Algoritmos de Feature Selection utilizados en estimación de esfuerzo de proyectos de desarrollo software

Trabajo Fin de Grado

**Grado en Ingeniería Informática**

**Autor**: Iván Iñaki Ajenjo Vicente

**Tutor**: Marta Fernández Diego

2019/2020

Resumen

En Machine Learning, es especialmente importante determinar aquellas variables que son relevantes para el objeto de estudio. En particular, los conjuntos de datos utilizados habitualmente en Ingeniería del Software tienen un alto número de variables, debiendo los investigadores y profesionales seleccionar aquellas que son más relevantes como variables independientes para el propósito de estimación de esfuerzo.

El objetivo del proyecto es conocer cómo se implementan estos algoritmos, especialmente los basados en la Teoría de la Información de Shannon. A partir de ahí se trata de adaptar algunos de ellos para mejorar su rendimiento.

**Palabras clave:** Feature Selection, Información mutua, desarrollo software, estimación de esfuerzo, ISBSG, R, Python

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed nisi turpis, iaculis a pulvinar quis, luctus et lorem. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nullam vitae purus eros, id auctor dolor. Sed et nisl quis nibh fermentum cursus ut at elit. Etiam condimentum porta leo quis tempor. Quisque commodo lobortis aliquet. Etiam tincidunt, libero ut vehicula euismod, justo augue lobortis sem, et facilisis velit lacus tristique dolor.

**Keywords :** integer, blandit, pharetra, urna, id.

Dedicatoria

Tabla de contenidos

[Tabla de Figuras 8](#_Toc46139746)

[Tabla de Código 9](#_Toc46139747)

[1 Introducción 10](#_Toc46139748)

[1.1 Motivación 10](#_Toc46139749)

[1.2 Objetivos 11](#_Toc46139750)

[1.3 Impacto esperado 11](#_Toc46139751)

[1.4 Estructura 11](#_Toc46139752)

[2 Estado del arte 11](#_Toc46139753)

[2.1 ISBSG 11](#_Toc46139754)

[2.2 Estimación de esfuerzo TO-DO 11](#_Toc46139755)

[2.3 Feature Selection 11](#_Toc46139756)

[2.4 Mutual Information 12](#_Toc46139757)

[3 Algoritmos de FS basados en MI propuestos 13](#_Toc46139758)

[3.1 MI vs MRMR 13](#_Toc46139759)

[3.2 1 lista vs 2 listas 14](#_Toc46139760)

[4 Metodología 15](#_Toc46139761)

[4.1 Herramientas 15](#_Toc46139762)

[4.1.1 Lenguajes de programación 15](#_Toc46139763)

[4.1.2 Entornos 16](#_Toc46139764)

[4.1.3 Librerias 17](#_Toc46139765)

[4.2 Pre-procesado de los datos 18](#_Toc46139766)

[4.2.1 Filtrado 18](#_Toc46139767)

[4.2.2 Set inicial de features 20](#_Toc46139768)

[4.2.3 Categorización 21](#_Toc46139769)

[4.3 Cross-validation múltiple 23](#_Toc46139770)

[4.3.1 Algoritmo MMRE 23](#_Toc46139771)

[4.3.2 Evaluador 24](#_Toc46139772)

[4.4 Generación de gráficas 24](#_Toc46139773)

[5 Resultados experimentales 27](#_Toc46139774)

[5.1 Precisión de los algoritmos de FS 27](#_Toc46139775)

[5.1.1 Convergencia de los algoritmos 27](#_Toc46139776)

[5.1.2 Influencia del valor de K 29](#_Toc46139777)

[5.1.3 Precisión de los algoritmos de FS 29](#_Toc46139778)

[5.2 Análisis de las variables seleccionadas 30](#_Toc46139779)

[5.2.1 Información mutua y redundancia de las variables 30](#_Toc46139780)

[5.2.2 Número de variables seleccionadas TO-DO 32](#_Toc46139781)

[5.2.3 Preferencia de uso de las variables 33](#_Toc46139782)

[6 Conclusiones 33](#_Toc46139783)

[6.1 Principales aportaciones 33](#_Toc46139784)

[6.2 Relación con los estudios cursados 33](#_Toc46139785)

[6.3 Limitaciones del trabajo 33](#_Toc46139786)

[6.4 Trabajos futuros 34](#_Toc46139787)

[7 Referencias bibliográficas 35](#_Toc46139788)

[8 Anexos 36](#_Toc46139789)

[8.1 Select\_features.py 36](#_Toc46139790)

[8.2 Dimensión de la solución propuesta 37](#_Toc46139791)

[8.3 Distintas implementaciones de MI 39](#_Toc46139792)

[8.4 Integración de Python con R 41](#_Toc46139793)

[8.5 Multiprocesamiento en Python 41](#_Toc46139794)

# Tabla de Figuras

[Figura 1 Estructura de datos de los resultados 21](#_Toc45648075)

[Figura 2 Evolución de las medias acumuladas de MMRE para k=1 25](#_Toc45648076)

[Figura 3 Box plot de la precisión (MMRE) dependiendo de los algoritmos FS 26](#_Toc45648077)

[Figura 4 Mutual Information de las variables independientes 28](https://d.docs.live.net/2e4f4e3558393dbf/UPV%202019/TFG/TFG%20con%20Plantilla%20v2.docx#_Toc45648078)

[Figura 5 MRMR de las variables independientes 29](https://d.docs.live.net/2e4f4e3558393dbf/UPV%202019/TFG/TFG%20con%20Plantilla%20v2.docx#_Toc45648079)

[Figura 6 mRMR de las variables seleccionadas 29](https://d.docs.live.net/2e4f4e3558393dbf/UPV%202019/TFG/TFG%20con%20Plantilla%20v2.docx#_Toc45648080)

[Figura 7 Numero de Variables seleccionadas por algoritmo 30](#_Toc45648081)

[Figura 8 Función de cálculo de MI con mutual\_info\_regression 33](#_Toc45648082)

[Figura 9 Función de cálculo mmre en R 33](#_Toc45648083)

[Figura 10 datos del código del proyecto 34](#_Toc45648084)

[Figura 11 ejemplo docstring para la función calcular\_mmre 34](#_Toc45648085)

# Tabla de Código

[Código 1 GFS 12](#_Toc45648192)

[Código 2 DFS 13](#_Toc45648193)

[Código 3 Selección de variables 17](#_Toc45648194)

[Código 4 Filtrado de proyectos 17](#_Toc45648195)

[Código 5 Recodificación de 1DBS 20](#_Toc45648196)

[Código 6 Recodificación de PPL 20](#_Toc45648197)

[Código 7 Calculo de MMRE 21](#_Toc45648198)

[Código 8 Evaluador 22](#_Toc45648199)

[Código 9 Inicialización de los datos 24](#_Toc45648200)

[Código 10 Subconjuntos de datos por método 24](#_Toc45648201)

[Código 11 Calculo de las medias acumuladas. 24](#_Toc45648202)

[Código 12 Grafica de medias acumuladas 25](#_Toc45648203)

[Código 13 Recodificación de las variables elegidas 25](#_Toc45648204)

[Código 14 Calculo de MI 28](#_Toc45648205)

# Introducción

El Aprendizaje Automático consiste en una disciplina de las ciencias informáticas, relacionada con el desarrollo de la Inteligencia Artificial, y que sirve para crear sistemas que pueden aprender por sí solos.

Es una tecnología que permite hacer automáticas una serie de operaciones con el fin de reducir la necesidad de que intervengan los seres humanos. Esto puede suponer una gran ventaja a la hora de controlar una ingente cantidad de información de un modo mucho más efectivo.

Lo que se denomina aprendizaje consiste en la capacidad del sistema para identificar una gran serie de patrones complejos determinados por una gran cantidad de parámetros. Al fin y al cabo, la máquina no aprende por si misma, sino un algoritmo de su programación que se modifica con la constante entrada de datos en la interfaz y de este modo puede predecir escenarios futuros o tomar decisiones de manera automática.

Los usos de esta tecnología son muy variados, se utiliza tanto en gran escala como en menor escala. Se utiliza para la detección de *malware*, en el comercio financiero, en el cuidado de la salud, marketing personalizado, motores de búsqueda… Sus aplicaciones son muy variadas, pero en el ejemplo tratado en este trabajo la utilizaremos para predecir el esfuerzo de desarrollo de los proyectos software.

## Motivación

Hoy en día se almacenan datos de forma masiva, en cualquier ámbito de la informática y en general de la tecnología. Estos son almacenados por si más adelante pueden ser de ayuda. Lo que esto nos produce es que tengamos un modelo de datos demasiado grande para poder trabajar con él con comodidad. Con esta gran cantidad de datos se pueden construir modelos con muchísima información. Esto por otro lado tiene alguna que otra contra. Estos modelos suelen ser muy pesados lo cual encarece el coste de computación y por tanto el tiempo de análisis. Es por esto por lo que cada vez más se utilizan algoritmos de *Feature Selection* o de selección de variable. Estos algoritmos se encargan de analizar las variables y seleccionar las mejores, reduciendo así el peso de los modelos. Esto hace que sean más sencillos de interpretar, que el tiempo de entrenamiento sea más corto, reducir el sobreajuste, etc.

Por tanto, la premisa central al utilizarse una técnica de selección de variable es el hecho de que un dato contiene muchas variables redundantes o irrelevantes y por tanto estas pueden ser eliminadas. Quiero recalcar la diferencia entre una variable irrelevante, con una redundante ya que esta diferencia es importante a la hora de plantear nuestros algoritmos. Las variables irrelevantes son aquellas que no contienen información de una variable objetivo. Mientras que una variable relevante, puede convertirse en redundante en presencia de otra variable con la que esté fuertemente relacionada.

En el desarrollo software, la estimación de esfuerzo es el proceso de predecir de la forma más realista posible la cantidad de esfuerzo requerido para desarrollar y mantener software. La práctica más común para la estimación de un proyecto software es la estimación de un experto. Una persona con experiencia, basándose en estas, realiza una estimación del esfuerzo del proyecto.

Utilizando aprendizaje automático esta estimación de esfuerzo puede verse mejorada. El aprendizaje automático como se ha comentado anteriormente se utiliza con muchos fines diferentes, pero uno de ellos es en los predictores.

## Objetivos

El objetivo del trabajo es el de analizar y evaluar distintos algoritmos de *Feature Selection* y utilizarlos para la creación de modelos de datos para la predicción del esfuerzo de desarrollo software.

El desarrollo se realizará utilizando Python para el manejo de los datos y la realización de parte de los cálculos y R que también se utiliza para realizar cálculos. Las gráficas y tablas se han generado utilizando Python. Para la evaluación del modelo se utiliza un script en Python que realiza las distintas iteraciones del algoritmo. Este llama a una librería de R que es utilizada para realizar la imputación.

El script deberá ir probando las variables previamente seleccionadas y ordenadas según unos criterios definidos, realizará los correspondientes cálculos de MMRE, esto se explicará más adelante, y todo este proceso se realizará midiendo los tiempos de ejecución.

