado al dataset

Se puede ver que hay 462 entradas, con 10 columnas cada una. Hay columnas de varios tipos: cinco de tipo ($float64$), cuatro de tipo ($int64$), y una ($object$). Además, se puede ver que no falta ninguna entrada para ninguna columna.

Imprimimos por pantalla los primeros cinco registros para visualizar los datos y formatos del database

```
df.head(5)
```

row.names	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
1	160	12.00	5.73	23.11	1	49	25.30	97.20	52	1
2	144	0.01	4.41	28.61	0	55	28.87	2.06	63	1
3	118	0.08	3.48	32.28	1	52	29.14	3.81	46	0
4	170	7.50	6.41	38.03	1	51	31.99	24.26	58	1
5	134	13.60	3.50	27.78	1	60	25.99	57.34	49	1

Figura 5: Primeros cinco datos del dataset antes de ser modificados.

Por otro lado, si analizamos los histogramas de la Figura 6 se destaca lo siguiente:

1. Se puede apreciar que la edad de los individuos encuestados está distribuída de forma uniforme.
2. El atributo *chd* tiene sólamente dos columnas, porque es una variable binaria: sufrieron infarto, no sufrieron infarto.
3. El atributo *tobacco* muestra que más de 160 individuos tienen 0 kg de *cummulativetobacco*, es decir, que un porcentaje significativo de la muestra son no fumadores.
4. El atributo *alcohol* indica que más de 175 de los encuestados no consume alcohol actualmente y que el resto de los individuos toma poco alcohol.

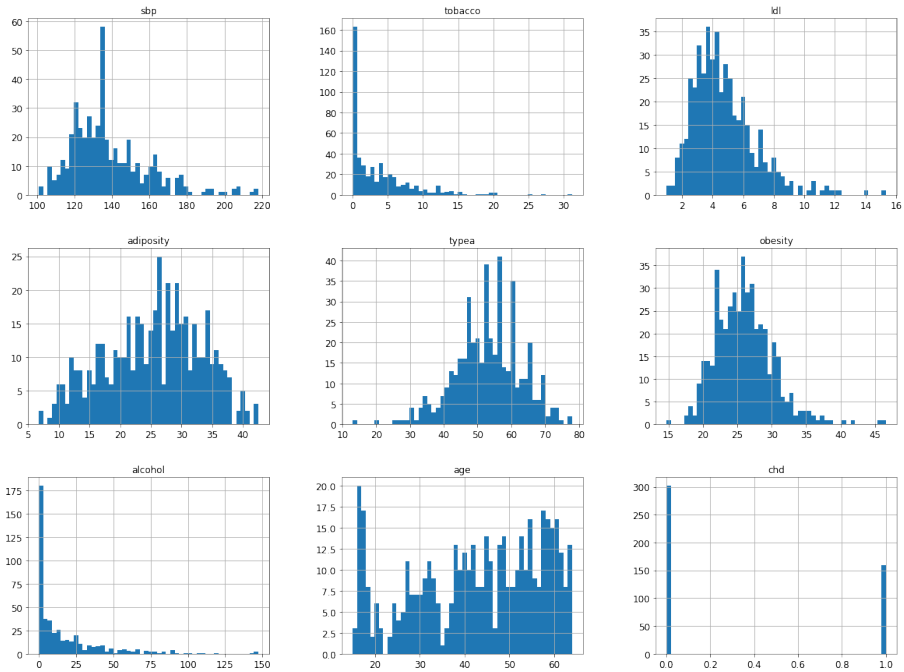

Figura 6: Histograma de cada atributo del South African Heart Disease dataset.

Correlación de datos

Se utilizó el coeficiente de correlación de Pearson para obtener la correlación entre cada par de atributos. El mapa de calor o heatmap de la figura 7 muestra los siguientes resultados:

