PROYECTO FINAL MÁSTER
CLASIFICADOR DOCUMENTOS MÉDICOS HOPE
2020 - 2021

Ruben Vasallo Gonzalez

16 de septiembre de 2020

Índice general

Capítulo 1

1.1. Resumen

El proyecto nace de la necesidad de poder disponer de una manera sencilla e inmediata, artículos médicos catalogados según los síntomas de pacientes, pudiendo hacer un *ranking* de más o menos interés en función del *feedback* aportado por los profesionales sanitarios sobre artículos relacionados con esos *síntomas*.

1.2. Astract

TODO

1.3. Keywords

clasificador articulos medicos, PCA, KNN, Regresion Logistica, Random Forest, SVM

Capítulo 2

Introducción

2.1. Definición del proyecto

El proyecto que aquí se presenta nace de la necesidad por parte del *proyecto HOPE* de clasificar y recomendar resultados sobre estudios clínicos de confianza y que estén actualizados. En Internet existe muchísima información sobre medicina y salud y no siempre toda es de fiar.

El proyecto HOPE (que significa *Health Operations for Personalized Evidence* en ingles) nace de la necesidad de ayudar a los profesionales sanitarios a encontrar la información que necesitan de la manera más rápida y fácil posible. Existe infinidad de información medica en Internet de miles de proyectos de investigación medica y esto hace que, muchas veces sea complicado encontrar la información sobre ensayos médicos para tratar información. En el ámbito de la medicina el tiempo perdido puede costar vidas y es un precio demasiado elevado a pagar, tanto a nivel económico como emocional.

Actualmente existen bases de datos de confianza en donde los profesionales sanitarios y el publico en general puede buscar informes y ensayos sobre estudios clínicos desarrollados anteriormente, pero no siempre es fácil o rápido encontrar estos resultados.

El proyecto HOPE es un sistema basado en inteligencia artificial para identificar los datos claves de casos clínicos registrados en la Historia Clínica Electrónica, en base a los cuales realiza una búsqueda única por paciente para proporcionar al profesional sanitario recomendaciones de tratamientos, estudios de investigación, información para el paciente, todo en base a registros de fuentes científicas de información. En este proyecto, profesionales sanitarios de todo el mundo puede consultar en una base de datos informes médicos relacionados con los síntomas que puedan tener sus pacientes y ver que otros tratamientos han dado resultado. Todo y con eso, el sistema no siempre devuelve los artículos más relevantes o actualizados por lo que, no siempre la información consultada es útil.

En este ámbito, los profesionales sanitarios pueden valorar si la información recibida ha sido útil o no respecto a la búsqueda que han realizado, por lo que con ese *feedback*, se pretende mejorar el sistema actual complementándolo con un modelo clasificador capaz de ayudar al actual a entregar realmente los artículos útiles basándose en el *feedback* que los profesionales sanitarios dan al sistema.

2.2. Estado del arte

Recomendadores que existen actualmente:

Capítulo 3

Objetivos del Máster

3.1. Objetivo principal

OP - Poder recomendar al profesional sanitario cuales son los artículos más útiles que pueden ayudar en el tratamiento del paciente, en base a los síntomas que este tiene, pudiendo realizar un *ranking* de mas interés a menos.

3.2. Objetivos secundarios

Para poder cumplir con el objetivo principal OP1, desglosaremos los siguientes objetivos secundarios:

OS1 - Extraer la información de la base de datos y tratarla para quedarnos solo con la que consideramos valida.

OS2 - Hacer un análisis de componentes principales (estudio de que atributos son relevantes para alcanzar el objetivo).

OS3 - Enriquecer de los datos (*data augmentation*) prediciendo los resultados que no están indicados si son relevantes o no. Aproximación por Vecinos más próximos (*K-Nearest-Neighbor*).

OS4 - Predecir los resultados usando el algoritmo de aprendizaje supervisado para clasificación llamado Regresión logística '*Logistic regression*'.

OS5 - Predecir los resultados usando el algoritmo de aprendizaje supervisado para clasificación llamado Bosques Aleatorios '*Random Forests*'.

OS6 - Predecir los resultados usando el algoritmo de aprendizaje supervisado para clasificación llamado Máquinas de vector soporte '*Support Vector Machines*'.

Capítulo 4

Metodología

4.1. Reuniones con el cliente

Para poder comprender y abordar con éxito el objetivo principal se realizaron 4 reuniones en donde el cliente expuso el problema a abordar y el origen de los datos para poder realizar el estudio.

En estas reuniones se pudo observar que los datos facilitados por el usuario requerían de una limpieza y tratamiento para poder cumplir el objetivo principal, ya que muchas observaciones tenían información poco relevante que podía generar ruido.

Realizando un primer análisis visual, se detecto que los datos aportados por el cliente eran insuficientes para completar el OP1, ya que solo se disponía de la información respecto de si un articulo había sido útil o no, pero no se disponía de la información suficientemente detallada para saber si había sido muy útil o poco útil para poder llegar a realizar un *ranking*. El cliente nos comenta que en el momento actual no dispone de ese nivel de detalle y se acuerda con el que, se realizara una aproximación para indicar si un articulo es útil o no dejando para mas adelante la opción de poder realizar *rankins* si se consigue ese nivel de detalle por parte del cliente.

También se pudo comprobar que el cliente disponía de un volumen de observaciones bajo por lo que se planteo la posibilidad de, o intentar obtener más observaciones facilitadas por el cliente, o intentar enriquecer las observaciones actuales generando nuevos datos por aproximación a los reales.

Finalmente se decidió estudiar si era viable generar nuevos valores por aproximación, debido a que en el momento en que se trato el problema, el cliente no podía facilitar más datos. Si a lo largo del estudio, el cliente conseguía facilitar nuevas observaciones, estas serian añadidas al estudio para aproximar mejor la solución final.

4.2. Extracción de los datos

Para cumplir con el OP1 mostramos los pasos que hemos seguido para extraer y procesar los datos:

El Origen de los datos se encuentra en una Base de datos SQL distribuida en dos tablas, que pasamos a detallar a continuación:

En la primera tabla llamada *fed_hope_sugerencia*, encontraremos la sugerencia que dio el programa HOPE en base a los parámetros que introdujo el profesional sanitario, almacenado en el atributo pedido y la respuesta que dio el programa, almacenado en el atributo respuesta. Todos los datos son almacenados en formato documento json.

En la figura 4.1 mostramos los atributos de la tabla *fed_hope_sugerencia*.

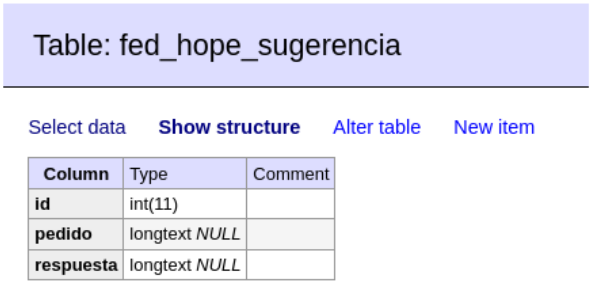


Table: fed_hope_sugerencia

Select data **Show structure** Alter table New item

Column	Type	Comment
id	int(11)	
pedido	longtext *NULL*	
respuesta	longtext *NULL*	