## Impacto esperado

## Estructura

# Estado del arte

## ISBSG[[1]](#footnote-2)

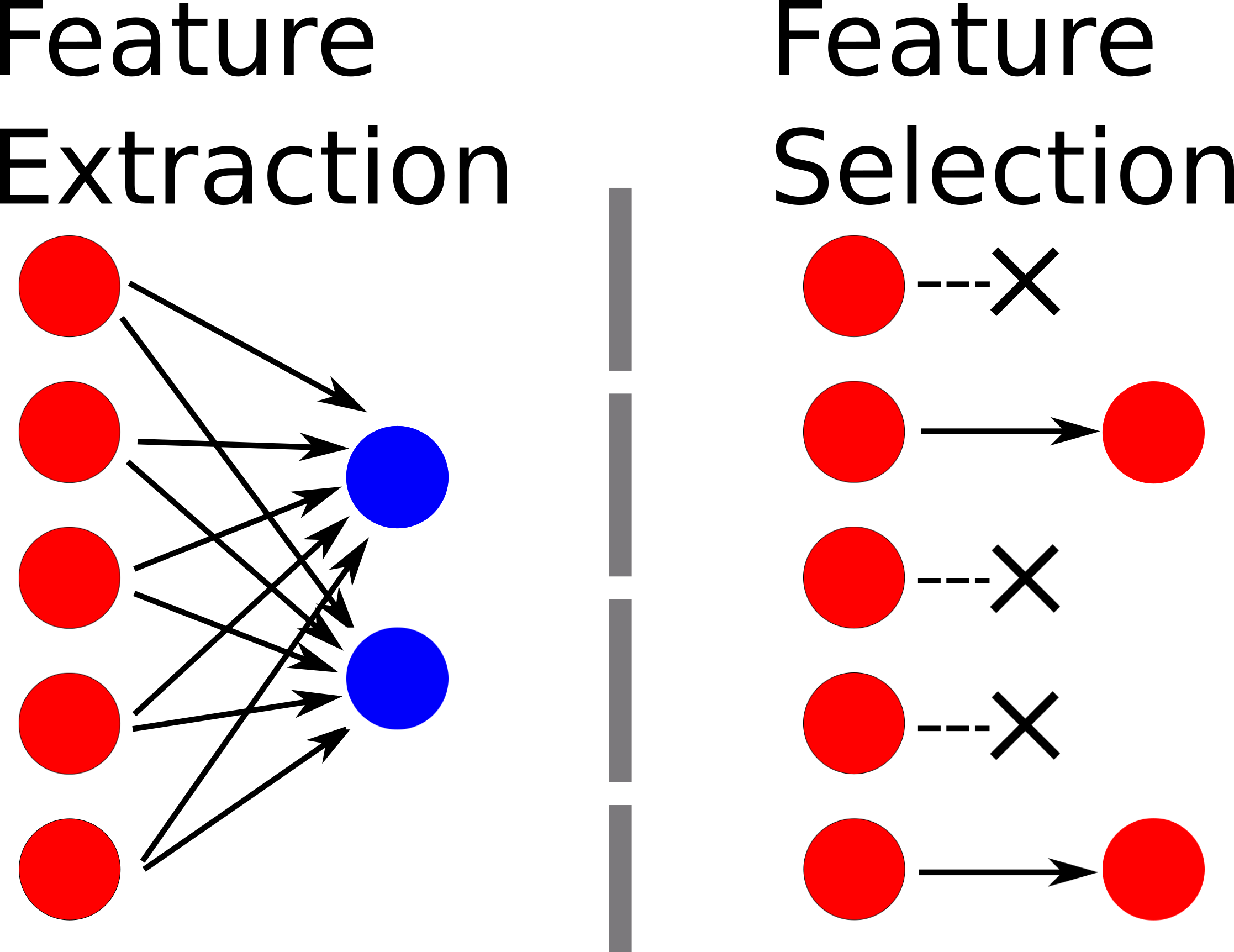
El *International Software Benchmarking Standards Group* (ISBSG) diseñó y mantiene dos repositorios internacionales públicos para mejorar la gestión de recursos de IT para negocios y gobiernos. El conjunto de datos de ISBSG ofrece una gran cantidad de información sobre el software completado, benchmarking, monitoreo, control de calidad… Sin embargo, hay cuestiones que deben tenerse en cuenta a la hora de utilizarlo. El trabajo experimental de este documento se basa en *ISBSG Release 12* que incluye 6006 proyectos y 126 características.

## Estimación de esfuerzo TO-DO

## Feature Selection

En *machine learning* y estadística, el *feature selection*, es el proceso de seleccionar un subconjunto de características pertinentes para su uso en construcción de modelos.

El objetivo de utilizar técnicas de FS es el de reducir el conjunto de datos con el que se trabaja a aquellos datos que son más relevantes. De forma que se eliminan características redundantes e irrelevantes. Las características redundantes e irrelevantes son dos tipos distintos, ya que una característica relevante puede ser redundante en presencia de otra característica relevante.

No se debe confundir con Feature Extraction, método a partir del cual se generan nuevas variables o características a partir de las existentes.

Un algoritmo de FS puede ser visto como una combinación de una técnica de búsqueda para proponer nuevos subconjuntos de características, junto con un evaluador que mide y puntúa los distintos subconjuntos. La elección del evaluador influye fuertemente en el algoritmo y son estas evaluaciones métricas las que distinguen entre las 3 categorías principales de algoritmos de selección de características.

* Envolvedores (Wrappers)
* Filtrado
* Embebidos

## Mutual Information

En la ciencia de datos hay muchos tipos de variables, en la propia base de datos de ISBSG hay unas 120 diferentes. Cada una con sus características propias, aunque las hay muy similares. Por ejemplo, si quisiéramos crear una base de datos de personas podríamos almacenar distintos datos, como por ejemplo su nombre, su edad, su estatura y peso etc… Algunas de ellas pueden ser totalmente independientes las unas de las otras por tanto no podríamos obtener información entre ellas. Por ejemplo, si comparamos el nombre de una persona, con su dirección sería prácticamente imposible el realizar una predicción. Pero si por el contrario comparamos su género con su altura si que podríamos llegar a una predicción posible. Ya que como norma general los hombres son más altos que las mujeres[[2]](#footnote-3). Evidentemente podemos encontrar estas variables con información y podemos utilizarlas a la hora de realizar una predicción de un valor aleatorio.

Por tanto, como hemos comentado antes algunas variables no son independientes una de las otras y por tanto esta dependencia nos puede causar información redundante cuando observamos ambas. Con MI queremos medir cuanta información comparten esas variables.

La *mutual information* o información mutua de dos variables aleatorias es una cantidad que mide la dependencia mutua de las dos variables, es decir, mide la reducción de la incertidumbre de una variable aleatoria, X, debido al conocimiento de otra variable aleatoria, Y.

Teniendo en cuenta estos conceptos podemos aprovecharlos en nuestro beneficio con el fin de obtener el mejor resultado posible. Conceptos de teoría de información, como este son utilizados frecuentemente en machine learning, con el fin de maximizar la calidad de los resultados.

# Algoritmos de FS basados en MI propuestos

## MI vs MRMR

El algoritmo de MI ordena las variables en función de la información mutua respecto a nuestra variable objetivo.

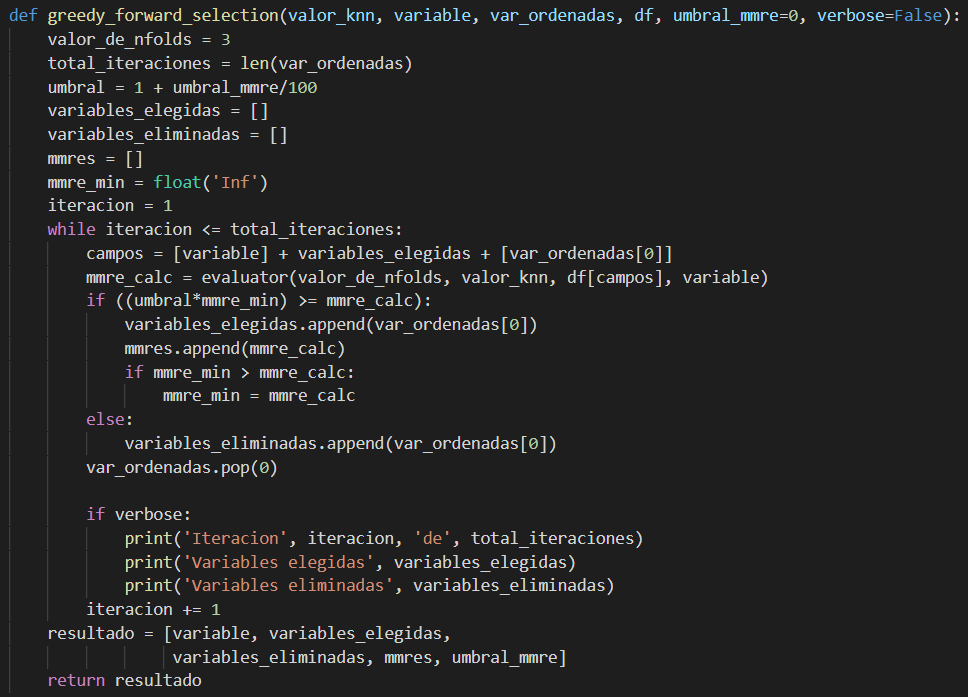
* MI\_1L. Este algoritmo emplea una estrategia sencilla. Consiste en añadir en cada paso la mejor variable de acuerdo con un criterio específico. Las variables se ordenan de acuerdo con la relevancia con la variable dependiente utilizando MI. Esta clasificación es obtenida utilizando el dataset completo. Después todas las variables son probadas de forma secuencial. Para saber si la variable es incluida en el modelo de predicción se prueba si mejora el modelo CBR previo en términos de MMRE. En caso de ser así la variable debe ser incluida entre las variables elegidas y en caso contrario se descarta
* Mi\_2L. Como el modelo tiene variables categóricas y variables numéricas, es interesante hacer una diferenciación entre estas. Por tanto, se ordenan en 2 listas, una con las variables categóricas y otra con las variables numéricas. Como en el caso anterior cada una de estas listas es ordenada de acuerdo con la información mutua de cada una de las variables respecto a la variable dependiente. En este caso el algoritmo prueba cada una de las 2 variables que encabezan las listas entre sí y elige la que más mejora el modelo de CBR utilizando MMRE. La variable seleccionada es eliminada de su lista y por tanto la próxima vez se vuelven a probar la primera de cada lista.

El algoritmo mRMR selecciona las variables con más información reduciendo la redundancia de estas. Eligiendo las variables con más información mutua con respecto a la variable objetivo, pero con menos información mutua entre sí mismas.

* mRMR\_1L. Se puede decir que una variable muy relevante para la dependiente puede ser inútil en caso de que su información se pueda obtener de otra de las variables seleccionadas. En ese caso la variable no debe ser seleccionada. Para resolver esto en cada paso de la búsqueda podemos seleccionar la variable con más diferencia entre la relevancia y la redundancia con las variables seleccionadas. La única diferencia con el algoritmo de MI\_1L es que en este caso las variables se ordenan de acuerdo con el criterio de mRMR, mínima redundancia máxima relevancia.
* mRMR\_2L. En este caso al igual que en MI\_2L se diferencia entre variables numéricas y categóricas. Y se sigue el mismo criterio de selección mencionado anteriormente.