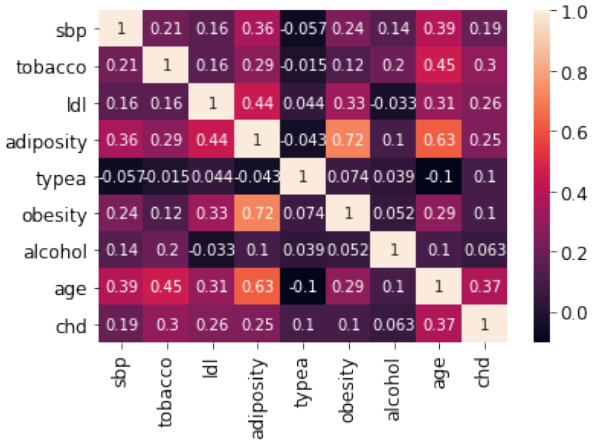

Figura 7: Heatmap de correlacion de todos los pares de atributos

Se puede observar que el atributo *obesity* tiene alta correlación con *adiposity*, como es de esperar, pero también y menos obvio, es que *age* correlaciona con *adiposity*. Esto último sugiere que los individuos desarrollan mayor adiposidad a medida que envejecen. Además podemos notar que los atributos con los que *chd* está más correlacionado son *age*, *ldl* y *adiposity*, es decir, que un individuo haya sufrido un chd está directamente correlacionado con su edad (cuanto más años tiene la persona, más probable es que sufra una chd), con su nivel de colesterol 'malo' (*ldl*) y con su nivel de *adiposidad*. Por otro, lado el atributo *typea* está inversamente correalacionado con muchos de los otros atributos, en particular con: *sbp*, *tobacco*, *adiposity* y *age*; es decir, el comportamiento de tipo A está inversamente correlacionada con la presión arterial sistólica, el tabaco acumulado, la adiposidad y la edad. Sin embargo la correlación, tanto directa como inversa, entre *typea* y el resto de los atributos es relativamente baja. Por último, es notable que ninguno de los atributos correlaciona con valores superioares a 0,5 con *chd*, pero todos aportan información en términos de correlación, teniendo todos valores superiores a 0,1, excepto alcohol que muestra estar poco correlacionado con *chd*.