Figura 4.1: Visualización de los atributos de la tabla *fed_hope_sugerencia*

Si analizamos el **atributo pedido**, podemos observar, tal y como se muestra en la figura 4.2, varios atributos haciendo referencia a los síntomas que consulta el profesional sanitario.


```
{
  "data": {
    "type": "emr--em",
    "attributes": {
      "name": null,
      "affected_organ": "",
      "age": "75",
      "diagnostic_main": "FISTULA PERITONEAL",
      "gender": "male",
      "medical_history": "Paciente de 75 au00f1os con antecedentes de gastrectomu00eda total por adenocarcinoma gu00e1strico que intercurriu00f3 con
      eventraciu00f3n y posterior formaciu00f3n de fu00edstula entero-atmosfu00e9rica. Actualmente cursa postoperatorio de resecciu00f3n intestinal
      mu00e1s eventroplastu00eda con colocaciu00f3n de malla. Al examen impresiona en regular estado general, lu00facido, hemodinu00e1micamente estable
      , sin signos de falla de bomba. Regular entrada de aire, rales crepitantes bilaterales. Abdomen blando, depresible, levemente doloroso a la
      palpaciu00f3n profunda. Herida cubierta por apu00f3sitos estu00e9riles. Catarsis positiva. Diuresis positiva.\n\nPROBLEMAS ACTIVOS:\n- POP
      resecciu00f3n intestinal mu00e1s eventroplastu00eda: Paciente clu00ednicamente estable, hemodinu00e1micamente compensado. Refiere buena
      tolerancia al dolor. Afebril hace 72 hs. Cumple 4to du00eda de tratamiento con piperacilina tazobactam por neumonu00eda broncoaspirativa con
      aislamiento de E.coli BLEE. Hemocultivos vienen negativos. TAC de tu00f3rax y abdomen informa: Consolidaciu00f3n bibasal bilateral mu00e1s
      derrame pleural bilateral. Colecciu00f3n laminar posterior a ambos mu00fasculos rectos de 10x2x0.4 cm, y otra colecciu00f3n en herida
      quiru00fargica de pared de 10x2.8x1.2 cm. Se da aviso a cirujano tratante. Persiste con estado nauseoso, por lo que continua con antiemu00e9ticos
       reglados. \nEn aislamiento de contacto por germen multirrestente. \nSe da informe. Control evolutivo."
    }
  }
}
```


Figura 4.2: Muestra de una observación del atributo pedido

Si analizamos el **atributo respuesta**, podemos observar entre otros datos, el listado de artículos médicos sugeridos relacionados con los síntomas descritos por el profesional sanitario. Esta respuesta es muy amplia pero entre todos los atributos, podemos observar un listado de identificadores de artículos, con sus fechas de revisión de estos, y unas palabras claves descriptivas para esos artículos.

A continuación mostramos en la figura 4.3 una pequeña parte del contenido de una observación del atributo respuesta.

Figura 4.3: Ejemplo de contenido del atributo respuesta de una observación

En la segunda tabla llamada *fed_hope_sugerencia_feedback*, encontraremos, tal y como se muestra en la figura 4.4, la opinión *feedback* (que utilidad ha tenido la información por parte del profesional sanitario) de la información recibida dado un articulo en concreto en una búsqueda en concreto. Esta información se relaciona con la tabla *fed_hope_sugerencia* a través del atributo *fed_hope_sugerencia_id*.

En esta tabla, esta representada la opinión *feedback* del profesional sanitario en el atributo *utilidad*, que denota un valor 0 para los artículos que han sido poco útiles respecto a la búsqueda realizada y 1 para los artículos que si han sido útiles.

Table: fed_hope_sugerencia_feedback

Select data **Show structure** Alter table New item

Column	Type	Comment
id	int(11)	
articulo	varchar(255) *NULL*	
utilidad	int(11) *NULL*	
comentario	varchar(255) *NULL*	
fed_hope_sugerencia_id	int(11) *NULL*	

Figura 4.4: Visualización de los atributos de la tabla *fed_hope_sugerencia_feedback*

4.2.1. Lectura de los datos

Para extraer los datos de la base de datos nos ayudaremos de las librerías *sqlalchemy* y *pymysql* programadas en lenguaje *python* que nos permitirá acceder a la información almacenada en una base de datos *MySQL* y devolvérnosla en formato *dataframe*, un formato que nos permite entre otras cosas, realizar transformaciones de los datos para conseguir nuestro objetivo final.

Este formato es interpretable por la librería *pandas* y *numpy*, dos librerías programadas en lenguaje *python*, muy comunes en el ámbito de la ciencia del dato, que nos facilitara entre otras cosas, poder hacer operaciones matemáticas con los datos de manera eficiente. A continuación mostramos en la figura 4.5 el código utilizado para extraer los datos de la tabla *fed_hope_sugerencia*.

Import data from DB.

In [1]:
```
# pip install pymysql
from sqlalchemy import create_engine
import pymysql
import pandas as pd
import numpy as np
```

In [2]:
```
dbConnectionURL = 'mysql+pymysql://root:hope@mysql-master/hope'
dbConnection = create_engine(dbConnectionURL)

df = pd.read_sql('SELECT id, pedido, respuesta FROM fed_hope_sugerencia', con=dbConnection)
```

In [3]:
```
df.head(10)
```

Out[3]:

	id	pedido	respuesta
0	29	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"ef6d63fb-afe8...
1	30	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"0b8a1cc8-ce17...
2	31	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"25733e18-3245...
3	32	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"40320232-7510...
4	33	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"f686d89e-fc8e...
5	34	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"d94d7c78-9941...
6	35	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"0a14cc6b-af7b...
7	36	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"245bb87d-b52c...
8	37	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"fad05206-04f6...
9	38	{"data":{"type":"emr--em","attributes":{"name"...	{"data":{"type":"emr--emr","id":"a0f8dabe-a795...

Figura 4.5: Lectura de los datos

Aplicaremos los mismos pasos para leer la información del feedback de los profesionales sanitarios de la tabla *fed_hope_sugerencia_feedback*

4.2.2. Conversión a formato columnar

Para poder trabajar con los datos, necesitaremos que estos estén en formato columnar (tabla relacional) por lo que necesitaremos convertir los datos de estos *json* en tablas relacionales (A esta acción se le conoce como *flattening* o aplanar).

Este paso consiste en coger cada uno de los atributos que tiene el json y convertirlos en columnas de una tabla, añadiendo los valores. Si el *json* tiene varios niveles, este proceso añadirá tantas columnas como niveles tenga el *json*, siempre que todas las observaciones del *json* tengan el mismo formato. Este caso se nos cumple para las observaciones del atributo Pedido. No es así para las observaciones del atributo respuesta en el que tendremos que hacer un tratamiento especial que detallaremos posteriormente.

Cuando se analizan los datos recuperados, se detecta que estos, contienen caracteres que informan de los saltos de linea o tabulacion. Estos caracteres pueden ser mal interpretados a la hora de leer los datos de los documentos en formato *json* por lo que sera necesario eliminarlos.

- ***Flattening* del atributo pedido**

Para realizar la acción de *flattening* en el atributo pedido, nos ayudaremos de la funcionalidad *json_normalize* del paquete pandas que realiza esta acción. A continuación mostramos en la figura 4.6 el código utilizado para el atributo pedido.

Flattening JSON

In [5]:
```
import ast
import json
from pandas import read_json, json_normalize #package for flattening json in pandas df
#https://stackoverflow.com/questions/39899005/how-to-flatten-a-pandas-dataframe-with-some-columns-as-json

pd.options.display.max_columns = None
#pd.options.display.max_rows = None
```

In [6]:
```
# Flatterin column "Pedido"

pedidosData = json_normalize(df['pedido'].apply(json.loads).tolist()).add_prefix('pedido.')

pedidosData
```

Out[6]:

	pedido.data.type	pedido.data.attributes.name	pedido.data.attributes.affected_organ	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	pedido
0	emr--em	None		75	FISTULA PERITONEAL	
1	emr--em	None		31	REHABILITACION NEUROLOGICA	
2	emr--em	None		76	INSUFICIENCIA CARDIACA	
3	emr--em	None		75	FISTULA PERITONEAL	
4	emr--em	None		31	REHABILITACION NEUROLOGICA	
...	...	...	...	...	...	
119	emr--em	None		74	DIFICULTAD RESPIRATORIA	
120	emr--em	None		48	REHABILITACION NEUROLOGICA	
121	emr--em	None		40	REHABILITACION NEUROLOGICA	
122	emr--em	None		43	TEP	
123	emr--em	None		37	DOLOR ABDOMINAL	

124 rows × 7 columns

Figura 4.6: *Flattening* del atributo pedido.

- ***Flattening* del atributo respuesta**

Debido a que la información almacenada en el documento *json*, en el atributo respuesta es muy compleja (debido a que esta contiene diferentes documentos con diferentes niveles de información) como se puede apreciar en el apartado X, no podemos aplanar la información directamente como hemos hecho con el atributo pedido. Por lo que tenemos que analizar que información nos interesa recoger para enriquecer el conjunto de datos.