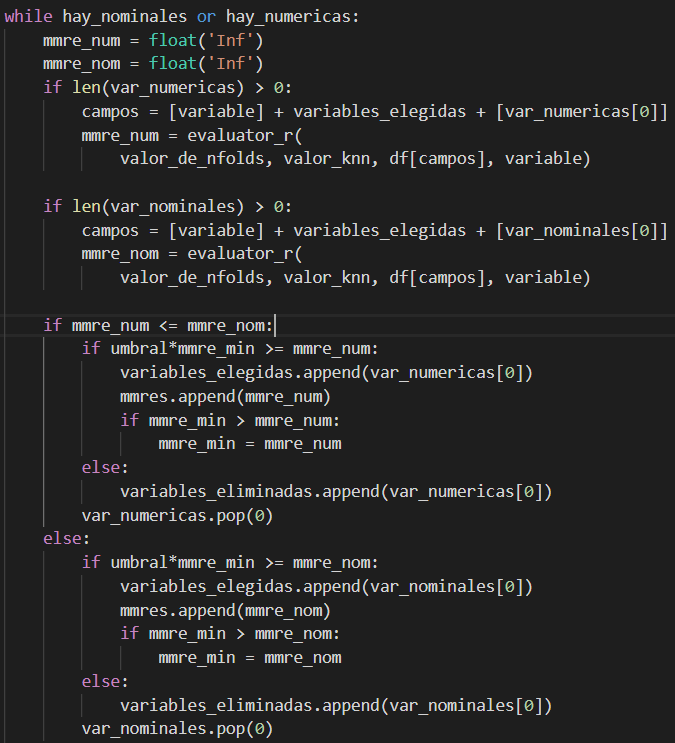
## 1 lista vs 2 listas

Greedy forward selection y doquire forward selection son las funciones que ejecutaran la selección de variables haciendo las llamadas necesarias al evaluador y comparando los resultados con los anteriores. El funcionamiento es sencillo, se selecciona la primera variable dependiente del dataset. Se calcula el MMRE con el evaluador, en caso de mejoría se selecciona la variable, en caso contrario esta se elimina.



Código 1 GFS

En el caso del DFS es algo diferente, ya que se emplean 2 listas 1 con las variables numéricas y otra con las variables categóricas. Se hace el cálculo de MMRE para la primera variable de ambas listas. Nos quedamos con la variable que mejora más el modelo basándose en el MMRE. Cuando no nos queden variables tendremos la lista de las variables elegidas, y el valor de MMRE del modelo con las variables elegidas. Este valor no será siempre el mismo porque como hemos visto en el evaluador, se utiliza una técnica de *k-fold cross validation* y por tanto en cada iteración se utiliza un particionado del *dataset* diferente.



Código 2 DFS

Como se puede ver en la iteración del bucle en doquire forward selection, la diferencia está en el hecho de trabajar con 2 listas de variables seleccionando así la que produce una mejoría mayor en el MMRE.

# Metodología

## Herramientas

### Lenguajes de programación

#### Python

Python[[3]](#footnote-4) es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad del código. Se trata de un lenguaje de programación multiparadigma ya que soporta orientación a objetos, programación imperativa y en menor medida programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License.

En el contexto de este trabajo la decisión era entre utilizar R o Python, finalmente se decidió hacerlo en Python por la experiencia previa utilizando el lenguaje.

#### R

R[[4]](#footnote-5) es un entorno y lenguaje de programación con un enfoque al análisis estadístico.

R nació como una reimplementación de software libre del lenguaje S, adicionado con soporte para alcance estático. Se trata de uno de los lenguajes de programación más utilizados en investigación científica, siendo además muy popular en los campos de aprendizaje automático (*machine learning*), minería de datos, investigación biomédica, bioinformática y matemáticas financieras. A esto contribuye la posibilidad de cargar diferentes librerias o paquetes con funcionalidades de cálculo y *graficación*.

Pese a que no es el lenguaje principal de programación del proyecto ha sido una herramienta muy útil y utilizada durante el desarrollo de este.

### Entornos

#### Anaconda



Anaconda[[5]](#footnote-6) es una distribución libre y abierta​ de los lenguajes Python y R, utilizada en ciencia de datos, y aprendizaje automático (machine learning). Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Está orientado a simplificar el despliegue y administración de los paquetes de software.

Las diferentes versiones de los paquetes se administran mediante el sistema de gestión de paquetes conda, el cual lo hace bastante sencillo de instalar, correr, y actualizar software de ciencia de datos y aprendizaje automático como ser Scikit-team, TensorFlow y SciPy.

#### Jupyter Notebook



Jupyter[[6]](#footnote-7) Notebook (anteriormente IPython Notebooks) es un entorno informático interactivo, open source, basado en la web para crear documentos de Jupyter notebook. El término "notebook" puede hacer referencia coloquialmente a muchas entidades diferentes, principalmente la aplicación web Jupyter, el servidor web Jupyter Python o el formato de documento Jupyter según el contexto. Un documento de Jupyter Notebook es un documento JSON, que sigue un esquema versionado y que contiene una lista ordenada de celdas de entrada/salida que pueden contener código, texto (usando Markdown), matemáticas, gráficos y texto enriquecidos, generalmente terminado con la extensión ".ipynb".

Jupyter Notebook se puede convertir a varios formatos de salida estándar (HTML, PDF …)

En lugar de utilizar Jupyter Notebook directamente sobre el navegador web, se utiliza su integración completa con Visual Studio Code.

### Librerias

#### Pandas

En computación y ciencia de datos Pandas[[7]](#footnote-8) es una biblioteca de software escrita como extensión de NunPy para manipulación y análisis de datos para el lenguaje de programación Python. Ofrece estructuras de datos y operaciones para manipular tablas numéricas.

Pandas es una librería para el análisis de datos que cuenta con las estructuras necesarias para limpiar los datos en bruto y que sean aptos para el análisis. Pandas es capaz de realizar tareas importantes como, fusionar datos o el tratamiento de datos perdidos.

La estructura básica de datos de Pandas es el DataFrame, una colección ordenada de columnas con nombres y tipos, parecido a una tabla de una base de datos. Sobre este se pueden aplicar filtros o realizar consultas para obtener la información deseada.

#### Scikit-Learn



Scikit-learn[[8]](#footnote-9) es una librería para aprendizaje automático de software libre para el lenguaje de programación Python. Incluye algoritmos de clasificación, regresión y análisis de grupos, k-means, etc. Está diseñada para interoperar con librerías numéricas y científicas como NumPy.

La gran variedad de algoritmos y utilidades de scikit-learn la convierten en una herramienta básica. En nuestro caso se utilizará tanto para calcular la Mutual Information como para realizar los K-fold.

#### Matplotlib

Matplotlib[[9]](#footnote-10) es una librería para la generación de gráficos a partir de datos contenidos en listas o arrays en Python. Proporciona una Api, pylab, diseñada para recordar a la de MATLAB. Se ha utilizado para generar todos los gráficos que aparecen a lo largo del trabajo.

#### Seaborn

Seaborn[[10]](#footnote-11) es una librería para hacer gráficos estadísticos en Python. Está construida sobre matplot y tiene una integración muy desarrollada con las estructuras de datos de Pandas. Será la herramienta utilizada para generar gran parte de los gráficos de resultados. Genera gráficos más atractivos e informativos de una forma sencilla.

## Pre-procesado de los datos

### Filtrado

Ya que ISBSG es un Dataset muy grande y heterogéneo, es necesario un proceso de preparación de datos antes de cualquier análisis.

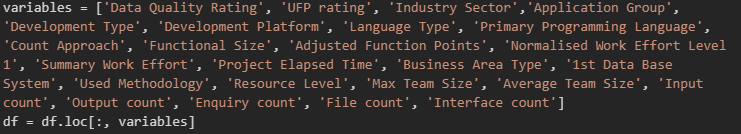
Como paso inicial se realiza la importación de la BD. En este caso se trata de un archivo csv delimitado por punto y coma. Como el archivo contenía más información que la propia base de datos, todos los datos innecesarios tales como recordatorios legales o de categorización de variables se eliminan. Como este proceso solamente es necesario hacerlo 1 vez se realiza manualmente. Se accede al archivo y se eliminan las 5 primeras líneas.

Una vez nos queda solamente información válida para la realización del proyecto podemos comenzar. Para realizar la importación de la BD se utiliza la librería Pandas la cual genera automáticamente un DataFrame con los datos importados. Como se van a realizar una serie de operaciones de limpieza del Dataset, se opta por realizar el trabajo utilizando un Jupyter Notebook. Esto nos permite comprobar en todo momento el estado de las distintas variables que se van generando, pudiendo ver los cambios en tiempo real sin necesidad de ejecutar todo el script cada vez que se quiere cambiar algo de código.



Con el archivo correctamente formateado la importación se realiza en 1 línea de código.

Para seleccionar un 1er grupo de columnas o features se puede realizar utilizando la funcion loc()[[11]](#footnote-12) pasándole una lista con los identificadores de las columnas a seleccionar.



Código 3 Selección de variables

De esta forma vamos acotando el dataset con las columnas que queremos.

Tabla 1 Criterios de selección de proyectos

|  |  |
| --- | --- |
| Criterio de selección | Proyectos Restantes |
| Calidad de datos general Alta | 3935 |
| Calidad funcional Alta |
| Esfuerzo del equipo de desarrollo conocido | 2249 |
| Esfuerzo del ciclo de vida completo |
| IFPUG versión 4.0+ | 1884 |

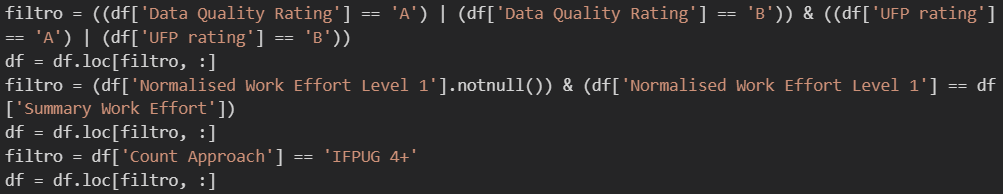
df['Data Quality Rating'] == 'A') | (df['Data Quality Rating'] == 'B'

df['UFP rating'] == 'A') | (df['UFP rating'] == 'B'

df['Normalised Work Effort Level 1'].notnull()

df['Normalised Work Effort Level 1'] == df['Summary Work Effort']

df['Count Approach'] == 'IFPUG 4+'



Código 4 Filtrado de proyectos

Como se puede observar en el fragmento de código, para realizar los distintos filtrados utilizamos la función loc(). Pero en esta introducimos las condiciones lógicas que deben superar nuestros proyectos objetivo.

### Set inicial de features

Tres variables de esfuerzo están disponibles en el dataset de ISBSG. La fundamental es Summary Work Effort (SWE), medido en horas. Es el esfuerzo total del proyecto contribuido por las empresas colaboradoras, pero SWE no cubre todas las fases del ciclo de vida del proyecto. Normalised Effort es la estimación de ISBSG del esfuerzo total cuando alguna de las fases que faltan son añadidas. Aun así, puede haber algunas inconsistencias entre proyectos, incluso cuando se utiliza Normalised Effort, porque el reporte de este esfuerzo proviene de diferentes participantes y esto se indica en la variable Resource Level. Level 1 implica a que el esfuerzo es reportado solamente por el equipo de desarrollo. Los Level 2 y 3 añaden el esfuerzo del equipo de soporte y las operaciones computacionales y el Level 4 añade el esfuerzo de los usuarios finales y los clientes. Por tanto, Normalised Work Effort Level 1 es el esfuerzo normalizado del equipo de desarrollo solamente.

Para empezar, nos quedaremos con 20 de las variables independientes más utilizadas en la estimación de modelos de esfuerzo.

