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.

Dedicadotoria

Tabla de contenidos

[Tabla de Figuras 5](#_Toc44413142)

[1 Introducción 6](#_Toc44413143)

[2 Métodos de Feature Selection basados en Mutual Information 6](#_Toc44413144)

[2.1 Feature Selection 6](#_Toc44413145)

[2.2 Algoritmos Feature Selection Propuestos 7](#_Toc44413146)

[2.2.1 Mutual Information 7](#_Toc44413147)

[2.2.2 Mínima Redundancia Máxima Relevancia 7](#_Toc44413148)

[3 Metodología 8](#_Toc44413149)

[3.1 Pre procesado de datos 8](#_Toc44413150)

[3.1.1 ISBSG 8](#_Toc44413151)

[3.1.2 Filtrado 8](#_Toc44413152)

[3.1.3 Set Inicial de Features 9](#_Toc44413153)

[3.1.4 categorización 10](#_Toc44413154)

[3.2 Estimación de esfuerzo de desarrollo software basado en casos 11](#_Toc44413155)

[3.3 Comparativa de algoritmos de Feature Selection 11](#_Toc44413156)

[3.4 Multiple 3-Fold Cross Validation 12](#_Toc44413157)

[4 Resultados Experimentales 12](#_Toc44413158)

[4.1 Precisión de los algoritmos de Feature Selection 12](#_Toc44413159)

[4.1.1 Convergencia de los algoritmos 12](#_Toc44413160)

[4.1.2 Influencia del valor K en la precisión de los algoritmos 13](#_Toc44413161)

[4.1.3 Precisión de los algoritmos de Feature Selection 13](#_Toc44413162)

[4.2 Análisis de las features seleccionadas 14](#_Toc44413163)

[4.2.1 Información Mutua y Redundancia de las variables independientes 14](#_Toc44413164)

[4.2.2 Numero de Variables seleccionadas dependiendo de los algoritmos 15](#_Toc44413165)

[4.2.3 Preferencia de uso de las variables dependiendo del algoritmo 16](#_Toc44413166)

[5 Tecnología utilizada 17](#_Toc44413167)

[5.1 Python 17](#_Toc44413168)

[5.2 Anaconda 17](#_Toc44413169)

[5.3 Pandas 17](#_Toc44413170)

[5.4 Scikit-learn 18](#_Toc44413171)

[5.5 Jupyter Notebook 18](#_Toc44413172)

[5.6 Matplotlib 18](#_Toc44413173)

[5.7 Seaborn 18](#_Toc44413174)

[6 Proceso de desarrollo 19](#_Toc44413175)

[6.1 Importación de la BD 19](#_Toc44413176)

[6.2 Limpieza de las variables seleccionadas (1DBS) 20](#_Toc44413177)

[6.3 Algoritmos de MI 21](#_Toc44413178)

[6.4 Algoritmo MMRE 22](#_Toc44413179)

[6.5 Evaluador 22](#_Toc44413180)

[6.6 Greedy Forward Selection y Doquire Forward Selection 23](#_Toc44413181)

[6.7 Generación de las gráficas 23](#_Toc44413182)

[Referencias 27](#_Toc44413183)

[8 Anexo 28](#_Toc44413184)

[8.1 Anexo A – Gráficas de ejecución de MI 28](#_Toc44413185)

[8.2 Anexo B – Integración de Python con R 30](#_Toc44413186)

# Tabla de Figuras

[Figura 1 Evolución de las medias acumuladas de MMRE para k=1 13](#_Toc44364331)

[Figura 2 Box plot de la precisión (MMRE) dependiendo de los algoritmos FS 13](#_Toc44364332)

[Figura 3 Mutual Information de las variables independientes 14](https://d.docs.live.net/2e4f4e3558393dbf/UPV%202019/TFG/TFG%20con%20Plantilla.docx#_Toc44364333)

[Figura 4 MRMR de las variables independientes 15](https://d.docs.live.net/2e4f4e3558393dbf/UPV%202019/TFG/TFG%20con%20Plantilla.docx#_Toc44364334)

[Figura 5 mRMR de las variables seleccionadas 15](https://d.docs.live.net/2e4f4e3558393dbf/UPV%202019/TFG/TFG%20con%20Plantilla.docx#_Toc44364335)