Después de analizar el documento, y ayudarnos del conocimiento del cliente, vemos que los atributos más interesantes son los que hacen referencia al identificador del articulo, las palabras claves asociadas al articulo por parte de la api pubmed y el mes y año de la revisión del articulo. Para recoger esta información nos crearemos una función que acceda directamente a estos atributos dado una observación. Después ejecutaremos esa función para cada observación ayudándonos de la función *apply*. A continuación mostramos este proceso en la figura 4.7.

In [7]:
```
# Flatterin column "respuesta"

def get_articles_from_respuesta(ld):
    jsonData = json.loads(ld)
    pubmedKeys = jsonData['data']['attributes']['pubmed_mt_opt']
    if pubmedKeys is None : pubmedKeys = []

    articles = list(jsonData['data']['attributes']['pubmed']['articles'])
    articlesIDs = []
    articlesRevisedYear = []
    articlesRevisedMonth = []
    for article in articles:
        articlesIDs.append(article['id'])
        articlesRevisedYear.append(article['revisedDate']['Year'])
        articlesRevisedMonth.append(article['revisedDate']['Month'])


    return dict({
        'articles': articlesIDs,
        'articlesRevisedYear': articlesRevisedYear,
        'articlesRevisedMonth': articlesRevisedMonth,
        'pubmed_keys': ','.join(pubmedKeys)
    })


respuestaData = json_normalize(df['respuesta'].apply(get_articles_from_respuesta).tolist()).add_prefix('respuest
a.')

respuestaData
```

Out[7]:

	respuesta.articles	respuesta.articlesRevisedYear	respuesta.articlesRevisedMonth	respuesta.pubmed_keys
0	[27395425, 28560554, 28641726, 26245344, 28942...	[2018, 2018, 2017, 2016, 2018, 2014, 2018, 201...	[01, 04, 12, 12, 06, 06, 09, 04, 01, 04]	Intestines,Therapeutics,Catharsis,Wounds and I...
1	[30210096, 27617939, 27210858, 26412482, 25487...	[2019, 2017, 2017, 2017, 2016, 2016, 2019, 201...	[03, 04, 08, 06, 06, 09, 02, 03, 01, 11]	Back,Wounds and Injuries,Catheterization,Rest,...
2	[21067951, 27616270, 27532500, 28426556, 27495...	[2011, 2017, 2017, 2019, 2017, 2009, 2017, 201...	[03, 07, 05, 01, 07, 03, 06, 05, 03, 03]	Heart Murmurs,Intestines,Lactic Acid,Therapeut...
3	[30179656, 28641726, 28694230, 27796647, 28867...	[2019, 2017, 2018, 2017, 2017, 2017, 2015, 201...	[03, 12, 05, 02, 11, 05, 08, 06, 01, 08]	Intestines,Therapeutics,Catharsis,Lower Extrem...
4	[29787536, 24840763, 28273653, 26836795, 26409...	[2019, 2014, 2017, 2016, 2016, 2019, 2013, 201...	[05, 10, 11, 11, 05, 04, 03, 09, 02, 12]	Abdomen,Catheterization,Headache,Diuresis,Extr...
...	...	...	...	...
119	[28641726, 30179656, 28694230, 27796647, 28867...	[2017, 2019, 2018, 2017, 2017, 2017, 2017, 201...	[12, 03, 05, 02, 11, 05, 09, 09, 01, 03]	Extremities,Catharsis,Tazobactam,Abdomen,Oxyge...
120	[27128826, 30336861, 30226191, 29371130, 29587...	[2017, 2019, 2019, 2018, 2018, 2019, 2019, 201...	[04, 01, 09, 10, 08, 01, 06, 08, 11, 04]	Catharsis,Abdomen,Lung
121	[30595510, 21554494, 26465238, 26875969, 30056...	[2019, 2012, 2016, 2016, 2019, 2019, 2016, 201...	[03, 04, 09, 08, 07, 10, 10, 05, 10, 06]	Abdomen,Wounds and Injuries,Lung,Stroke,Aphasi...
122	[30081165, 30629460, 26220984, 25749853, 28545...	[2018, 2019, 2016, 2016, 2018, 2020, 2015, 201...	[12, 03, 06, 04, 12, 02, 02, 09, 05, 04]	Extremities,Catharsis,Thromboembolism,Foramen ...
123	[30662053, 29879068, 26849395, 31061178, 31223...	[2019, 2018, 2016, 2019, 2019, 2019, 2018, 201...	[02, 06, 12, 12, 07, 03, 12, 05, 04, 04]	Fever,Catharsis,Infections,Abdominal Pain,Abdo...

124 rows × 4 columns

Figura 4.7: *Flattening* del atributo respuesta.

Una vez aplanado los dos atributos, los uniremos en un único conjunto de datos, junto a los datos originales de la tabla *fed_hope_sugerencia* para poder trabajar con ellos. Estos es importante para mantener el id de cada observación, de cara a poder luego identificar el feedback de los profesionales sanitarios con cada observación.

4.3. Procesado de los datos

4.3.1. Análisis de los datos

Una vez tenemos los datos en formato tabular, observamos que existen ciertos atributos que contienen listas de opciones como son los atributos *pubmed_keys* (que corresponde a las palabras clave que la api de pubmed nos devuelve para esta observación), *articles* (que corresponde a los ids de los artículos relacionados con esa observación), *articlesRevisedYear* i *articlesRevisedMonth* (que corresponde a los años y meses de los artículos según están ordenados en el atributo *articles*)

Como nuestro OP1 es poder recomendar artículos útiles, necesitamos tener una observación por articulo, para poder posteriormente analizar de manera independiente si ese articulo fue útil o no para la observación a la que hace referencia.

Por lo que necesitaremos expandir (duplicar) cada observación con solo un articulo que haga referencia a el. A continuación mostramos en la figura 4.8 el código para expandir el atributo *articles* (el resto de atributos su proceso seria similar).

Expand Articles

```
In [10]: dfCleaned = dfCleaned.explode('respuesta.articles').reset_index(drop=True)

         # Expand Articles Year
         dfCleanedArticlesRevisedYear = dfCleaned.explode('respuesta.articlesRevisedYear').reset_index(drop=True)
         dfCleaned['respuesta.articlesRevisedYear'] = dfCleanedArticlesRevisedYear['respuesta.articlesRevisedYear']

         # Expand Articles Month
         dfCleanedArticlesRevisedMonth = dfCleaned.explode('respuesta.articlesRevisedMonth').reset_index(drop=True)
         dfCleaned['respuesta.articlesRevisedMonth'] = dfCleanedArticlesRevisedMonth['respuesta.articlesRevisedMonth']


         dfCleaned
```

Out[10]:

nostic_main	pedido.data.attributes.gender	pedido.data.attributes.medical_history	respuesta.articles	respuesta.articlesRevisedYear	respuesta.articlesRevisedMonth
ERITONEAL	male	Paciente de 75 au00f1os con antecedentes de ga...	27395425	2018	01
ERITONEAL	male	Paciente de 75 au00f1os con antecedentes de ga...	28560554	2018	04
ERITONEAL	male	Paciente de 75 au00f1os con antecedentes de ga...	28641726	2017	12
ERITONEAL	male	Paciente de 75 au00f1os con antecedentes de ga...	26245344	2016	12
ERITONEAL	male	Paciente de 75 au00f1os con antecedentes de ga...	28942543	2018	06
...	...	...	...	...	...
ABDOMINAL	male	aciente de 36 au00f1os con antecedentes de ami...	29975804	2016	01
ABDOMINAL	male	aciente de 36 au00f1os con antecedentes de ami...	26362243	2019	04
ABDOMINAL	male	aciente de 36 au00f1os con antecedentes de ami...	30711130	2015	10
ABDOMINAL	male	aciente de 36 au00f1os con antecedentes de ami...	27932159	2019	05
ABDOMINAL	male	aciente de 36 au00f1os con antecedentes de ami...	19104113	2015	02

Figura 4.8: Expansión del atributo *articles*.