De este set inicial de 20 descartaremos variables con un nivel de datos perdidos superior al 60%: Average Team Size, Business Area Type, Max Team Size e Input Count, Output Count, Enquirity Count, File Count e Interface Count.

También nos aseguraremos de que NWEL1 no tiene valores nulos y que los valores de Resource Level sean 1. Después de esto Resource Level puede ser descartada del set de variables puesto a que ya no nos aportará nada de información.

En este momento el subset incluye 1884 proyectos y 11 variables independientes y la dependiente NWEL1.

Por último, nos deshacemos de todos los proyectos que tienen valores nulos en alguna de las variables seleccionadas, lo que nos da un dataset final de 621 proyectos y 12 variables. Las variables independientes son las siguientes:

• Adjusted Function Points (AFP) es el tamaño ajustado para IFPUG, NESMA, FiAMA y MARK II. El tamaño es ajustado por un factor de conversión a AFP.

• Aplication Group (AG) es una variable derivada que agrupa Application Type de los proyectos en un único valor.

• 1st Data Base System (1DBS), la base de datos primaria utilizada en el proyecto. Esta variable tendrá que ser tratada más adelante en la categorización.

• Development Platform (DP) define la Plataforma de desarrollo determinada por el sistema operativo utilizado. Cada proyecto está clasificado como PC, Mid Range, Mainframe o Multi-Platform. DP es el mejor indicador del entorno en el que un proyecto es desarrollado.

• Development Type (DT) define si el Proyecto es un New Development, Enchancement o Re-Development

• Functional Size (FSZ) representa una función no ajustada de tamaño.

• Industry Sector (IS) identifica el tipo de organización que cede los datos del proyecto

• Language Type (LT) define el tipo de lenguaje de programación utilizado para el proyecto. La tercera generación es la dominante en nuestro subset, seguido de los de cuarta generación. En la práctica los lenguajes de 4a generación requieres un esfuerzo menor en la fase de programación, pero requieren un esfuerzo mayor en la fase de diseño.

• Project Elapsed Time (PET) representa el total de tiempo que ha transcurrido para el proyecto en meses.

• Primary Programming Language (PPL) indica cual es el lenguaje de programación principal del proyecto. Como los lenguajes de programación son de un tipo u otro en concreto esta información suele ser redundante con LT.

• Used Methodology (UM) define cuando una metodología ha sido utilizada en el desarrollo de un proyecto o no.

Tabla 2 Variables seleccionadas de ISBS

|  |  |  |
| --- | --- | --- |
| Adjusted Function Points | AFP | Continua |
| Aplication Group | AG | Categórica |
| 1st Data Base System | 1DBS | Categórica |
| Development Platform | DP | Categórica |
| Development Type | DT | Categórica |
| Functional Size | FSZ | Continua |
| Industry Sector | IS | Categórica |
| Language Type | LT | Categórica |
| Project Elapsed Time | PET | Continua |
| Primary Programming Language | PPL | Categórica |
| Used Methodology | UM | Categórica |
| Normalised Work Effort Level 1 | NWEL1 | Continua |

### Categorización

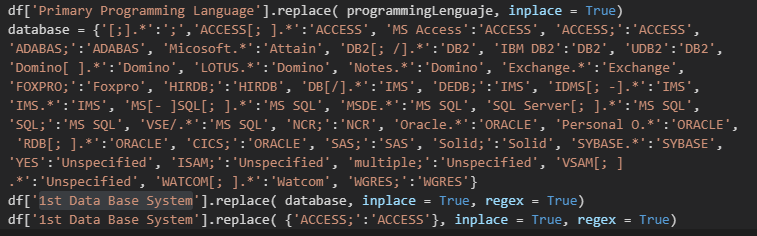
Algunas de estas variables tienen demasiados valores distintos o no están codificados en el mismo formato. Por tanto, se deben normalizar para minimizar la confusión y maximizar la consistencia.

En concreto se han recategorizado 2 variables: PPL y 1DBS. En el caso de PPL 2 de los proyectos tenían valores inválidos los cuales se han codificado como “Unspecified” y otros 3 proyectos se han convertido a nombres más comunes. Con estos cambios se han obtenido 32 valores diferentes.

En el caso de 1DBS ha sido algo más complejo. Como se ha comentado anteriormente 1DBS es la tecnología de base de datos primaria del software. Esta variable no está normalizada y simplemente incluye strings descriptivas en lugar de categorías predefinidas. Los valores no están definidos en un formato consistente. Algunos de los valores como “Yes”, “Multiple”, “ISAM”, etc, se han codificado como “Unspecified”. Y en el resto de los valores se han agrupado, por ejemplo “Oracle 7”, “Oracle 7.3”, se han codificado simplemente como “Oracle”.

Por desgracia, como se ha comentado anteriormente, la columna de 1st DataBase System no está correctamente formateada para su utilización. En la mayoría de los proyectos aparecen más de 1 sistema, o está repleto de “;”, o un mismo valor aparece codificado de más de 1 forma.

Por tanto, hay que realizar una serie de operaciones para limpiar los valores de esta columna. En este caso utilizaremos un diccionario y expresiones regulares[[12]](#footnote-13). Las expresiones regulares son patrones utilizados para encontrar una determinada combinación de caracteres dentro de una cadena de texto. Proporcionan una manera muy flexible de buscar o reconocer cadenas de texto.



Código 5 Recodificación de 1DBS

Como se puede ver se tiene que definir cada uno de los valores deseados para cada una de las expresiones regulares, las cuales se encargarán de realizar la búsqueda de los patrones y remplazar el valor por el definido.

Para comprobar nuestros resultados es muy útil la utilización de la función value\_counts()[[13]](#footnote-14). La cual nos devuelve en orden descendente las variables ordenadas por la cantidad de veces que aparecen. Así como dicha cantidad, por tanto, podemos observar si hay alguna variable que no cumple ningún requisito anterior y por tanto no se recodifica.

También se ha tenido que realizar una limpieza de la columna Primary Programming Language, pero en menor escala y por tanto mucho más sencilla.



Código 6 Recodificación de PPL

Donde una serie de valores no válidos se han codificado como “Unspecified” y otros se han codificado como apariciones anteriores, de esta forma se aumenta la calidad de los datos y la consistencia de estos.

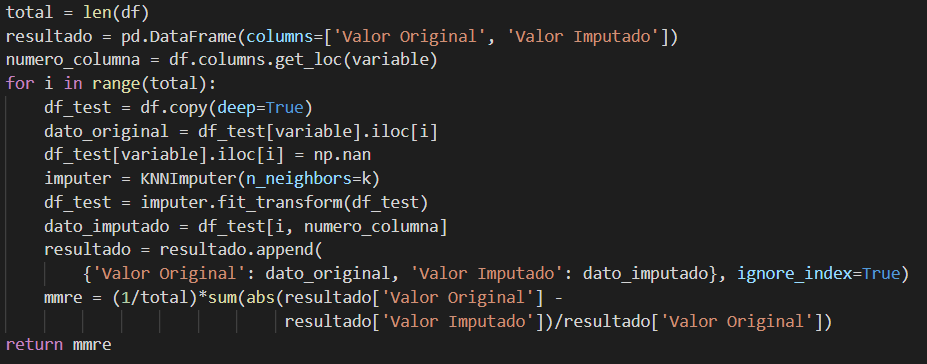
En este caso también se ha utilizado value\_counts() pero el proceso es mucho más sencillo ya que la codificación de los valores es mucho más consistente que en 1DBS.

## Cross-validation múltiple

La validación cruzada o cross-validation es una técnica muy utilizada para evaluar los resultados de un análisis estadístico. Proviene de la mejora del método de retención. Consiste en dividir en varios conjuntos complementarios los datos de muestra, realizar el análisis de un subconjunto y validar el análisis con el otro subconjunto.

En la validación cruzada de K iteraciones los datos de muestra se dividen en K subconjuntos. Uno de los subconjuntos se utiliza como casos de prueba y el resto (k-1) se utilizan como datos de entrenamiento. El proceso es repetido durante k iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente se hace la media aritmética de los resultados de cada iteración para obtener un único resultado. Se trata de un método muy preciso ya que evaluamos a partir de K combinaciones de datos de entrenamiento y de prueba, pero tiene una clara desventaja y es que su coste desde un punto de vista computacional es muy elevado. En este trabajo, cross validation se ejecuta 500 veces. En conclusión, para cada ejecución todo el data-set, los 621 proyectos se utilizan, pero cada vez las particiones son diferentes ya que las 3 divisiones se obtienen de forma aleatoria.

### Algoritmo MMRE

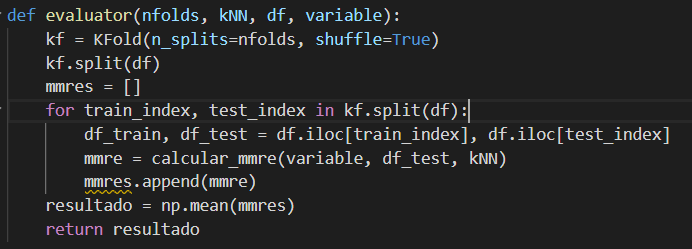
En este caso para el cálculo del MMRE se ha utilizado KNNImputer[[14]](#footnote-15) para realizar la imputación del valor a estimar utilizando los k vecinos más cercanos. Para esta solución se crea un DataFrame de resultados, donde tenemos una columna con el valor original y otra con el valor estimado. Por lo tanto, el algoritmo guarda el valor original, lo elimina del dataframe y lo imputa. Una vez imputado se recoge el valor para guardarlo en nuestro dataframe de resultados y el proceso se repite, pero con la siguiente fila del dataframe original. 

Código 7 Calculo de MMRE

En la funcion final solamente se devuelve el valor de MMRE para el dataframe determinado, pero durante la fase de desarrollo se devuelve tambien el dataframe con los valores imputados. Esto es de gran ayuda para realizar el posible debugging necesario para corregir errores.

### Evaluador

Para la función del evaluador se ha utilizado KFold[[15]](#footnote-16), esto se hace para realizar las cross validations del modelo. El funcionamiento es sencillo el dataset se divide y se alterna para calcular el valor de MMRE. Esta división es aleatoria para cada ejecución del evaluador. Con cada una de las kfolds se obtiene el valor de MMRE y finalmente se hace la media de los distintos resultados.



Código 8 Evaluador

La función del evaluador es llamada por las funciones greedy forward selection y doquire forward selection.

## Generación de gráficas

Para generar las gráficas y los resultados se ha utilizado un jupyter notebook, donde se importan todos los datos previamente generados y se hacen los cálculos necesarios para generar las gráficas.

La estructura de los resultados generados es la siguiente:



Figura 1 Estructura de datos de los resultados

Valor de MMRE, valor de K, lista de variables elegidas, método, tiempo que ha costado hacer el cálculo y la iteración.

En un primer momento se utilizó matplot, como se puede ver en las gráficas de MI o de mRMR, pero más adelante en el desarrollo, cuando se generó la gráfica de las medias acumuladas de MMRE matplot no era suficiente, por tanto, se planteó buscar alternativas. En una de las reuniones el tutor Fernando Gonzalez-Ladrón-de-Guevara comentó la posibilidad de utilizar seaborn. Seaborn, como se ha comentado antes, solamente es un complemento de matplot. Provee una API que trabaja sobre matplot y ofrece distintas opciones de personalización de estilo y colores. Haciendo ya por si solo las gráficas mucho más elegantes y legibles.