# Introducción

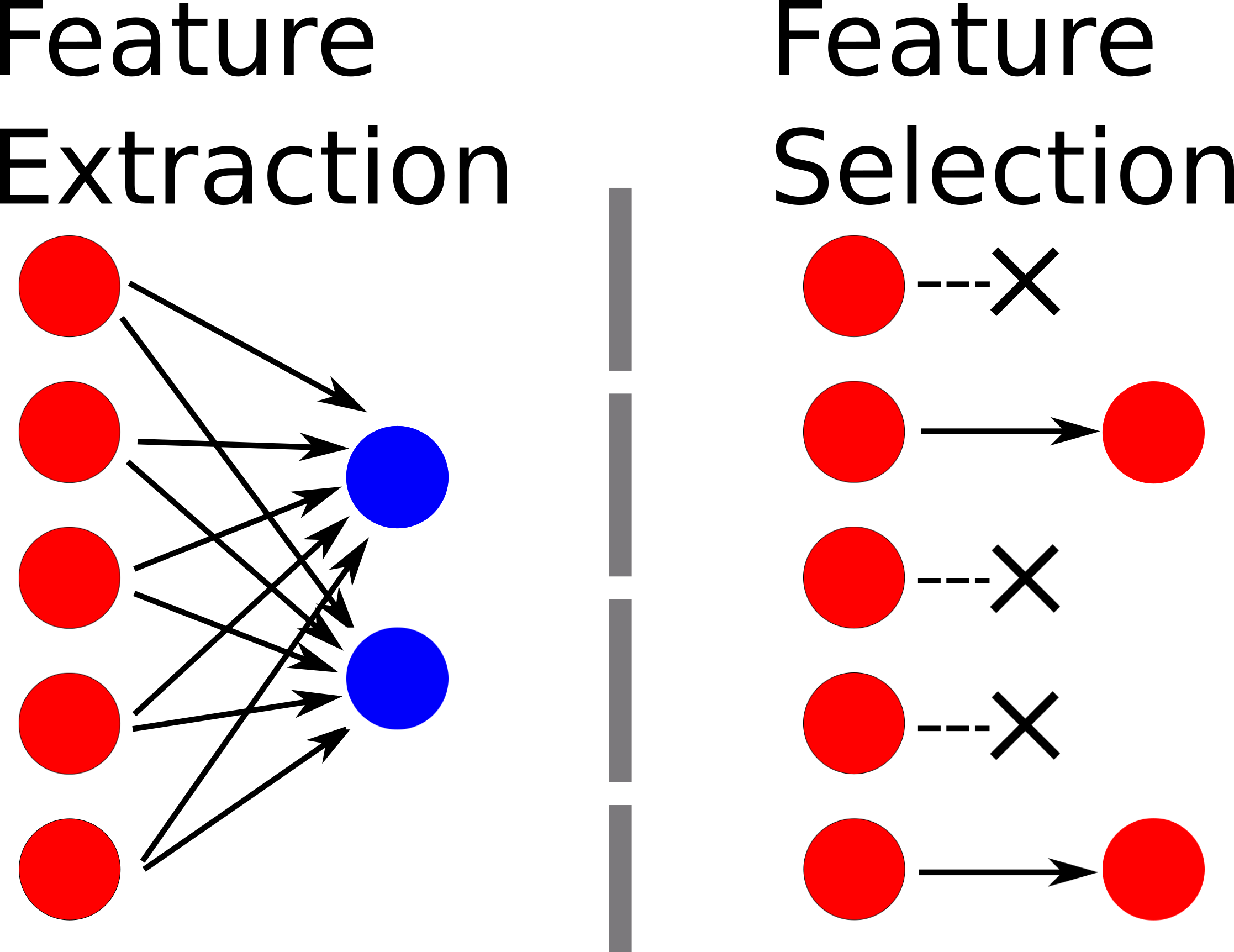
Generalmente, la estimación de esfuerzo del desarrollo software, requiere datos matemáticos en un formato factible a través del Pre Procesado de Datos (PPD). Las técnicas de PPD consisten en una reducción de los datos y el tratamiento de los datos que no están disponibles. La reducción de datos consiste en la reducción del tamaño del dataset mediante Feature Selection (FS) o Case Selection. FS reduce el tamaño de los DataSet selecciónando un subset óptimo con las variables más relevantes, descartando el resto. Las variables o features eliminadas son o irrelevantes para el problema tratado o redundantes comparadas con el resto de las variables seleccionadas. Por tanto, al tener un data set reducido se reduce también el coste de computación de los análisis que se realizan.