Después de tener un articulo por observación, observamos que tenemos atributos poco relevantes (como el atributo *data.type* o *Name* que contiene siempre el mismo valor) o que no contienen información alguna (como

el atributo *affected_organ*) como se puede observar en la figura 4.9. Eliminaremos estos atributos junto a otros con la misma casuistica, para no generar ruido en el posterior análisis predictivo.

Out[11]:

	id	pedido.data.type	pedido.data.attributes.name	pedido.data.attributes.affected_organ	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	p
0	29	emr--em	None		75	FISTULA PERITONEAL	
1	29	emr--em	None		75	FISTULA PERITONEAL	
2	29	emr--em	None		75	FISTULA PERITONEAL	
3	29	emr--em	None		75	FISTULA PERITONEAL	
4	29	emr--em	None		75	FISTULA PERITONEAL	
...	...	...	...	...	...	...	
1235	152	emr--em	None		37	DOLOR ABDOMINAL	
1236	152	emr--em	None		37	DOLOR ABDOMINAL	
1237	152	emr--em	None		37	DOLOR ABDOMINAL	
1238	152	emr--em	None		37	DOLOR ABDOMINAL	
1239	152	emr--em	None		37	DOLOR ABDOMINAL	

1240 rows × 13 columns

Figura 4.9: Se detectan algunos atributos con poca o nula relevancia.

Y con estos pasos hemos cubierto el OS1

4.3.2. Análisis de componentes principales

El análisis de componentes principales (o como se le conoce en ingles por '*Principal Component Analysis*' o *PCA*), es un componente fundamental en el análisis de los datos, ya que permite reducir el número de atributos de un conjunto de datos para eliminar el ruido que los posteriores análisis/modelos predictivos funcionen con mejor precision.

Por poner un ejemplo sencillo, imaginemos que tenemos un conjunto de datos de modelos de coche con muchísimos atributos de estos, como por ejemplo, el nombre del modelo, el color, el número de puertas, la cilindrada y la potencia, entre otros. Probablemente si intentáramos analizar los datos en busca de cual es el modelos que menos consume y quisiéramos crear un modelo predictivo con este objetivo, podríamos observar a simple vista que tenemos atributos que no nos son necesarios y que generaría distracción (ruido) a la hora de conseguir nuestro objetivo, como es el caso del atributo color, o el nombre del modelo.

El *PCA* nos ayudara a encontrar cuales son los atributos más significativos del conjunto de datos para conseguir predecir el atributo que queremos, que en el caso que nos toca, es el atributo '*utilidad*'.

Es cierto que, en nuestro conjunto de datos, no tenemos un gran volumen de atributos, pero este proceso nos puede ayudar a eliminar atributos con poca relevancia, para así, poder simplificar el modelo predictivo final.

Para realizar el *PCA* nos ayudaremos de la librería *sklearn.decomposition* que ya nos ofrece implementada la lógica para ejecutarlo. Pero antes, tenemos que transformar todos los atributos de Categóricos (texto) a Continuos (números continuos). Este paso se realiza para que el *PCA* pueda realizar operaciones matemáticas sobre los valores de las observaciones. A este proceso se le conoce como factorización (*factorize*). Realizar esta transformación es tan sencillo como coger cada uno de los valores del atributo y asignarles un número. A continuación mostramos en la figura 4.10 un ejemplo de transformación de atributo categórico a continuo.

```
In [5]: dataDiagnosticMain, categoriesDiagnosticMain = pd.factorize(dfPCA['pedido.data.attributes.diagnostic_main'])

        categoriesDiagnosticMain

Out[5]: Index(['FISTULA PERITONEAL', 'INSUFICIENCIA RESPIRATORIA', 'POLITRAUMATISMO',
               'ABDOMEN AGUDO', 'TORACOTOMIA', 'INFECCION DE PARTES BLANDAS',
               'DOLOR ABDOMINAL', 'INFECCION URINARIA', 'HEMORRAGIA DIGESTIVA',
               'ACV.ISQUEMICO', 'DISNEA', 'CETOACIDOSIS DIABETICA'],
              dtype='object')

        0 => first element found => 'FISTULA PERITONEAL'

        1 => second element found => 'INSUFICIENCIA RESPIRATORIA'

        ...

In [6]: dfPCA['pedido.data.attributes.diagnostic_main'] = dataDiagnosticMain

        dfPCA.head(10)
```

Out[6]:

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	pedido.data.attributes.gender	respuesta.pubmed_keys	articulo	uti
0	75	0	male	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu...	27395425	
1	75	0	male	Abdomen,Blood Culture,Catharsis,Diuresis,Drug ...	28694230	
2	36	1	male	Abdomen,Analgesics,Antitubercular Agents,Cipro...	28805236	
3	51	2	male	Abdomen,Analgesics,Bone,Catharsis,Electroconvu...	27537587	
4	51	2	male	Abdomen,Analgesics,Bone,Catharsis,Electroconvu...	28148670	
5	18	3	male	Abdomen,Anti-Bacterial Agents,Diuresis,Operati...	25055513	
6	18	3	male	Abdomen,Anti-Bacterial Agents,Diuresis,Operati...	29279563	
7	18	3	male	Abdomen,Anti-Bacterial Agents,Diuresis,Operati...	29279563	
8	18	3	male	Abdomen,Anti-Bacterial Agents,Diuresis,Operati...	28065368	
9	76	4	male	Abdomen,Amiodarone,Analgesia,Angiodysplasia,Hy...	30762794	

Figura 4.10: Factorización de un atributo.

Una vez realizada la factorización de todos los atributos necesarios, el siguiente paso, antes de ejecutar el PCA, sera estandarizar todos los valores de las observaciones a un rango de entre 1 y -1. Esto se realiza para igualar la importancia de todos los atributos, ya que en el paso anterior, al factorizar los valores de los atributos, se nos puede dar el caso de tener valores muy altos (por ejemplo, al factorizar un atributo con 100 valores diferentes, se nos dará casos de observaciones que en un atributo tienen el valor 100, que puede ser más alto que otros valores que no se han transformado). Al realizar el PCA, si no se hace esta estandarización de los datos, los valores más altos se les dará más peso, pero no por eso pueden ser relevantes. Por lo que es imperativo el realizar esta estandarización.

Para realizar esta estandarización nos ayudaremos de la librería *sklearn.preprocessing* que ya tiene implementado el modelo de estandarización llamado *StandardScaler*. A continuación mostramos en la figura 4.11 un ejemplo de estandarización del conjunto de datos excluyendo de este el atributo a predecir (utilidad)

```
featuresTransformed = StandardScaler().fit_transform(x)

featuresTransformed
```

```
array([[ 0.91709628, -2.24479066,  0.        , -1.81819247, -0.19317962],
       [ 0.91709628, -2.24479066,  0.        , -1.65608091,  0.31397864],
       [-0.8747549 , -1.85670821,  0.        , -1.49396934,  0.35732434],
       [-0.18558137, -1.46862576,  0.        , -1.33185777, -0.13766811],
       [-0.18558137, -1.46862576,  0.        , -1.33185777,  0.100948  ],
       [-1.70176313, -1.0805433 ,  0.        , -1.16974621, -1.10687007],
       [-1.70176313, -1.0805433 ,  0.        , -1.16974621,  0.54253988],
       [-1.70176313, -1.0805433 ,  0.        , -1.16974621,  0.54253988],
       [-1.70176313, -1.0805433 ,  0.        , -1.16974621,  0.06842018],
       [ 0.96304118, -0.69246085,  0.        , -1.00763464,  1.12171293],
       [ 0.96304118, -0.69246085,  0.        , -1.00763464,  1.12171293],
       [ 0.96304118, -0.69246085,  0.        , -1.00763464,  0.66295943],
       [ 0.96304118, -0.69246085,  0.        , -1.00763464, -1.74384989],
       [-1.19636921, -0.30437839,  0.        , -0.84552307,  0.41640915],
       [-1.19636921, -0.30437839,  0.        , -0.84552307,  0.32427757],
       [ 1.10087588,  0.08370406,  0.        , -0.68341151, -0.59661724],
```

Figura 4.11: Estandarización del conjunto de datos.