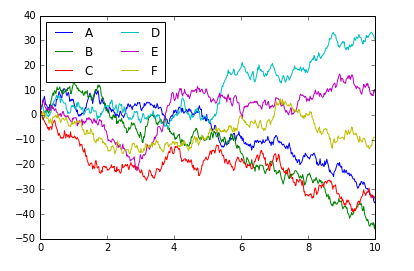
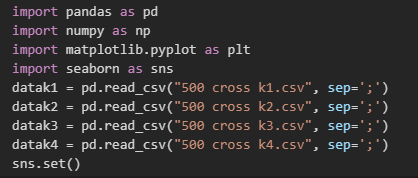


Ilustración 1 Ejemplo gráfica Matplot



Ilustración 2 Ejemplo gráfica seaborn

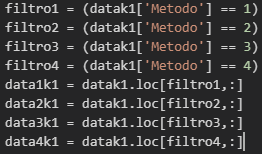
Para empezar a generar las gráficas lo primero es importar los datos y establecer seaborn como la herramienta de gráficos por defecto.



Código 9 Inicialización de los datos

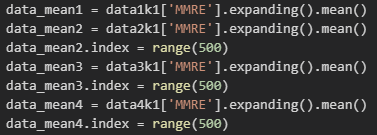
Sns.set() nos permite establecer seaborn para que trabaje directamente con los Dataframes de Pandas.

Con la estructura de datos diseñada como resultado de las ejecuciones del algoritmo, se genera 1 columna con todos los valores de MMRE para cada una de las iteraciones. Para acceder a ellas, separamos los datos en los distintos métodos aplicando un filtro, como se ha hecho anteriormente con los proyectos.



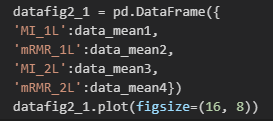
Código 10 Subconjuntos de datos por método

Una vez tenemos los datos separados por métodos se calcula la media acumulada de la columna de MMRE para cada uno de los métodos. Para esto utilizamos las funciones expanding()[[16]](#footnote-17) y mean() del pandas.Dataframe.



Código 11 Calculo de las medias acumuladas.

Para generar la gráfica la forma más sencilla es utilizar un DataFrame con los datos.

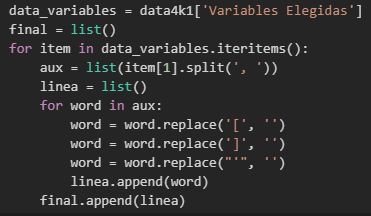


Código 12 Grafica de medias acumuladas

De esta forma generamos la gráfica prácticamente automática, como hemos hecho el sns.set() esta se genera utilizando seaborn, obteniendo una gráfica más agradable visualmente.

El resto de las gráficas y datos generados siguen un proceso más o menos parecido por tanto no voy a comentar como se han generado todas. Pero si que es interesante el proceso seguido para analizar la elección de variables del algoritmo.

La estructura de datos que sigue la columna de Variables Elegidas no ha sido la mejor, ya que Pandas la identifica como una cadena de caracteres. Por tanto, debemos limpiarla y procesarla antes de poder trabajar con ella. Esto no es un proceso complicado, separamos la cadena por las “,” y procedemos a la limpieza de todos los caracteres que no nos interesan.



Código 13 Recodificación de las variables elegidas

Con una correcta codificación de los resultados, al realizar la importación en pandas podemos definirlo como que son objetos y por tanto este proceso no sería necesario.

# Resultados experimentales

## Precisión de los algoritmos de FS

### Convergencia de los algoritmos

En primer lugar, la convergencia de los algoritmos es analizada. Para cuantificar la variación, cross-validation se repite 500 veces para estimar la distribución del rendimiento estadístico. Esto nos permite establecer el número de *cross validations* para el trabajo experimental.

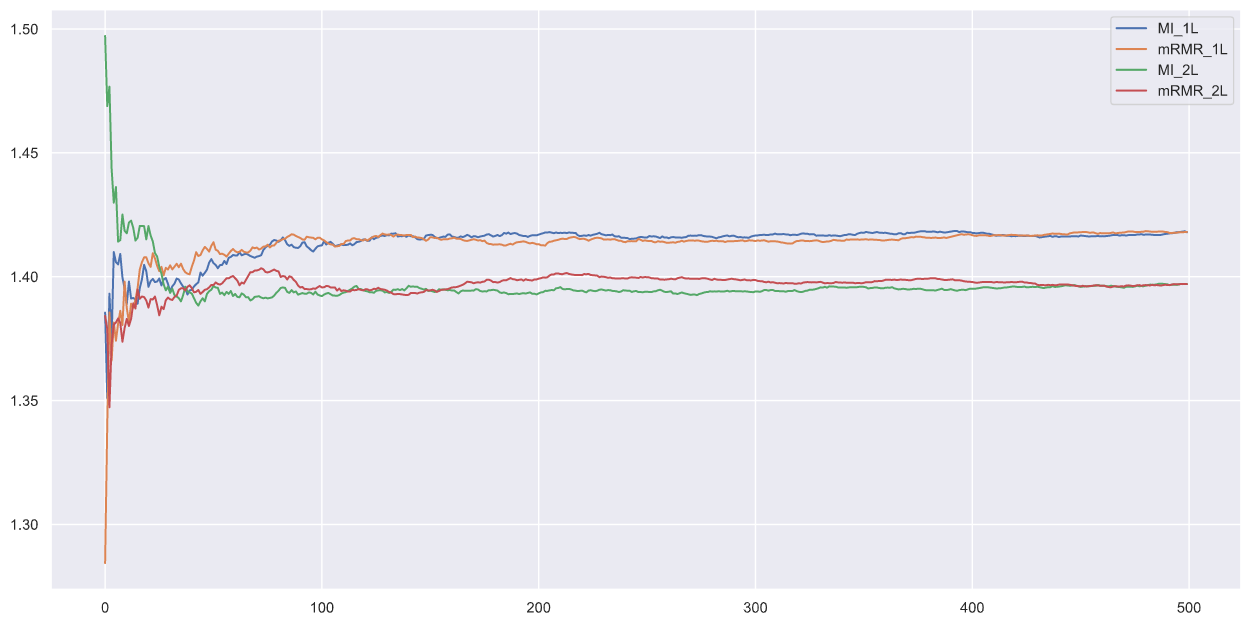


Figura 2 Evolución de las medias acumuladas de MMRE para k=1

La figura 1 muestra la evolución de la media de MMRE a lo largo de las *cross validations* realizadas para cada algoritmo con valor de k=1. En la figura se puede observar que la media fluctúa mucho a lo largo de las primeras 80 iteraciones, pero tras 250 se estabiliza y la fluctuación disminuye.

Pero sí que hay una clara diferencia entre los algoritmos que utilizan 1 lista (MI\_1L y mRMR\_1L) y 2 listas (MI\_2L y mRMR\_2L). Como se puede ver los valores de MMRE son inferiores en estos últimos.

Para comprobar la convergencia del algoritmo las medias son comprobadas con un margen de tolerancia. Cuando las medias acumulativas de los valores de mmre cambian menos de un determinado margen en un número de iteraciones. En esta tabla se pueden comprobar el número de iteraciones necesarias para cumplir con el margen de tolerancia. Los valores de esta oscilan entre el 0.1% y el 0.01%.

Tabla 3 Convergencia de los Algoritmos para los distintos valores de tolerancia (k=1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tolerancia | MI\_1L | mRMR\_1L | MI\_2L | mRMR\_2L |
| 0.1% | 145 | 140 | 138 | 146 |
| 0.09% | 158 | 144 | 142 | 158 |
| 0.08% | 189 | 177 | 195 | 170 |
| 0.07% | 212 | 186 | 208 | 193 |
| 0.06% | 271 | 245 | 283 | 230 |
| 0.05% | 356 | 298 | 325 | 285 |
| 0.04% | 356 | 310 | 366 | 350 |
| 0.03% | 412 | 420 | 422 | 414 |
| 0.02% | 0 | 0 | 0 | 0 |
| 0.01% | 0 | 0 | 0 | 0 |

### Influencia del valor de K

El mejor valor de k depende de los datos y de la aplicación de estos. Se ha comprobado el rendimiento del algoritmo para los distintos valores de k con los cuatro algoritmos de *Feature Selection*. En nuestro caso nos interesan los valores más parecidos al objetivo. Por tanto, nos interesan los valores de k pequeños. Se han seleccionado valores de k de 1 a 4 (1<=k<=4).

En la tabla 4 se muestra el valor de MMRE para los distintos valores de k teniendo en cuenta los 4 algoritmos sobre las 500 iteraciones de validación. Los mejores resultados son obtenidos para k = 1. Por tanto, el valor de k para las soluciones quedará fijado en k = 1.

Tabla 4 Valores de la media MMRE para los distintos valores de k

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| K | MI\_1L | mRMR\_1L | MI\_2L | mRMR\_2L |
| 1 | 1.4176 | 1.41809 | 1.39706 | 1.39702 |
| 2 | 1.4895 | 1.49497 | 1.4709 | 1.47229 |
| 3 | 1.5944 | 1.58901 | 1.5759 | 1.5734 |
| 4 | 1.6834 | 1.6791 | 1.6671 | 1.6596 |

### Precisión de los algoritmos de FS

Seleccionado el valor de k en 1, k=1, la media y la varianza de los valores de MMRE son MI\_1L (), mRMR\_1L (), MI\_2L (), mRMR\_2L (). La figura 3 nos muestra el *box plot* con los resultados de los valores de MMRE para los 4 algoritmos.