# Métodos de Feature Selection basados en Mutual Information

## 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 puntua 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

## Algoritmos Feature Selection Propuestos

Los algoritmos de FS propuestos utilizan Mutual Information como una medida de relevancia y redundancia. Para el cálculo de MI se han utilizado distintas librerías para poder realizar comprobaciones. Se ha utilizado normalized\_mutual\_info\_score de Scikit Learn. También se ha utilizado la librería info\_gain.info\_gain. Y se ha utilizado information.gain del paquete de R FSelector. Para realizar los cálculos finalmente se utilizó la función de la librería de Scikit Learn. La sintaxis es normalized\_mutual\_info\_score(labels\_true, labels\_pred).

### Mutual Information

La 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.

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 si 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.

### Mínima Redundancia Máxima Relevancia

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.

Por tanto, podemos diferenciar los algoritmos de distintas formas. En un primer caso se pueden diferenciar por el criterio seguido a la hora de ordenar las variables, MI\_1L y MI\_2L ordenan las variables en función de su relevancia (MI) con respecto a la variable dependiente NWEL1 Por otro lado tenemos mRMR\_1L y mRMR\_2L que también tienen en cuenta la redundancia de las variables entre ellas reduciéndola.

En el segundo caso o la segunda forma de diferenciar los algoritmos es en el caso de que estos utilicen 1 lista o 2. Es decir si diferencian entre las variables categóricas y las variables numéricas. Por tanto, podemos agrupar los algoritmos de MI\_1L y mRMR\_1L ya que estos solamente utilizan 1 lista de variables como entrada y no diferencian entre las variables numéricas y las categóricas. Por otro lado, MI\_2L y mRMR\_2L si utilizan 2 listas diferenciando entre variables numéricas y categóricas.

# Metodología

## Pre procesado de datos

### ISBSG

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.

### Filtrado

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

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+'

### 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”.

## Estimación de esfuerzo de desarrollo software basado en casos

El razonamiento basado en casos es el proceso de solucionar nuevos problemas basándose en soluciones a problemas anteriores. El razonamiento basado en casos o case base reasoning (CBR) es un tipo de sistema experto. Un sistema experto es aquel que intenta imitar el comportamiento de un ser humano experto en alguna temática.

Ahora bien, tenemos que analizar los elementos necesarios para que un sistema experto sea realmente útil y por tanto una herramienta efectiva tanto por la interacción con el usuario como por la calidad de la respuesta. Para proporcionar una información útil el sistema necesita una serie de elementos. En primer lugar, una base de conocimientos donde se almacenará todo el conocimiento que tiene el experto sobre el tema que tratará. También es necesaria una base de hechos, donde se guardarán todas las deducciones realizadas por el sistema. Un motor de inferencia que generará las conclusiones a partir de la base de conocimiento y la de hechos. Y por último una interfaz de usuario, que en nuestro caso son una serie de gráficas y tablas que genera el propio sistema.

La utilización de una aproximación de CBR para la estimación de esfuerzo viene dada de la base de que los proyectos de desarrollo de software son similares entre si mismos. Por tanto, en este caso se utilizarán proyectos similares de nuestra base de conocimiento para generar una estimación de nuestro nuevo proyecto. K-nearest neighbor (KNN) es una técnica de clasificación que debe ser una de las primera a estudiar cuando se tiene un conocimiento previo de la distribución de los datos. Esta técnica es ampliamente utilizada en CBR. KNN es bastante sensible dependiendo del numero de variables utilizadas y es menos efectiva cuando hay muchas variables irrelevantes o que añaden ruido. Por tanto, con una buena selección de variables se obtiene una mejor clasificación mediante KNN. Muchos estudios consideran que lo mejor es seleccionar la analogía más cercana (k=1). En este trabajo se han hecho pruebas para distintos valores de K (1,2,3,4) para cubrir los valores más utilizados.

Después de seleccionar los proyectos más parecidos, la predicción de un nuevo proyecto está determinada por múltiples técnicas, por el proyecto más parecido, la media de los proyectos, la mediana… En este trabajo se utiliza la media de los k-nearest neighbors para estimar el coste de esfuerzo.