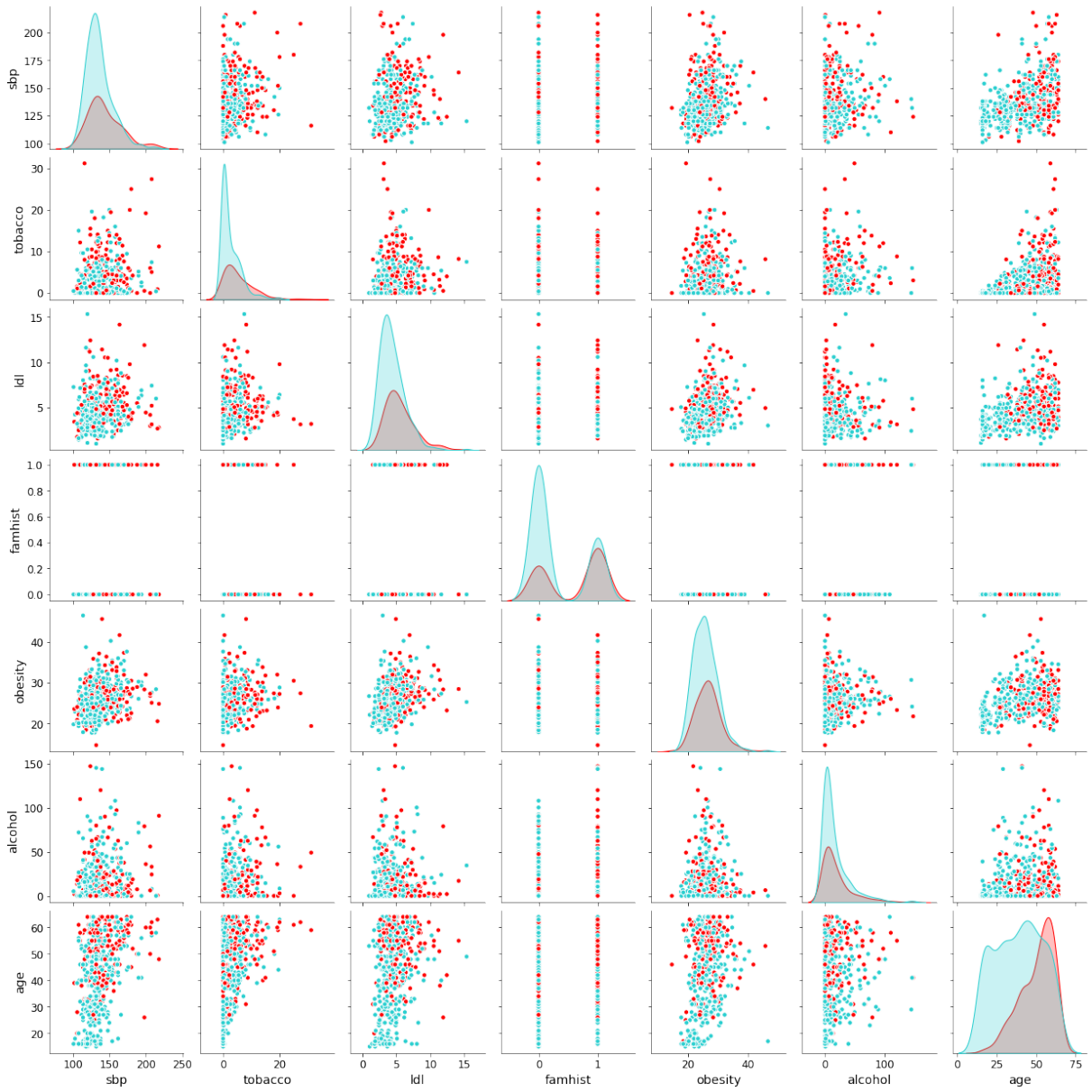

Figura 8: `Pairplot` de los factores de riesgo: los casos están representados con rojo y los controles con azul.

Otra forma de comprobar la correlación entre atributos es utilizar la función `pairplot`, que traza todos los pares de atributos, cada plot muestra un par de factores de riesgo (atributos), los casos están represenados con el color rojo y los controles con azul. La variable que representa la existencia de casos de enfermedades cardíacas en la familia (famhis) es binaria. Estos plots se ven en la figura 8. Idealmente nos gustaría ver un par de atributos que separen por color en dos clusters lo más distantes posible, esto indicaría que este par de atributos tiene una fuerte correlación con el hecho de que un individuo sufra o no *chd*, pero como podemos ver en la imágen, esto no sucede. Además, al observar la diagonal de la matriz, se nota que las curvar rojas y azules, en la mayoría de los casos, se pisan, por lo tanto, no separan bien los positivos y negativos de chd, pero si se observa con atención, la celda de la diagonal correspondiente al atributo *age* es la que más separa los dos casos (la curva roja y azul), lo que coincide con lo visto en la figura 7: el atributo *age* es el que más se correlaciona con *chd*. Todo esto nos sugiere que preguntar por cada uno de los atributos de a uno, no servirá para clasificar correctamente, es decir, los árboles de decisión no serán de mucha ayuda, necesitaremos un modelo más complejo que combine todas estas

variables, por eso utilizaremos $RandomForests$.

Preparar los datos para los algoritmos de Machine Learning

Antes de invocar a los algoritmos de machine learning es necesario preparar los datos para:

- Poder reproducir estas transformaciones fácilmente en cualquier conjunto de datos (p. Ej., la próxima vez que obtenga un conjunto de datos nuevo).
- Construir gradualmente una biblioteca de funciones de transformación que se podrá reutilizar en proyectos futuros.
- Probar fácilmente varias transformaciones y ver qué combinación de transformaciones funciona mejor

2.2. Entrenando los modelos

En el apartado anterior nos encargamos de realizar el análisis y la transformación necesaria de los datos para poder aplicar correctamente los algoritmos de regresión que mencionamos en el marco teórico. Recordemos que el objetivo principal es predecir con el menor error posible el valor medio de las propiedades que figuran en el data set. Para ello es necesario realizar todos los pasos que se enumeran a continuación:

- Cargar el dataset.
- Dividir en Train y Test (en 80/20).
- Entrenar el modelo de Regresión con los datos de Train (Cross-Validation) con distintos hiperparámetros
- Fijar parámetrios, realizar `predict` sobre el Conjunto de Test.