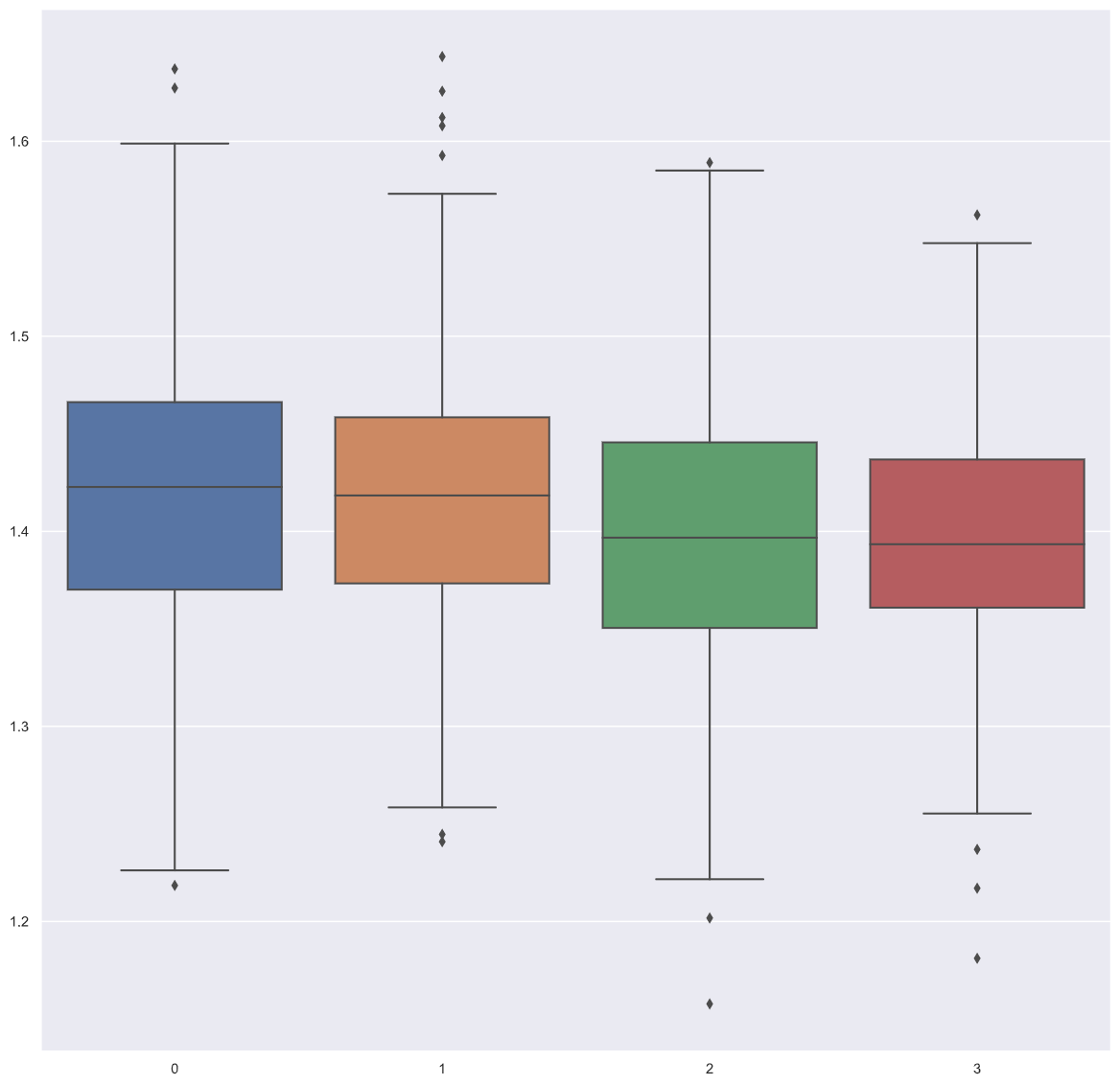


Figura 3 Box plot de la precisión (MMRE) dependiendo de los algoritmos FS

A parte de la precisión de la predicción, el coste computacional también se tiene en cuenta. Los algoritmos están probados en un AMD Ryzen 7 3700X @4.10 GHz y 16Gb de RAM. La tabla 5 muestra la media y la desviación típica de los tiempos de ejecución en 500 iteraciones para cada algoritmo para k=1.

Los *wrappers* son comúnmente criticados por requerir unos niveles de computación muy elevados. En la tabla 5 se pueden observar que los algoritmos que utilizan 1 lista 1L tienen unos tiempos de ejecución inferiores a los que utilizan 2 listas 2L. También se puede observar que no hay una diferencia substancial entre la utilización de Mi o de mRMR. Los tiempos son prácticamente los mismos, la diferencia está como se ha comentado anteriormente en si se utiliza una única lista o varias.

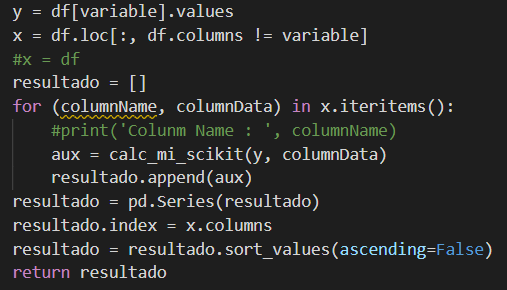
Tabla 5 Tiempos de ejecución de los algoritmos

|  |  |  |
| --- | --- | --- |
| Algoritmo | Media (segundos) | Desviación Típica |
| MI\_1L | 37.094 | 0.140 |
| mRMR\_1L | 37.450 | 0.309 |
| MI\_2L | 47.588 | 4.017 |
| mRMR\_2L | 47.516 | 3.958 |

## Análisis de las variables seleccionadas

### Información mutua y redundancia de las variables

Para realizar el cálculo de Mutual Information de las variables del dataframe. Se separan los valores de la variable objetivo del resto. Una vez lo tenemos separado recorremos cada una de las variables dependientes con la variable objetivo utilizando la función de MI.



Código 14 Calculo de MI

En este caso la función devuelve un pandas.Series[[17]](#footnote-18) con los valores del MI ordenados de mayor a menor. De esta forma posteriormente accediendo al índice de la serie obtenemos una lista con las variables ordenadas.

Para la creación de este algoritmo se han utilizado múltiples funciones, como se puede ver en el anexo 8.3, y formas de calcular MI finalmente quedándonos con la que más nos ha gustado.

La siguiente figura muestra el resultado de la ejecución del algoritmo de MI de las diferentes variables teniendo en cuenta el *dataset* completo. Estas variables son ordenadas de forma descendiente y nos servirá para ordenar los algoritmos MI\_1L y MI\_2L.

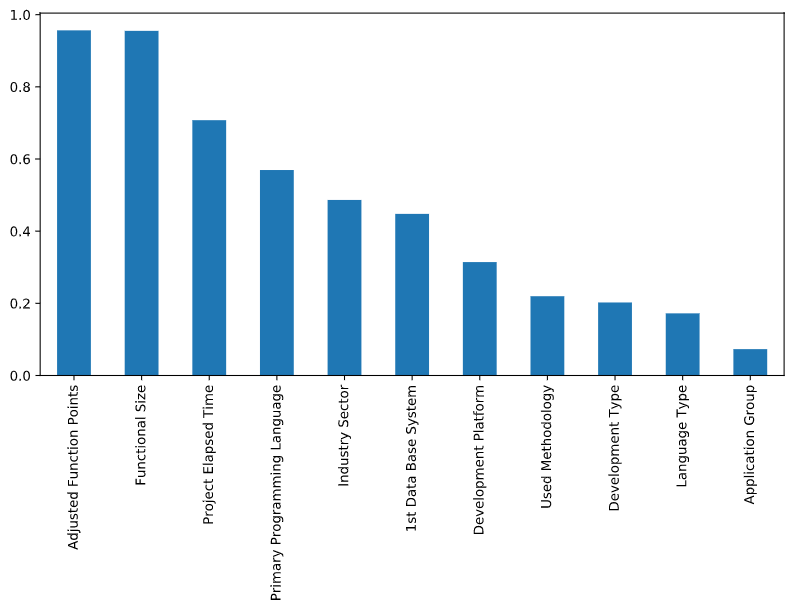
El resultado se ha obtenido con la función normalized\_mutual\_info\_score aplicada a todas las columnas del *dataset*. Pero se han hecho pruebas con distintas librerías tanto en Python como integrando R.

Figura 4 Mutual Information de las variables independientes

El resto de las gráficas se pueden observar en el anexo A del documento.

Como se puede observar en la figura 1 las variables Adjusted Function Points y Funcional Size tienen el MI más alto respecto a la variable de esfuerzo, seguido de Project Elapsed Time y de Primary Programming Language.

Llegados a este punto también es necesario analizar el orden de las variables teniendo en cuenta los algoritmos mRMR\_L1 y mRMR\_L2.

Por supuesto la primera variable seleccionada es la misma que la elegida por el algoritmo de MI. Como AFP tiene un valor de MI muy elevado quiere decir que realmente es una variable muy parecida a la de esfuerzo. Eso nos provoca que si comparamos el MI de AFP con el resto de variables nos sale una gráfica practicamente identica a la Figura 1. Esto hace que la segunda variable seleccionada sea Development Platform. Functional Size y Project Elapsed Time son las siguientes.

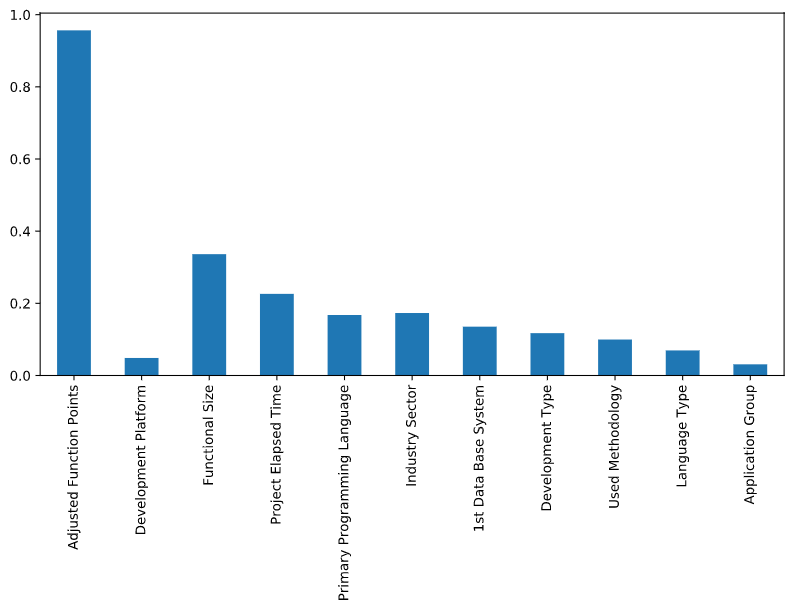


Figura 5 MRMR de las variables independientes

Figura 6 mRMR de las variables seleccionadas

### Número de variables seleccionadas TO-DO

En esta sección se analiza el número de variables seleccionadas por cada algoritmo. Se han hecho 500 iteraciones del algoritmo y en cada una de ellas se han seleccionado una serie de variables determinada.

Aparentemente los métodos que utilizan 2 listas emplean menos variables en construir los modelos que los que emplean 1, tal y como se puede ver en la tabla 4. Pero la diferencia no parece ser del todo destacable, aunque si es existente.

Tabla 6 Número de variables seleccionadas por algoritmo

|  |  |  |
| --- | --- | --- |
| Algoritmo | Media | Desviación Típica |
| MI\_1L | 3.84 | 1.2574 |
| mRMR\_1L | 3.91 | 1.2605 |
| MI\_2L | 4.068 | 1.36652 |
| mRMR\_2L | 4.042 | 1.27455 |