Una vez realizado estos pasos podemos ejecutar el *PCA*. Con el conjunto de datos completo, y tal y como podemos ver en la figura 4.12 , el *PCA* nos muestra que con solo 3 atributos, el modelo es capaz de explicar (predecir) el 95 % de las observaciones.

```
print('explained variance ratio (first three components): %s' %
str(pca.explained_variance_ratio_))
print('sum of explained variance (first three components): %s' %
str(sum(pca.explained_variance_ratio_)))
```

```
explained variance ratio (first three components): [0.44726263 0.25669502 0.25009763]
sum of explained variance (first three components): 0.9540552760689917
```

Figura 4.12: Resultado del *PCA*.

Si extrapolamos los resultados de las observaciones a un gráfico que mostramos en la figura 4.13 dibujando los 3 componentes principales como si fueran 3 dimensiones, y pintamos los valores a predecir sobre estos (aplicando el color rojo a las observaciones que no consideran útiles, y azul a los que si se consideran útiles), podemos observar que la gran mayoría de los rojo se encuentran en el plano del componente principal 1 y dos, mientras que el azul esta mas repartidos en los dos planos.

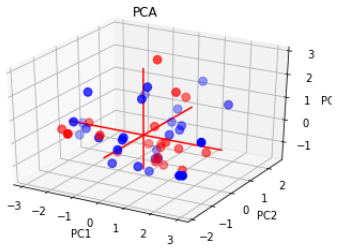


Figura 4.13: Proyección delos resultados del *PCA* en una gráfica.

Una vez ejecutado el modelo, podemos pedirle a este que nos indique cuales son los 3 componentes que ha detectado que son los principales. Para hacer esto le pediremos al modelo que nos muestre los resultados de los 3 componentes sobre los atributos del conjunto de datos. Estos resultados se pueden ver en la figura 4.14. Para saber cuales son los componentes, tendremos que quedarnos para cada componente principal, el valor del atributo que más se acerque a 1.

```
pd.DataFrame(pca.components_,columns=features,index = ['PC1','PC2','PC3'])
```

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	pedido.data.attributes.gender	respuesta.pubmed_keys	articulo
PC1	-0.050066	0.705007	-1.110223e-16	0.705462	0.052744
PC2	0.760475	0.071805	-1.318390e-16	0.030411	-0.644668
PC3	-0.630542	-0.130930	-1.110223e-16	0.142297	-0.751682

Figura 4.14: Proyección de los resultados del *PCA* sobre los atributos del conjunto de datos.

Para este caso, el *PCA* nos muestra que los 3 componentes principales son, el atributo *pubmed_keys*, el atributo *age* y el atributo *diagnostic_main*.

Después de comentar los resultados con el cliente, acordamos hacer dos pruebas más:

• **Ejecutar el *PCA* solo teniendo en cuenta las observaciones que tienen informado el atributo utilidad, y añadiendo el mes y año del articulo.**

En este caso, el *PCA* nos muestra tal y como podemos ver en la figura 4.15 , que necesitamos 5 atributos para que el modelo sea capaz de explicar (predecir) el 97 % de las observaciones.

```
print('explained variance ratio (first three components): %s' %
str(pca.explained_variance_ratio_))
print('sum of explained variance (first three components): %s' %
str(sum(pca.explained_variance_ratio_)))
```

```
explained variance ratio (first three components): [0.32833981 0.23094825 0.17265137 0.14192254 0.09557452]
sum of explained variance (first three components): 0.9694364891147347
```

Figura 4.15: Resultado del *PCA* de la segunda ejecución.

Para este caso, el *PCA* nos muestra como podemos ver en la figura 4.16 , que los 5 componentes principales son, el atributo *diagnostic_main*, el atributo *Year*, el atributo *pubmed_keys*, el atributo *age* y el atributo *articulo*.

```
pd.DataFrame(pca.components_,columns=features,index = ['PC1','PC2','PC3','PC4','PC5'])
```

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	pedido.data.attributes.gender	respuesta.articlesRevisedYear	respuesta.articlesRevisedMont
PC1	-0.094054	0.603912	-1.110223e-16	-0.365651	0.33837
PC2	0.135318	0.327845	-5.273559e-16	0.362710	-0.55606
PC3	-0.891259	-0.174252	-5.828671e-16	-0.236812	-0.13360
PC4	0.382620	-0.115174	-1.929013e-15	-0.621075	0.16245
PC5	0.098479	0.051453	3.774758e-15	-0.540677	-0.72925

PC1 => diagnostic_main PC2 => pubmed_keys/Year PC3 => pubmed_keys PC4 => age PC5 => articulo

Figura 4.16: Proyección de los resultados del *PCA* sobre los atributos del conjunto de datos para la segunda ejecución.

• Ejecutar el *PCA* solo teniendo en cuenta las observaciones que tienen informado el atributo utilidad, añadiendo el mes y año del articulo, eliminando los atributos *articulo* y *gender* y expandiendo el atributo *pubmed_keys* a un valor por observación, en vez de tener todas las palabras clave de cada articulo en una única observación.

En este caso, el *PCA* nos muestra tal y como podemos ver en la figura 4.17 , que necesitamos 4 atributos para que el modelo sea capaz de explicar (predecir) el 90 % de las observaciones.

```
print('explained variance ratio (first three components): %s' %
str(pca.explained_variance_ratio_))
print('sum of explained variance (first three components): %s' %
str(sum(pca.explained_variance_ratio_)))
```

```
explained variance ratio (first three components): [0.31205207 0.24763151 0.19472659 0.14857077]
sum of explained variance (first three components): 0.9029809474497172
```

Figura 4.17: Resultado del *PCA* de la tercera ejecución.

Para este caso, el *PCA* nos muestra como podemos ver en la figura 4.18 , que los 4 componentes principales son, el atributo *diagnostic_main*, el atributo *Month*, el atributo *pubmed_keys* y el atributo *age*.

```
pd.DataFrame(pca.components_,columns=features,index = ['PC1','PC2','PC3','PC4'])
```

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	respuesta.articlesRevisedYear	respuesta.articlesRevisedMonth	respuesta.pubmed_keys
PC1	0.202186	0.688327	-0.236152	-0.080358	0.650462
PC2	-0.299028	-0.000864	-0.648047	0.698266	-0.055149
PC3	-0.908866	-0.006936	0.153411	-0.221720	0.318152
PC4	-0.015928	-0.145721	-0.702451	-0.673071	-0.179024

PC1 => diagnostic_main / pubmed_keys PC2 => articlesRevisedMonth PC3 => pubmed_keys PC4 => age

Figura 4.18: Proyección de los resultados del *PCA* sobre los atributos del conjunto de datos para la tercera ejecución.

4.4. Enriquecimiento de los datos. Aproximación por Vecinos más próximos (K-NN)

TODO

4.5. Modelos Predictivos

4.5.1. Regresión logística *'Logistic regression'*

TODO

4.5.2. Bosques Aleatorios *'Random Forest'*

TODO

4.5.3. Maquinas de Vector Soporte *'Support Vector Machines'*

TODO

4.6. Resultados

TODO

Capítulo 5

Conclusiones

TODO

Capítulo 6

Bibliográfica

Índice de figuras

Capítulo 7

Anexos

7.1. Random Forest

S4 - HOPE random forest

September 16, 2020

1 Import data from DB.

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
dfOrg = pd.read_csv('hope_dataset_cleaned.csv')