## Comparativa de algoritmos de Feature Selection

Los 4 algoritmos propuestos en el apartado 2.2 son comparados y probados respecto a una base. En este caso un algoritmo de greedy forward selection (GFS) es utilizado para comparar la precisión y la eficiencia de los algoritmos propuestos, basados en una o dos listas ordenadas.

GFS empieza evaluando todas las Features del subset, en la primera iteración se incluye la mejor Feature. En la siguiente iteración se busca la mejor teniendo en cuenta la previamente seleccionada y todas las restantes.

Para la comparativa se tiene en cuenta la precisión y el coste computacional de los algoritmos propuestos. Además del análisis del número de variables seleccionadas y la preferencia de uso de estas ha sido examinado.

## Multiple 3-Fold Cross Validation

La validación cruzada o cross-validation es una técnica muy utilizada para evaluar los resultados de un análisis estadístico. 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.

# Resultados Experimentales

A continuación, se presentan los resultados experimentales. Se analiza el rendimiento de los cuatro algoritmos de FS propuestos y es comparado contra el algoritmo GFS, considerando el error y analizando las variables seleccionadas.

## Precisión de los algoritmos de Feature Selection

### 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.

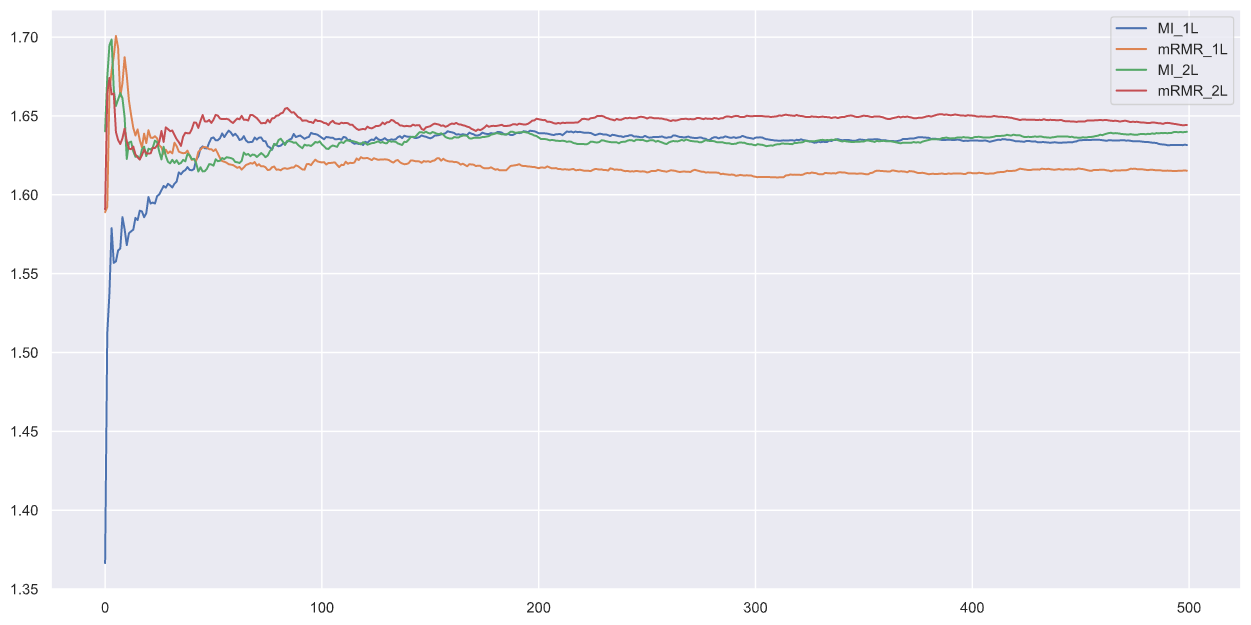


Figura 1 Evolución de las medias acumuladas de MMRE para k=1

### Influencia del valor K en la precisión de los algoritmos

### Precisión de los algoritmos de Feature Selection

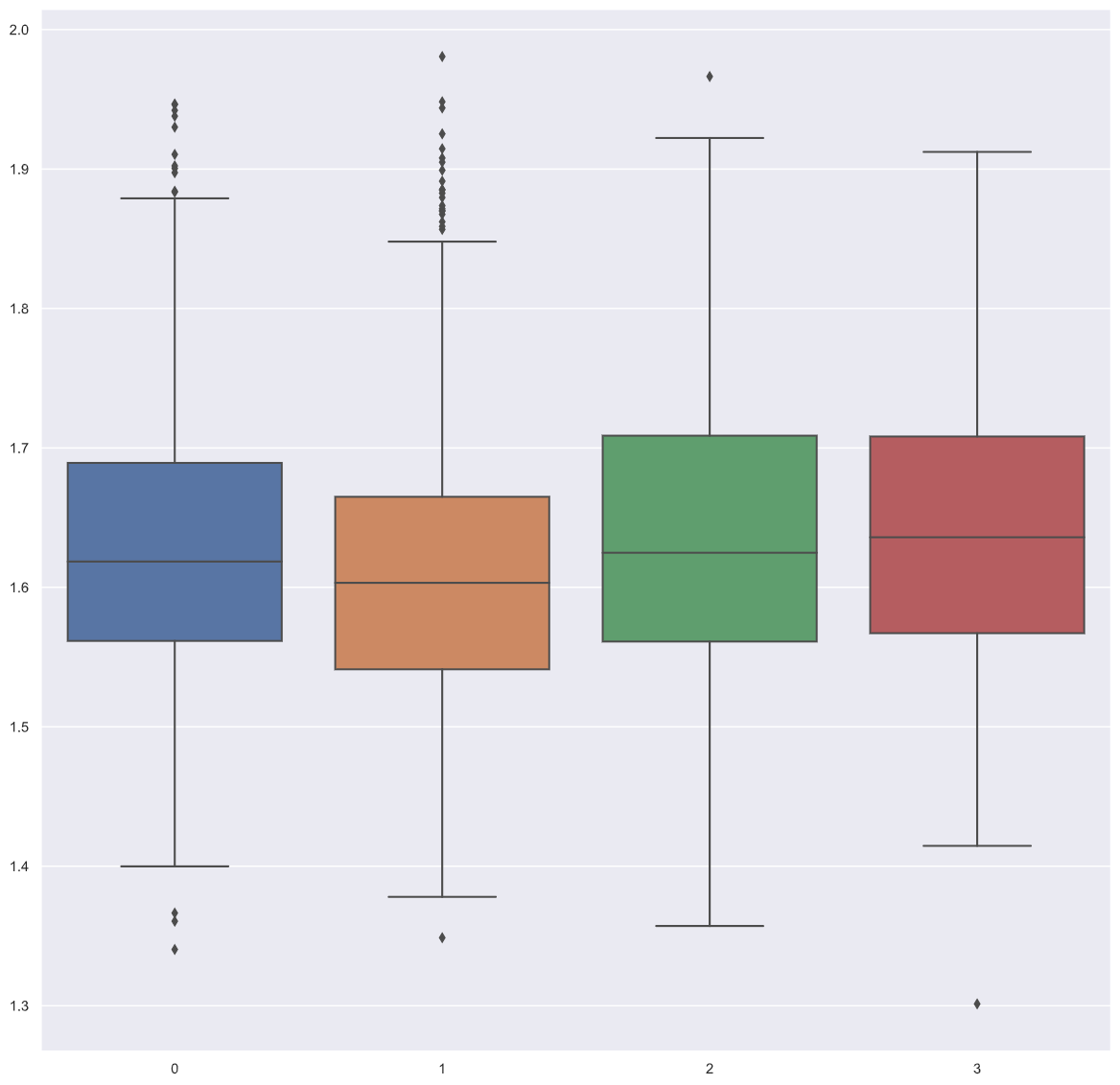


Figura 2 Box plot de la precisión (MMRE) dependiendo de los algoritmos FS

A parte de la precisión de la predicción, el coste computacional también se tiene en cuenta. Los algoritmos están probados en un AMD Ryzen 5 1600 @3.80 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.

Tabla 3 Tiempos de ejecución de los algoritmos

|  |  |  |
| --- | --- | --- |
| Algoritmo | Media (segundos) | Desviación Típica |
| MI\_1L | 25.718 | 0.133 |
| mRMR\_1L | 26.084 | 0.289 |
| MI\_2L | 36.213 | 4.268 |
| mRMR\_2L | 36.141 | 4.052 |

## Análisis de las features seleccionadas

### Información Mutua y Redundancia de las variables independientes

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.