Dividimos en dos, un set de training con un 80 % de los datos y un set de testing con el 20 % restante.

```
[ ]  # Dividir los datos en  test (20%) y train (80%).
     from sklearn.model_selection import train_test_split
     #col = ['sbp','tobacco','ldl','adiposity','famhist','obesity','alcohol','age']
     col = ['sbp','tobacco','ldl','adiposity','famhist','typea','obesity','alcohol','age']
     X_train, X_test, y_train, y_test = train_test_split(df[col], df['chd'], test_size=0.2, random_state=1234)
```

Antes de empezar con el entrenamiento vamos a describir el funcionamiento de una herramienta que nos provee Scikit-Learn llamado `GridSearchCV`, la cual nos facilita el fine-tune y la elección de los mejores hiperparámetros para el método de regression. Se le debe indicar con qué hiperparámetros se desea experimentar y con qué valores probar, y evaluará todas las posibles combinaciones de valores de hiperparámetros utilizando `cross-validation`. Por ejemplo, el siguiente código busca la mejor combinación de valores de hiperparámetros para `RandomForestRegressor`:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
param_grid = [
        {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
        {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
    ]
forest_reg = RandomForestRegressor()
grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
scoring='neg_mean_squared_error',
return_train_score=True)
grid_search.fit(housing_prepared, housing_labels)
```

`GridSearchCV` recibe el método de regresión y el vector de hiperparámetros entre otros. `Param_grid` le dice a `Scikit-Learn` que primero evalúe todas las combinaciones $3 \times 4 = 12$ valores de hiperparámetros especificados en `n_estimators` y `max_features` de la primer linea, y luego intente todas las combinaciones $2 \times 3 = 6$ de valores de hiperparámetros en la segunda linea, pero esta vez con el hiperparámetro `bootstrap` establecido en False en lugar de True (que es el valor predeterminado para este hiperparámetro). Con todo, la búsqueda de la cuadrícula explorará $12 + 6 = 18$ combinaciones de valores de hiperparámetros para `RandomForestRegressor` , y entrenará cada modelo cinco veces (ya que estamos utilizando validación cruzada de cinco veces). En otras palabras, en total, habrá $18 \times 5 = 90$ rondas de entrenamiento. Y nos retornará aquella combinación de hiperparámetros que minimicen una condición particular, en este caso `scoring=neg_mean_squared_error`. Ahora apliquemos esta herramienta con nuestros datos

3. Resultados

Random Forest.

```
[ ] from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
    from sklearn.preprocessing import StandardScaler
    from sklearn.metrics import accuracy_score, recall_score, classification_report, confusion_matrix
    from sklearn.ensemble import RandomForestClassifier
```

```
[ ] clf_rf = RandomForestClassifier(random_state=42)
    param_grid = {
        'n_estimators': [50, 100, 500, 1000],
        'criterion': ['gini', 'entropy'],
        'max_depth': [1, 3, 7, 20],
        'min_samples_split': [10, 40, 100],
        'min_samples_leaf': [10, 100],
    }
    gs_rf = GridSearchCV(clf_rf, param_grid, verbose=1, scoring='recall', cv=3)
    gs_rf.fit(X_train, y_train)
```

Hiperparámetros que minimizan el error en Random Forest.

```
Fitting 3 folds for each of 192 candidates, totalling 576 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 576 out of 576 | elapsed:  6.3min finished
Train Recall: 0.4333702473237357
Test Recall: 0.3939393939393939
Train Accuracy: 0.8130081300813008
Test Accuracy: 0.6989247311827957
{'criterion': 'gini', 'max_depth': 7, 'min_samples_leaf': 10, 'min_samples_split': 10, 'n_estimators': 500}
```

Resumen Random Forest

```
[ ] print('Recall')
    print(pd.Series(test_recalls).sort_values(ascending=False))
    print('')
    print('Accuracy')
    print(pd.Series(test_accuracies).sort_values(ascending=False))
```

```
Recall
Random Forest    0.393939
dtype: float64

Accuracy
Random Forest    0.698925
dtype: float64
```

4. Conclusiones

Observamos que al aplicar random forest regression al segmento de test de la base de datos de la South African heart disease obtuvimos una precisión de casi 0.7 al pronosticar la presencia de un infarto de miocardio en los individuos. Si bien en ámbitos médicos puede no resultar muy preciso (suele utilizarse más de 0,999), hay que recordar que esta precisión está íntimamente ligada a los datos de entrenamiento, con mejores datos quizás se podría obtener una mejor presición, además es posible que otros métodos de machine learning presente mejores resultados que el obtenido con random forest.

Referencias

[1] Aurélien Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow Concepts, Tools, and Techniques to Build Intelligent Systems. págs. 177–189.

[2] Trevor Hastie, Robert Tibshirani, Jerome Friedman, The Elements of Statistical Learning Data Mining, Inference, and Prediction. págs. 587–603.