print(dfOrg.shape[0])
```

```
1243
```

[3]:
```
dfOrg.head(10)
```

[3]:

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	\
0	75.0	FISTULA PERITONEAL	
1	75.0	FISTULA PERITONEAL	
2	75.0	FISTULA PERITONEAL	
3	75.0	FISTULA PERITONEAL	
4	75.0	FISTULA PERITONEAL	
5	75.0	FISTULA PERITONEAL	
6	75.0	FISTULA PERITONEAL	
7	75.0	FISTULA PERITONEAL	
8	75.0	FISTULA PERITONEAL	
9	75.0	FISTULA PERITONEAL	

	pedido.data.attributes.gender	articulo	respuesta.articlesRevisedYear	\
0	male	27395425	2018	
1	male	28560554	2018	
2	male	28641726	2017	
3	male	26245344	2016	
4	male	28942543	2018	
5	male	24782153	2014	
6	male	28002229	2018	
7	male	27505109	2017	
8	male	24850546	2015	
9	male	29371050	2019	

	respuesta.articlesRevisedMonth	\
0	1	
1	4	
2	12	
3	12	
4	6	
5	6	
6	9	
7	4	
8	1	
9	4	

	respuesta.pubmed_keys	utilidad
0	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	1.0
1	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
2	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
3	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
4	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
5	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
6	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
7	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
8	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN
9	Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…	NaN

2 Transform (factorice) from Categories to continuous atributes

Transform 'pedido.data.attributes.diagnostic_main' atribute

[4]:
```
dataDiagnosticMain, categoriesDiagnosticMain = pd.factorize(dfOrg['pedido.data.
↪attributes.diagnostic_main'])

dfOrg['pedido.data.attributes.diagnostic_main'] = dataDiagnosticMain
```

Transform 'gender' atribute

[5]:
```
dataGender, categoriesGender = pd.factorize(dfOrg['pedido.data.attributes.
↪gender'])

dfOrg['pedido.data.attributes.gender'] = dataGender
```

Transform 'respuesta.pubmed_keys' atribute

[6]:
```
categoriesORGPubMedKeys = dfOrg['respuesta.pubmed_keys'].value_counts()

print("total: " + str(categoriesORGPubMedKeys.size))
```



```
total: 80
```

[7]:
```
dataPubMedKeys, categoriesPubMedKeys = pd.factorize(dfOrg['respuesta.
↪pubmed_keys'])

dfOrg['respuesta.pubmed_keys'] = dataPubMedKeys
```

[8]:
```
dfOrg.head(10)
```

[8]:

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	\
0	75.0	0	
1	75.0	0	
2	75.0	0	
3	75.0	0	
4	75.0	0	
5	75.0	0	
6	75.0	0	
7	75.0	0	
8	75.0	0	
9	75.0	0	

	pedido.data.attributes.gender	articulo	respuesta.articlesRevisedYear	\
0	0	27395425	2018	
1	0	28560554	2018	
2	0	28641726	2017	
3	0	26245344	2016	
4	0	28942543	2018	
5	0	24782153	2014	
6	0	28002229	2018	
7	0	27505109	2017	
8	0	24850546	2015	
9	0	29371050	2019	

	respuesta.articlesRevisedMonth	respuesta.pubmed_keys	utilidad
0	1	0	1.0
1	4	0	NaN
2	12	0	NaN
3	12	0	NaN
4	6	0	NaN
5	6	0	NaN
6	9	0	NaN
7	4	0	NaN
8	1	0	NaN
9	4	0	NaN

[9]:
```
print("age NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.attributes.age'])].
↪shape[0]))
```



```
print("diagnostic_main NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.
↪attributes.diagnostic_main'])].shape[0]))
print("gender NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.attributes.
↪gender'])].shape[0]))
print("articulo NaN => " + str(dfOrg[pd.isnull(dfOrg['articulo'])].shape[0]))
print("articlesRevisedYear NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
↪articlesRevisedYear'])].shape[0]))
print("articlesRevisedMonth NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
↪articlesRevisedMonth'])].shape[0]))
print("pubmed_keys NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
↪pubmed_keys'])].shape[0]))
print("utilidad NaN => " + str(dfOrg[pd.isnull(dfOrg['utilidad'])].shape[0]))
```

```
age NaN => 10
diagnostic_main NaN => 0
gender NaN => 0
articulo NaN => 0
articlesRevisedYear NaN => 0
articlesRevisedMonth NaN => 0
pubmed_keys NaN => 0
utilidad NaN => 1192
```

Remove row with age eq NaN

[10]:
```
dfOrg = dfOrg[pd.notnull(dfOrg['pedido.data.attributes.age'])]
```

3 Standardize the Data

Choosed "age", "diagnostic_main", "year", "pubmed_keys" and "articulo" attributes (based on PCA_V2 study)

[11]:
```
from sklearn.preprocessing import StandardScaler

features = ["pedido.data.attributes.age",
    "pedido.data.attributes.diagnostic_main",
    "respuesta.articlesRevisedYear",
    "respuesta.pubmed_keys",
    "articulo"]

# Separating out the features
x = dfOrg.loc[:, features].values

featuresTransformed = StandardScaler().fit_transform(x)

dfStandarized = pd.DataFrame(featuresTransformed, index=dfOrg.index,␣
↪columns=features)
```

```
dfStandarized['utilidad'] = dfOrg['utilidad']

dfStandarized
```

[11]:
```
      pedido.data.attributes.age  pedido.data.attributes.diagnostic_main  \
0                       1.443474                               -1.360638
1                       1.443474                               -1.360638
2                       1.443474                               -1.360638
3                       1.443474                               -1.360638
4                       1.443474                               -1.360638
…                            …                                       …
1238                   -0.429381                               -0.580827
1239                   -0.429381                               -0.580827
1240                   -0.429381                               -0.580827
1241                   -0.429381                               -0.580827
1242                   -0.429381                               -0.580827

      respuesta.articlesRevisedYear  respuesta.pubmed_keys  articulo  utilidad
0                          0.643671              -1.650220 -0.221939       1.0
1                          0.643671              -1.650220  0.137839       NaN
2                          0.224418              -1.650220  0.162904       NaN
3                         -0.194835              -1.650220 -0.577070       NaN
4                          0.643671              -1.650220  0.255793       NaN
…                               …                      …         …         …
1238                      -0.194835               1.520816  0.574852       NaN
1239                       1.062924               1.520816 -0.540973       NaN
1240                      -0.614089               1.520816  0.801912       NaN
1241                       1.062924               1.520816 -0.056202       NaN
1242                      -0.614089               1.520816 -2.782199       NaN

[1233 rows x 6 columns]
```

4 Separe data by utilidad is defined

[12]:
```
dfDataSetComplete = dfStandarized[pd.notnull(dfStandarized['utilidad'])]

print(dfDataSetComplete.shape[0])

dfDataSetToPredict = dfStandarized[pd.isnull(dfStandarized['utilidad'])]

print(dfDataSetToPredict.shape[0])
```

```
51
1182
```


5 Random Forest

We check the number of results

[13]:
```
dfDataSetComplete.groupby('utilidad').size()
```

[13]:
```
utilidad
0.0    21
1.0    30
dtype: int64
```

Separe “utilidad” atribute from dataToTrain

[14]:
```
X = np.array(dfDataSetComplete.drop(['utilidad'],1))
y = np.array(dfDataSetComplete['utilidad'])
X.shape
```

[14]:
```
(51, 5)
```

[15]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

6 Exploring number of estimators

Via the sample size n of the bootstrap sample, we control the bias-variance tradeoff of the random forest. By choosing a larger value for n, we decrease the randomness and thus the forest is more likely to overfit. On the other hand, we can reduce the degree of overfitting by choosing smaller values for n at the expense of the model performance. In most implementations, including the RandomForestClassifier implementation in scikit-learn, the sample size of the bootstrap sample is chosen to be equal to the number of samples in the original training set, which usually provides a good bias-variance tradeoff.

https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb

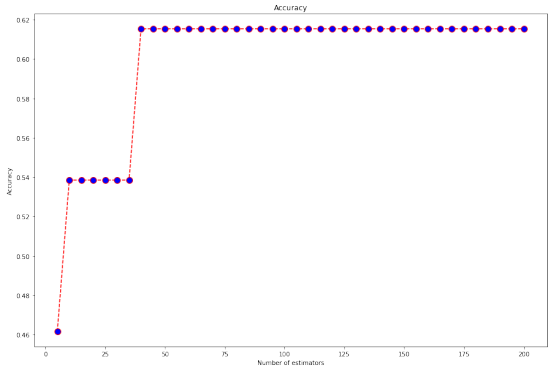
[16]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score

k_range = range(5, 205, 5)
accuracy = []

for k in k_range:
    forest_test = RandomForestClassifier(
        criterion='entropy',
        n_estimators=k,
        random_state=0
```



```
    )
    forest_test.fit(X_train, y_train)
    y_pred_test = forest_test.predict(X_test)

    accuracy.append(accuracy_score(y_test, y_pred_test))
```