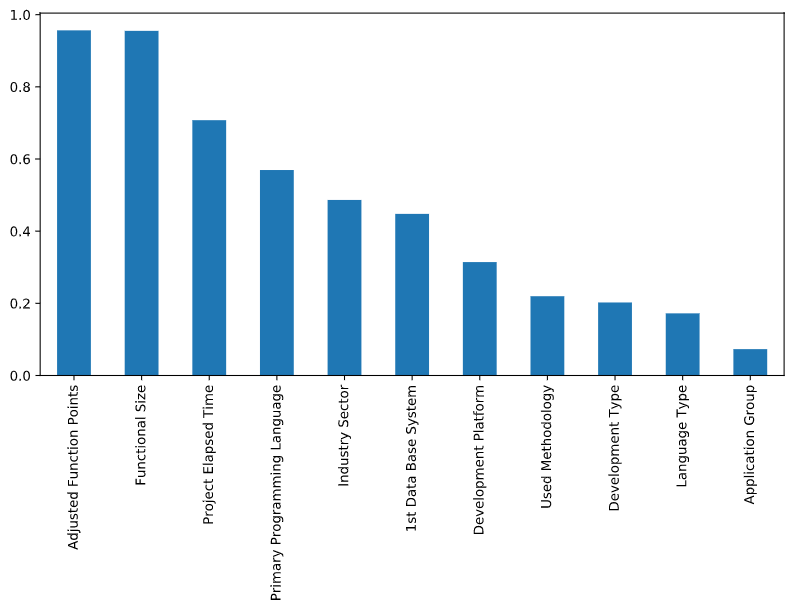
El resultado se ha obtenido con la función normalized\_mutual\_info\_score aplicada a todas las columnas del dataset. Pero se han hecho pruebas con distintas librerías tanto en Python como integrando R.

Figura 3 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

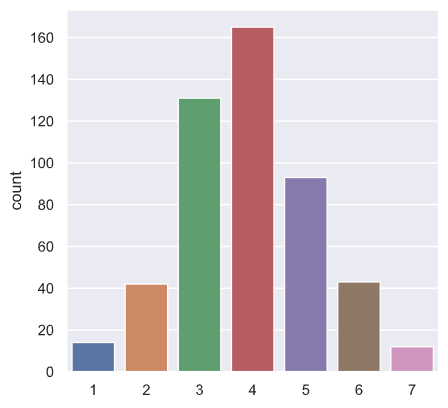
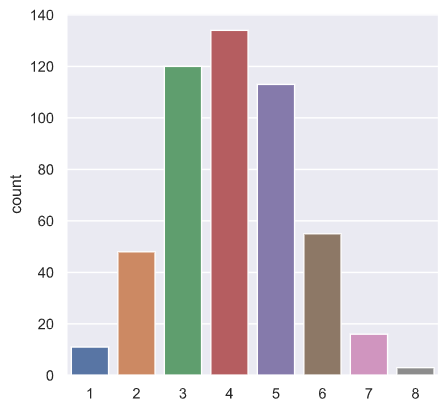
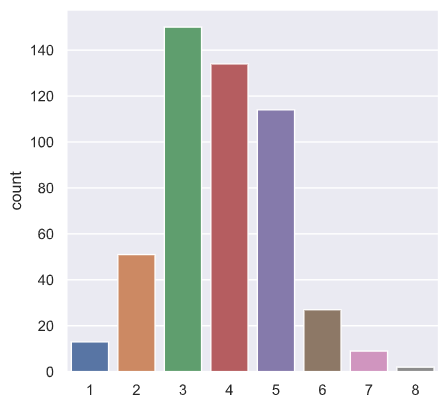
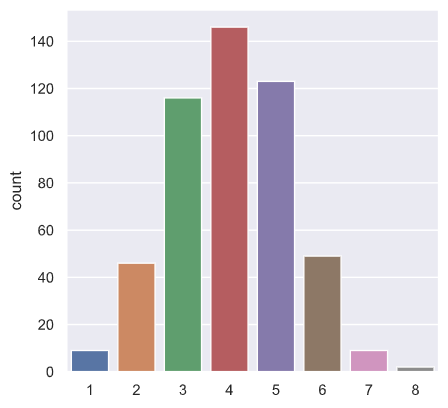
### Numero de Variables seleccionadas dependiendo de los algoritmos

Figura 4 MRMR de las variables independientes

Figura 5 mRMR de las variables seleccionadas

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.