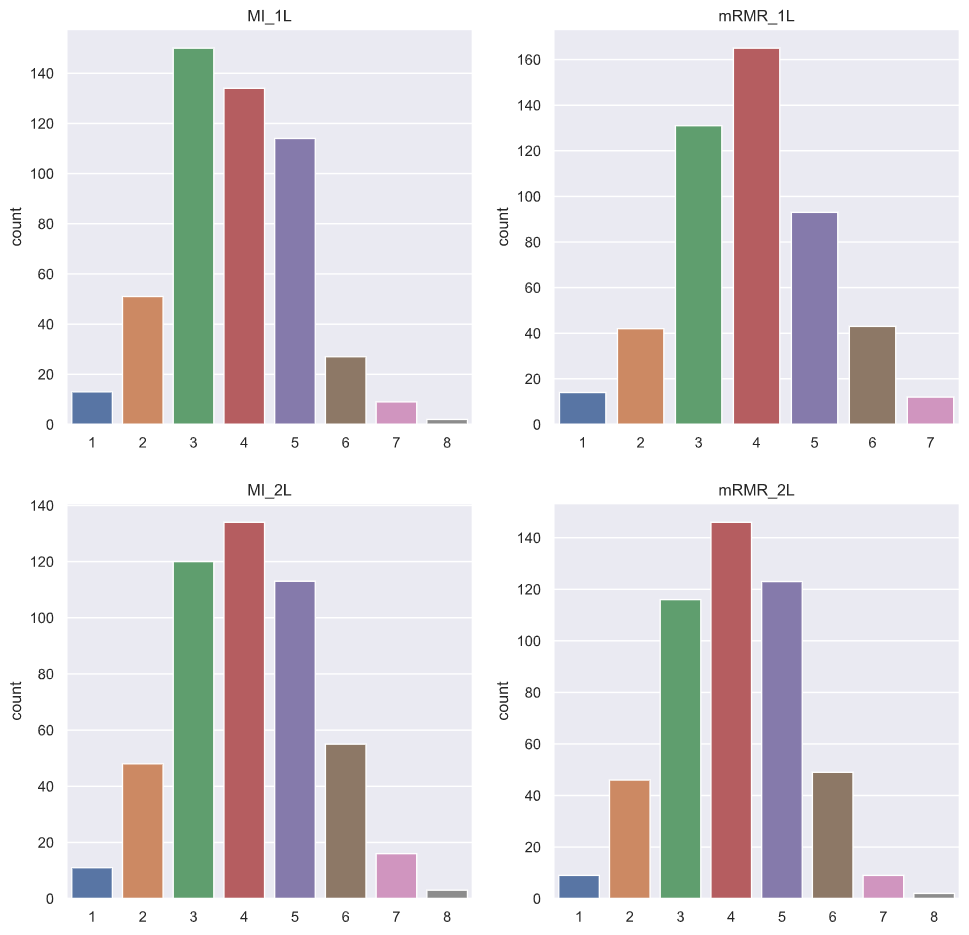


Figura 7 Numero de variables seleccionadas por algoritmo

### Preferencia de uso de las variables

# Conclusiones

## Principales aportaciones

## Relación con los estudios cursados

## Limitaciones del trabajo

## Trabajos futuros

# Referencias bibliográficas

1. Página de Wikipedia con información de Feature Selection

<https://en.wikipedia.org/wiki/Feature_selection>

1. Web de scolarpedia con información de Mutual Information

<http://www.scholarpedia.org/article/Mutual_information#:~:text=Mutual%20information%20is%20one%20of,variable%20given%20knowledge%20of%20another.>

1. Libro Python Data Science Handbook (la biblia)

<https://jakevdp.github.io/PythonDataScienceHandbook/>

1. Web de Towards data science

<https://towardsdatascience.com/>

1. Web de Medium.com

<https://medium.com/>

1. Documentación y tutoriales Seaborn

<https://seaborn.pydata.org/tutorial.html>

1. Web de ISBSG

<https://www.isbsg.org/>

# Anexos

## Select\_features.py

Quiero mencionar la librería select\_features.py la cual está desarrollada desde 0 por mi para este proyecto. Dado que el proyecto forma de 3 partes claramente diferenciadas, preparación de los datos, cálculo de resultados y representación de estos, a nivel de código también se divide en 3 partes.

La primera como se ha comentado anteriormente se produce en un jupyter notebook lo que facilita el desarrollo de esta. Ocurre lo mismo con la representación de los resultados.

Pero para generar los cálculos lo más sencillo era generar una librería a parte con los métodos necesarios para realizar los cálculos. Tener el código separado nos permite reutilizarlo en caso necesario en otros proyectos de Python. Por tanto, instalando las dependencias necesarias de los módulos utilizados en los cálculos tenemos una librería totalmente funcional. Todo el código se ha desarrollado teniendo en mente su posible reutilización de cara al futuro y este se publicará en github. Aun así, esto no ha sido posible al 100% ya que hay funciones que se han creado específicamente para este proyecto y por tanto no se podrán reutilizar en otros. Por ejemplo, para utilizar las funciones en R se tiene que recodificar el dataframe para que sea un objeto en R válido.

Las funciones disponibles en esta librería son las siguientes:

* setupR\_enviroment
* setupR\_enviroment\_knn
* calcular\_mi\_R
* calcular\_mi\_R\_2v
* calcular\_mrmr\_R
* recode\_dataframe\_R
* calc\_mi\_scikit
* calcular\_mi\_manual
* calcular\_mi
* recode\_dataframe
* recode\_dataframe\_v2
* calcular\_mrmr\_v2
* calcular\_mmre
* calcular\_mmre\_R
* determinar\_numero\_variables
* evaluator\_r
* evaluator
* greedy\_forward\_selection
* greedy\_forward\_selection\_r
* doquire\_forward\_selection
* doquire\_forward\_selection\_r

Como se puede intuir por los nombres de las funciones se han creado hasta 5 diferentes para realizar el cálculo de MI para distintos usos. Entre los usos están evidentemente el hecho de comprobar los resultados de las funciones comparándolos con otras.

Para comprobar resultados se han utilizado llamadas a métodos en R que realizan los mismos cálculos desde Python. Es por eso que para todas las funciones en Python existen sus respectivas funciones en R.

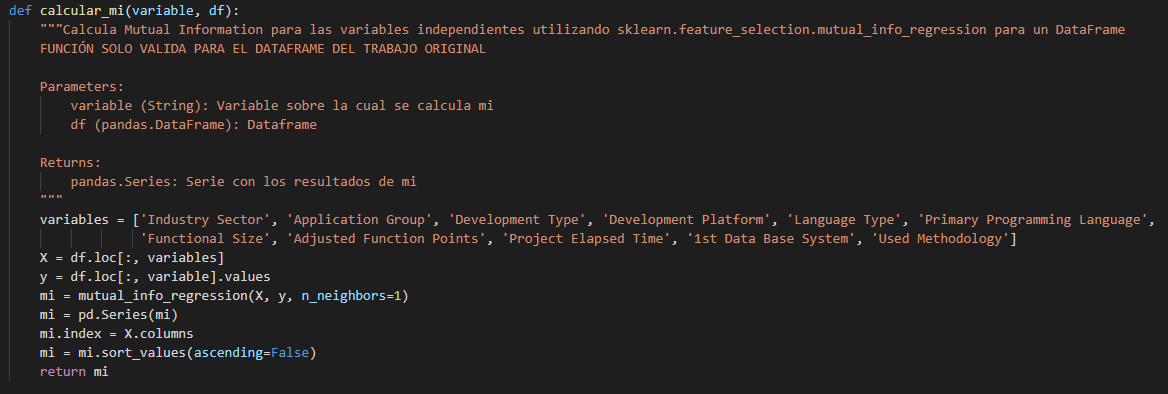


Figura 8 Función de cálculo de MI con mutual\_info\_regression

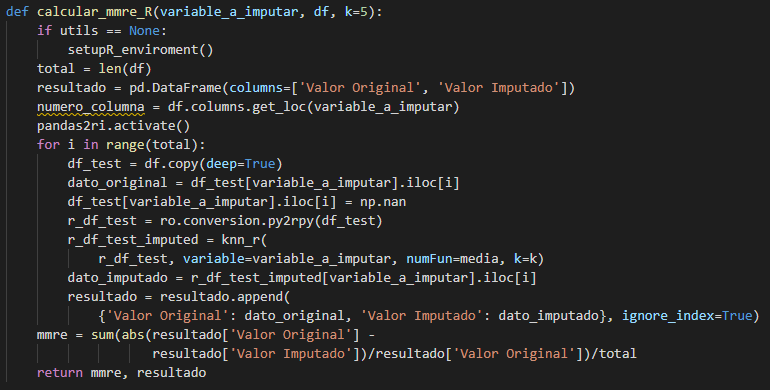


Figura 9 Función de cálculo mmre en R

Pese a esta librería, no todas las funciones están implementadas en ella, funciones cuya utilidad es exclusiva de este proyecto se han extraído de esta para evitar ruido y hacer un código más mantenible y legible. Por ejemplo, las funciones utilizadas para modificar el dataframe y hacerlo válido en R. Los nombres de las columnas del dataframe no puede tener espacios en R mientras que en Python no es un problema. Como solución sencilla se pueden cambiar los espacios por “\_”.

## Dimensión de la solución propuesta

En total el proyecto ocupa 33.5MB incluyendo a las BDs. Esto se debe al tamaño de las BDs y al de los archivos jupyter notebook generados durante el desarrollo y para la representación de resultados.

A continuación, se explican en detalle la carpeta con el código más importante del proyecto.

Los archivos con código Python, es decir los “.py” son 4 y tienen un total de 708 líneas de código y 126 de comentarios. Se ha de tener en cuenta que Python es un lenguaje de programación donde muchas operaciones que en otros lenguajes ocupan unas cuantas líneas en Python se realizan en 1. Por tanto la generación de ese código es más rápida que en otros lenguajes.

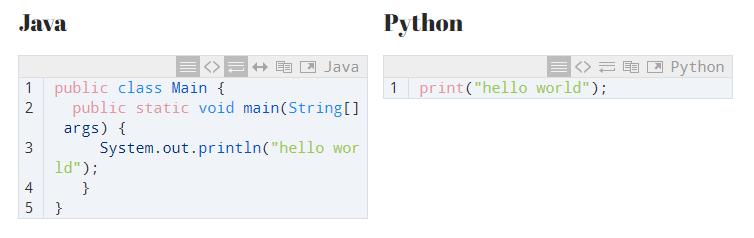


Ilustración 3 Comparación código en Java y Python

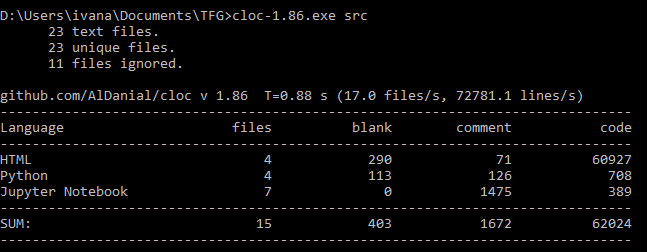


Figura 10 datos del código del proyecto

Siguiendo las guías de PEP 257 se han generado los “docstrings” de los métodos de la librería select\_features.py. Así como se ha añadido algún comentario para alguna línea más compleja.

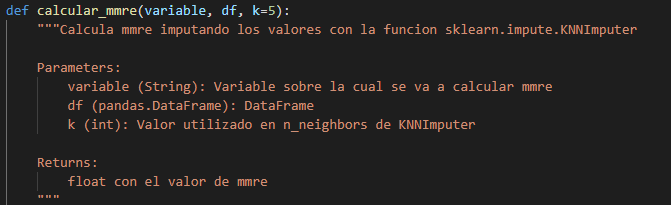


Figura 11 ejemplo docstring para la función calcular\_mmre

## Distintas implementaciones de MI

Esta primera gráfica se corresponde a la librería info\_gain.