[17]:
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(figsize=(15, 10))
axs.plot(k_range, accuracy, color='red', linestyle='dashed', marker='o',
        markerfacecolor='blue', markersize=10)
axs.set_title('Accuracy')
axs.set_xlabel('Number of estimators')
axs.set_ylabel('Accuracy')
```

[17]:
```
Text(0, 0.5, 'Accuracy')
```



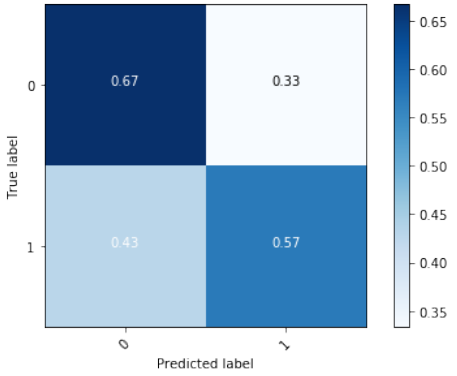
6.1 Evaluating the Algorithm

[18]:
```
forest = RandomForestClassifier(
    criterion='entropy',
    n_estimators=40,
    random_state=0
)

forest.fit(X_train, y_train)

y_pred = forest.predict(X_test)

print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.57      0.67      0.62         6
         1.0       0.67      0.57      0.62         7

    accuracy                           0.62        13
   macro avg       0.62      0.62      0.62        13
weighted avg       0.62      0.62      0.62        13

0.6153846153846154
```

[19]:
```
import itertools


cnf_matrix = confusion_matrix(y_test, y_pred)

def plot_confusion_matrix(cm, classes):
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cmap=plt.cm.Blues

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], ".2f"),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
```

```
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

n_classes=["0","1"]
plot_confusion_matrix(cnf_matrix, classes=n_classes)
```



7 Run Prediction

[20]:
```
result = forest.predict(dfDataSetToPredict[[
    "pedido.data.attributes.age",
    "pedido.data.attributes.diagnostic_main",
    "respuesta.articlesRevisedYear",
    "respuesta.pubmed_keys",
    "articulo"
]])

result
```


[20]:
```
array([1., 1., 1., …, 1., 0., 0.])
```

https://chrisalbon.com/machine_learning/trees_and_forests/random_forest_classifier_example/
https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html
https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/

7.2. Random Forest V2

S4 - HOPE random forest-V2

September 16, 2020

1 Import data from DB.

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
dfOrg = pd.read_csv('hope_dataset_cleaned.csv')

print(dfOrg.shape[0])
```

```
1243
```

[3]:
```
dfOrg.head(10)
```

[3]:
```
   pedido.data.attributes.age pedido.data.attributes.diagnostic_main  \
0                        75.0                     FISTULA PERITONEAL
1                        75.0                     FISTULA PERITONEAL
2                        75.0                     FISTULA PERITONEAL
3                        75.0                     FISTULA PERITONEAL
4                        75.0                     FISTULA PERITONEAL
5                        75.0                     FISTULA PERITONEAL
6                        75.0                     FISTULA PERITONEAL
7                        75.0                     FISTULA PERITONEAL
8                        75.0                     FISTULA PERITONEAL
9                        75.0                     FISTULA PERITONEAL

  pedido.data.attributes.gender  articulo  respuesta.articlesRevisedYear  \
0                          male  27395425                           2018
1                          male  28560554                           2018
2                          male  28641726                           2017
3                          male  26245344                           2016
4                          male  28942543                           2018
5                          male  24782153                           2014
6                          male  28002229                           2018
7                          male  27505109                           2017
8                          male  24850546                           2015
9                          male  29371050                           2019
```



```
1                          male  27395425                           2018
2                          male  27395425                           2018
3                          male  27395425                           2018
4                          male  27395425                           2018
5                          male  27395425                           2018
6                          male  27395425                           2018
7                          male  27395425                           2018
8                          male  27395425                           2018
9                          male  27395425                           2018

   respuesta.articlesRevisedMonth respuesta.pubmed_keys  utilidad
0                               1              Abdomen       1.0
1                               1       Adenocarcinoma       1.0
2                               1          Antiemetics       1.0
3                               1        Blood Culture       1.0
4                               1             Catharsis       1.0
5                               1              Diuresis       1.0
6                               1               Fistula       1.0
7                               1          Gastrectomy       1.0
8                               1    Incisional Hernia       1.0
9                               1            Intestines       1.0
```

2 Transform (factorice) from Categories to continuous atributes

Transform 'pedido.data.attributes.diagnostic_main' atribute

[5]:
```
dataDiagnosticMain, categoriesDiagnosticMain = pd.factorize(dfOrg['pedido.data.
↪attributes.diagnostic_main'])

dfOrg['pedido.data.attributes.diagnostic_main'] = dataDiagnosticMain
```

Transform 'gender' atribute

[6]:
```
dataGender, categoriesGender = pd.factorize(dfOrg['pedido.data.attributes.
↪gender'])

dfOrg['pedido.data.attributes.gender'] = dataGender
```

Transform 'respuesta.pubmed_keys' atribute

[7]:
```
categoriesORGPubMedKeys = dfOrg['respuesta.pubmed_keys'].value_counts()

print("total: " + str(categoriesORGPubMedKeys.size))
```

```
total: 353
```



```
   respuesta.articlesRevisedMonth  \
0                               1
1                               4
2                              12
3                              12
4                               6
5                               6
6                               9
7                               4
8                               1
9                               4

                                respuesta.pubmed_keys  utilidad
0  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       1.0
1  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
2  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
3  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
4  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
5  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
6  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
7  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
8  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
9  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
```

Expand pubmed_keys attribute

[4]:
```
dfOrg['respuesta.pubmed_keys'] = dfOrg['respuesta.pubmed_keys'].apply(lambda x :
↪ str(x).split(','))

dfOrg = dfOrg.explode('respuesta.pubmed_keys').reset_index(drop=True)

dfOrg.head(10)
```

[4]:
```
   pedido.data.attributes.age pedido.data.attributes.diagnostic_main  \
0                        75.0                     FISTULA PERITONEAL
1                        75.0                     FISTULA PERITONEAL
2                        75.0                     FISTULA PERITONEAL
3                        75.0                     FISTULA PERITONEAL
4                        75.0                     FISTULA PERITONEAL
5                        75.0                     FISTULA PERITONEAL
6                        75.0                     FISTULA PERITONEAL
7                        75.0                     FISTULA PERITONEAL
8                        75.0                     FISTULA PERITONEAL
9                        75.0                     FISTULA PERITONEAL

  pedido.data.attributes.gender  articulo  respuesta.articlesRevisedYear  \
0                          male  27395425                           2018
```


[8]:
```
dataPubMedKeys, categoriesPubMedKeys = pd.factorize(dfOrg['respuesta.
↪pubmed_keys'])

dfOrg['respuesta.pubmed_keys'] = dataPubMedKeys
```

[9]:
```
dfOrg.head(10)
```

[9]:
```
   pedido.data.attributes.age  pedido.data.attributes.diagnostic_main  \
0                        75.0                                       0
1                        75.0                                       0
2                        75.0                                       0
3                        75.0                                       0
4                        75.0                                       0
5                        75.0                                       0
6                        75.0                                       0
7                        75.0                                       0
8                        75.0                                       0
9                        75.0                                       0

   pedido.data.attributes.gender  articulo  respuesta.articlesRevisedYear  \
0                              0  27395425                           2018
1                              0  27395425                           2018
2                              0  27395425                           2018
3                              0  27395425                           2018
4                              0  27395425                           2018
5                              0  27395425                           2018
6                              0  27395425                           2018
7                              0  27395425                           2018
8                              0  27395425                           2018
9                              0  27395425                           2018

   respuesta.articlesRevisedMonth  respuesta.pubmed_keys  utilidad
0                               1                      0       1.0
1                               1                      1       1.0
2                               1                      2       1.0
3                               1                      3       1.0
4                               1                      4       1.0
5                               1                      5       1.0
6                               1                      6       1.0
7                               1                      7       1.0
8                               1                      8       1.0
9                               1                      9       1.0
```