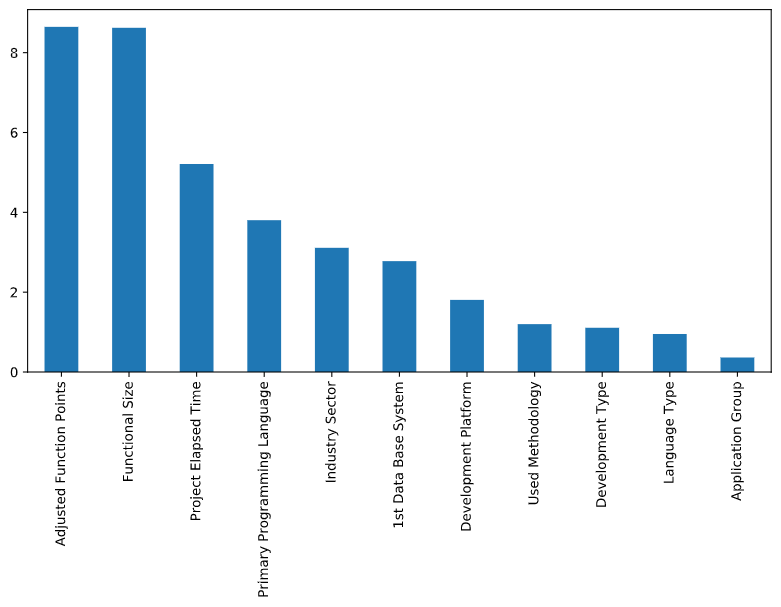
<https://github.com/Thijsvanede/info_gain>

Figura 12 MI con info\_gain

Esta grafica se hace aplicando manualmente al dataframe mutual\_info\_score de scikit

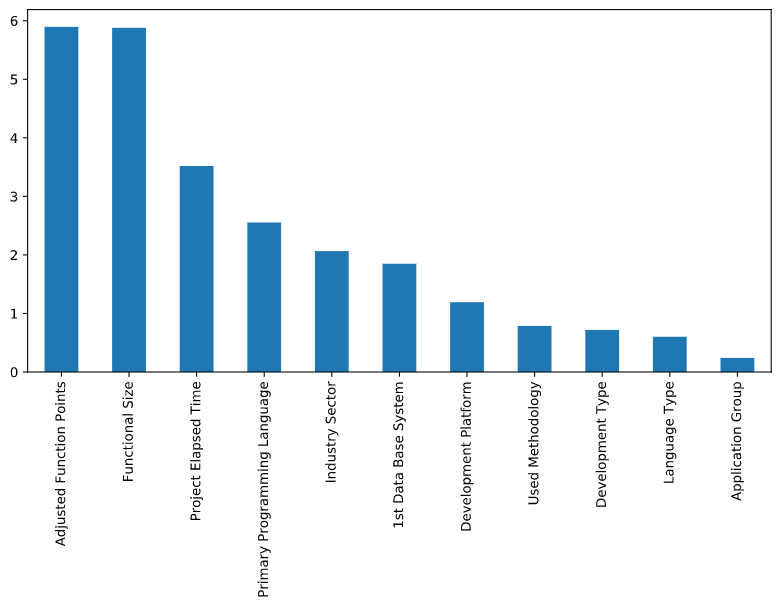
<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mutual_info_score.html>

Figura 13 MI con mutual\_info\_score

La tercera gráfica corresponde al método que comentamos en la reunión mutual\_info\_regression.

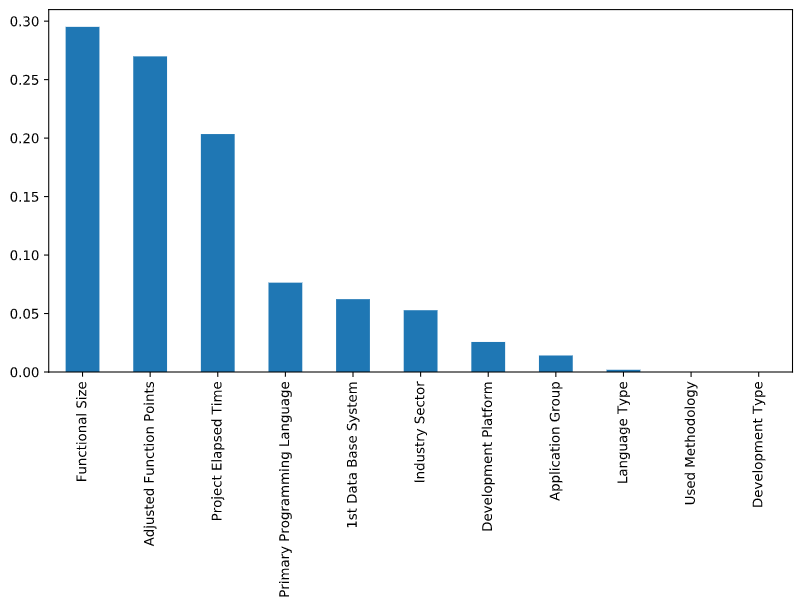


Figura 14 MI con mutual\_info\_regression

<https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html>

Gráfica de Mi con Fselector

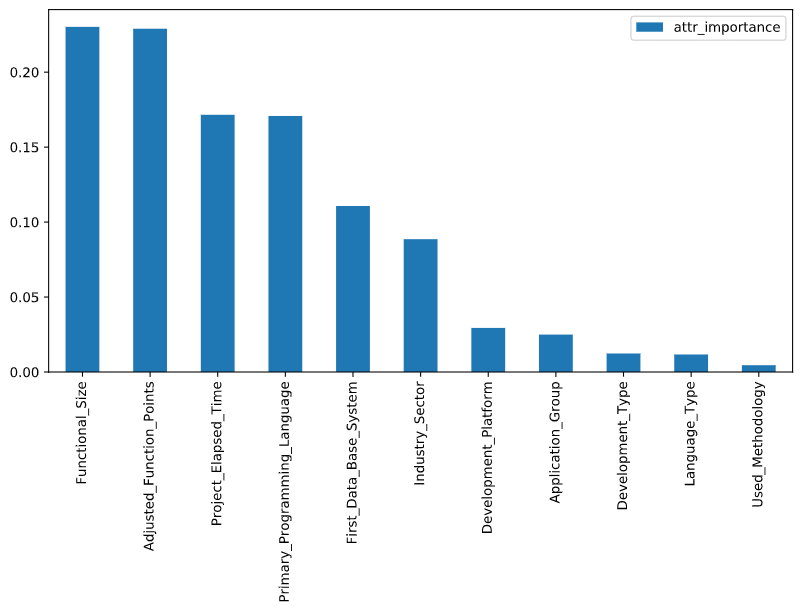


Figura 15 MI con FSelector

<https://cran.r-project.org/web/packages/FSelector/index.html>

## Integración de Python con R

A pesar de las múltiples herramientas que tiene Python para realizar los cálculos de MI, se decide realizar una integración con el paquete FSelector de R. Para realizar la misma se utiliza el módulo RPY2[[18]](#footnote-19).

Para realizar la instalación del módulo rpy2, se utiliza el procedimiento habitual.

“*pip install rpy2*”

Para poder utilizar RPY2 correctamente es necesaria la iniciación del entorno con los paquetes necesarios.

utils = rpackages.importr('utils')

utils.chooseCRANmirror(ind=1)

packages = ('FSelector')

utils.install\_packages(StrVector(packages))

FSelector = importr("FSelector")

information\_gain = FSelector.information\_gain

Si es cierto que no todo es tan fácil ya que por lo menos trabajando desde Windows debemos tener configurado correctamente el PATH. Así como el entorno Java correcto en caso de tener R instalado en 64 bits Java tiene que ser también de 64 bits. En mi ordenador estaba instalado por defecto la versión de 32bits, a pesar de que el sistema operativo es de 64.

La integración de R en Python nos permite combinar las ventajas de ambos lenguajes en un entorno único y más productivo. Además de poder utilizar paquetes de R para los que no existe un equivalente nativo en Python.

Recalcar que al utilizar esta librería si modificamos la asignación de recursos del sistema operativo, en este caso Windows obtenemos una mejora notable en los tiempos de ejecución. Evidentemente la utilización de la librería ralentiza el proceso respecto a realizarse de forma nativa en cualquiera de los 2 lenguajes.

## Multiprocesamiento en Python

Lo primero que es necesario definir es la diferencia entre el multiprocesamiento y la ejecución multihilo. En el multiprocesamiento múltiples procesos se ejecutan utilizando 1 o más núcleos. Mientras que en la ejecución multihilo, múltiples hilos se ejecutan sobre un mismo proceso. Por tanto, la ejecución multihilo nos viene bien para tareas que se bloquean por la I/O o cuando se espera a una respuesta o comunicación en red, etc. Cuando lo que necesitamos es obtener más potencia de las cpus para realizar cálculos complejos utilizaremos el multi procesamiento.

Aclarado esto a la hora de realizar este trabajo se ha investigado con las librerías de Python de multiprocesamiento con el fin de acelerar la ejecución.

Con la ayuda de la documentación oficial de Python se puede consultar información sobre el módulo multiprocessing[[19]](#footnote-20). Una vez con esto se empieza a hacer pruebas generando múltiples procesos.

**from** **multiprocessing** **import** Process

**def** f(name):

print('hello', name)

**if** \_\_name\_\_ == '\_\_main\_\_':

p = Process(target=f, args=('bob',))

p.start()

p.join()

El problema de este método es que debemos generar los procesos, iniciarlos y después asegurarnos de que estos finalizan 1 a 1. Esto se puede realizar utilizando una lista de procesos e iterando a través de esta. Pero esto nos deja el problema de que necesitamos muchas iteraciones ya que nuestro script realiza 500 runs por cada algoritmo, para cada valor de k. Por tanto, generar 8000 procesos no es una opción real. Por lo menos no en un ordenador de uso doméstico.

Por lo tanto, buscando en la documentación llegamos al módulo Pool. Este crea una “piscina” de procesos donde definimos un número máximo de procesos o trabajadores y conforme se van utilizando y terminando estos vuelven a ser lanzados hasta completar todos los cálculos.

**from** **multiprocessing** **import** Pool

**def** f(x):

**return** x\*x

**if** \_\_name\_\_ == '\_\_main\_\_':

**with** Pool(5) **as** p:

print(p.map(f, [1, 2, 3]))

Esto se fue desarrollando mientras el ordenador iba procesando iteraciones del script de resultados. Debido a que a pesar de tener resuelto el problema de ir generando los procesos e iniciarlos conforme van terminando nos queda el hecho de que se han de recoger los resultados. Para cuando se obtiene una solución correcta del script de resultados utilizando multiprocesamiento los resultados ya se han generado.

Para intercambiar datos entre procesos se pueden utilizar Pipes[[20]](#footnote-21), de esta forma se pueden enviar datos entre el proceso principal y los trabajadores.

Realmente la solución se puede realizar, pero ya no es útil ya que se han obtenido los datos de forma secuencial antes de tener terminado el script concurrente.

1. <https://www.isbsg.org/> [↑](#footnote-ref-2)
2. <https://es.wikipedia.org/wiki/Estatura> [↑](#footnote-ref-3)
3. <https://www.python.org/> [↑](#footnote-ref-4)
4. <https://www.r-project.org/> [↑](#footnote-ref-5)
5. <https://www.anaconda.com/> [↑](#footnote-ref-6)
6. <https://jupyter.org/> [↑](#footnote-ref-7)
7. <https://pandas.pydata.org/> [↑](#footnote-ref-8)
8. <https://scikit-learn.org/> [↑](#footnote-ref-9)
9. <https://matplotlib.org/> [↑](#footnote-ref-10)
10. <https://seaborn.pydata.org/> [↑](#footnote-ref-11)
11. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html> [↑](#footnote-ref-12)
12. <https://www.w3schools.com/python/python_regex.asp> [↑](#footnote-ref-13)
13. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.value_counts.html> [↑](#footnote-ref-14)
14. <https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html> [↑](#footnote-ref-15)
15. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html> [↑](#footnote-ref-16)
16. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.expanding.html> [↑](#footnote-ref-17)
17. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html> [↑](#footnote-ref-18)
18. <https://pypi.org/project/rpy2/> [↑](#footnote-ref-19)
19. <https://docs.python.org/3/library/multiprocessing.html> [↑](#footnote-ref-20)
20. <https://docs.python.org/3/library/multiprocessing.html#multiprocessing.Pipe> [↑](#footnote-ref-21)