[10]:
```
print("age NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.attributes.age'])].
↪shape[0]))
print("diagnostic_main NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.
↪attributes.diagnostic_main'])].shape[0]))
```

```
print("gender NaN => " + str(dfOrg[pd.isnull(dfOrg['pedido.data.attributes.
↪gender'])].shape[0]))
print("articulo NaN => " + str(dfOrg[pd.isnull(dfOrg['articulo'])].shape[0]))
print("articlesRevisedYear NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
↪articlesRevisedYear'])].shape[0]))
print("articlesRevisedMonth NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
↪articlesRevisedMonth'])].shape[0]))
print("pubmed_keys NaN => " + str(dfOrg[pd.isnull(dfOrg['respuesta.
↪pubmed_keys'])].shape[0]))
print("utilidad NaN => " + str(dfOrg[pd.isnull(dfOrg['utilidad'])].shape[0]))
```

```
age NaN => 10
diagnostic_main NaN => 0
gender NaN => 0
articulo NaN => 0
articlesRevisedYear NaN => 0
articlesRevisedMonth NaN => 0
pubmed_keys NaN => 0
utilidad NaN => 14758
```

Remove row with age eq NaN

[11]:
```
dfOrg = dfOrg[pd.notnull(dfOrg['pedido.data.attributes.age'])]
```

3 Standardize the Data

Choosed “age”, “diagnostic_main”, “month” and “pubmed_keys” attributes (based on PCA_V3 study)

[12]:
```
from sklearn.preprocessing import StandardScaler

features = ["pedido.data.attributes.age",
    "pedido.data.attributes.diagnostic_main",
    "respuesta.articlesRevisedMonth",
    "respuesta.pubmed_keys",
    "utilidad"
]

# Separating out the features
x = dfOrg.loc[:, features].values

featuresTransformed = StandardScaler().fit_transform(x)

dfStandarized = pd.DataFrame(featuresTransformed, index=dfOrg.index,␣
↪columns=features)
dfStandarized['utilidad'] = dfOrg['utilidad']
```



```
dfStandarized
```

[12]:

	pedido.data.attributes.age	pedido.data.attributes.diagnostic_main	\
0	1.285887	-1.503163	
1	1.285887	-1.503163	
2	1.285887	-1.503163	
3	1.285887	-1.503163	
4	1.285887	-1.503163	
…	…	…	
15583	-0.607930	-0.586347	
15584	-0.607930	-0.586347	
15585	-0.607930	-0.586347	
15586	-0.607930	-0.586347	
15587	-0.607930	-0.586347	

	respuesta.articlesRevisedMonth	respuesta.pubmed_keys	utilidad
0	-1.463658	-1.089722	1.0
1	-1.463658	-1.080463	1.0
2	-1.463658	-1.071203	1.0
3	-1.463658	-1.061944	1.0
4	-1.463658	-1.052684	1.0
…	…	…	…
15583	-1.178433	-0.330441	NaN
15584	-1.178433	-0.978608	NaN
15585	-1.178433	0.891817	NaN
15586	-1.178433	-0.876753	NaN
15587	-1.178433	0.901077	NaN

[15578 rows x 5 columns]

4 Separe data by utilidad is defined

[13]:
```
dfDataSetComplete = dfStandarized[pd.notnull(dfStandarized['utilidad'])]

print(dfDataSetComplete.shape[0])

dfDataSetToPredict = dfStandarized[pd.isnull(dfStandarized['utilidad'])]

print(dfDataSetToPredict.shape[0])
```

```
830
14748
```


5 Random Forest

We check the number of results

[14]:
```
dfDataSetComplete.groupby('utilidad').size()
```

[14]:
```
utilidad
0.0    346
1.0    484
dtype: int64
```

Separe “utilidad” atribute from dataToTrain

[15]:
```
X = np.array(dfDataSetComplete.drop(['utilidad'],1))
y = np.array(dfDataSetComplete['utilidad'])
X.shape
```

[15]:
```
(830, 4)
```

[16]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

6 Exploring number of estimators

Via the sample size n of the bootstrap sample, we control the bias-variance tradeoff of the random forest. By choosing a larger value for n, we decrease the randomness and thus the forest is more likely to overfit. On the other hand, we can reduce the degree of overfitting by choosing smaller values for n at the expense of the model performance. In most implementations, including the RandomForestClassifier implementation in scikit-learn, the sample size of the bootstrap sample is chosen to be equal to the number of samples in the original training set, which usually provides a good bias-variance tradeoff.

https://towardsdatascience.com/gini-index-vs-information-entropy-7a7e4fed3fcb

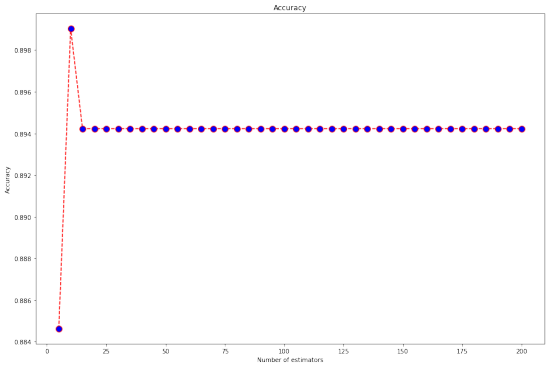
[17]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
↪accuracy_score

k_range = range(5, 205, 5)
accuracy = []

for k in k_range:
    forest_test = RandomForestClassifier(
        criterion='entropy',
        n_estimators=k,
        random_state=0
```



```
    )
    forest_test.fit(X_train, y_train)
    y_pred_test = forest_test.predict(X_test)

    accuracy.append(accuracy_score(y_test, y_pred_test))
```

[18]:
```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(figsize=(15, 10))
axs.plot(k_range, accuracy, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
axs.set_title('Accuracy')
axs.set_xlabel('Number of estimators')
axs.set_ylabel('Accuracy')
```

[18]:
```
Text(0, 0.5, 'Accuracy')
```



6.1 Evaluating the Algorithm

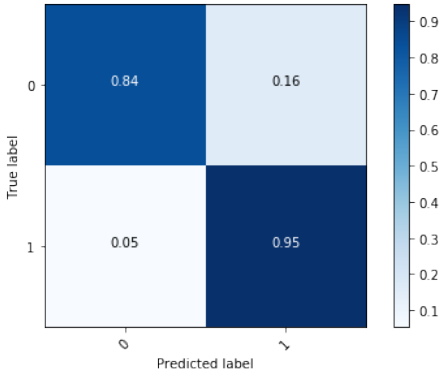
[19]:
```
forest = RandomForestClassifier(
    criterion='entropy',
    n_estimators=10,
    random_state=0
)

forest.fit(X_train, y_train)

y_pred = forest.predict(X_test)

print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.93      0.84      0.88        95
         1.0       0.88      0.95      0.91       113

    accuracy                           0.90       208
   macro avg       0.90      0.89      0.90       208
weighted avg       0.90      0.90      0.90       208

0.8990384615384616
```

[20]:
```
import itertools


cnf_matrix = confusion_matrix(y_test, y_pred)

def plot_confusion_matrix(cm, classes):
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cmap=plt.cm.Blues

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], ".2f"),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
```



```
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

n_classes=["0","1"]
plot_confusion_matrix(cnf_matrix, classes=n_classes)
```



7 Run Prediction

[21]:
```
result = forest.predict(dfDataSetToPredict[["pedido.data.attributes.age",
    "pedido.data.attributes.diagnostic_main",
    "respuesta.articlesRevisedMonth",
    "respuesta.pubmed_keys"
]])

result
```

[21]:
```
array([1., 1., 1., …, 0., 0., 0.])
```