1. MI\_1L
2. mRMR\_1L
3. MI\_2L
4. mRMR\_2L

Tabla 4 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 |

### Preferencia de uso de las variables dependiendo del algoritmo

# Tecnología utilizada

## Python

Python[[1]](#footnote-1) 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.

## Python para todos (1): Instalación del entorno Anaconda - Think ...Anaconda

Anaconda[[2]](#footnote-2) 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.

## Pandas

En computación y ciencia de datos Pandas[[3]](#footnote-3) 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 - Wikipedia, la enciclopedia libreScikit-learn

Scikit-learn[[4]](#footnote-4) 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.

## Proyecto Jupyter - Wikipedia, la enciclopedia libreJupyter Notebook

Jupyter[[5]](#footnote-5) 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.

## Matplotlib

Matplotlib[[6]](#footnote-6) 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[[7]](#footnote-7) 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.

# Proceso de desarrollo

Tras realizar las correspondientes valoraciones previas con la tutora del trabajo, se plantean los primeros retos a superar. Por ejemplo, se plantea el utilizar la librería Pandas como ayuda ya que posee muchas funcionalidades útiles a la hora de trabajar con datos.

Como punto de partida se aprende a:

* Generar y manipular un dataframe
* Importar datos desde un csv
* Recortar dataframes
* Generar gráficas muy básicas con los datos del dataframe.

Otra parte de la puesta en marcha es la creación de un repositorio donde se incluirá todo el código y los datos necesarios para la realización del proyecto. También se utilizará para compartir la memoria del trabajo y las actas de todas las reuniones.

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Importación de la BD | 10 | Algoritmo de MMRE |
| 2 | 1er filtrado de columnas | 11 | Evaluador |
| 3 | Filtrado de proyectos | 12 | Algoritmo greedy forward selection |
| 4 | 2º Filtrado de columnas | 13 | Algoritmo doquire forward selection |
| 5 | Eliminación de valores nulos | 14 | Script de ejecución de cálculos |
| 6 | Limpieza de las variables seleccionadas (1DBS) | 15 | Creación de las gráficas |
| 7 | Recodificación del DF |
| 8 | Algoritmos de MI |
| 9 | Algoritmos de mRMR |

Algunas de las tareas se han subdividido para que sea más fácil su realización.

## Importación de la BD

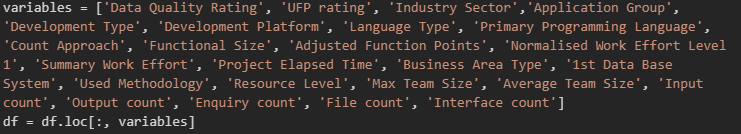
Como paso inicial se realiza la importación de la BD. En este caso se trata de un archivo csv delimitado por punto y coma. Como el archivo contenía más información que la propia base de datos, todos los datos innecesarios tales como recordatorios legales o de categorización de variables se eliminan. 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()[[8]](#footnote-8) pasándole una lista con los identificadores de las columnas a seleccionar.

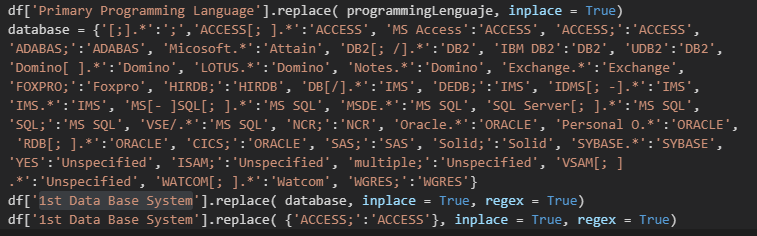


De esta forma vamos acotando el dataset con las columnas que queremos.

## Limpieza de las variables seleccionadas (1DBS)

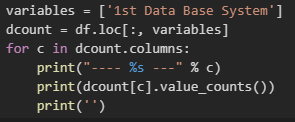
Por desgracia la columna de 1st DataBase System no está correctamente formateada para su utilización. En la mayoría de los proyectos aparecen más de 1 sistema, o está repleto de “;”, o un mismo valor aparece codificado de más de 1 forma.

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[[9]](#footnote-9). 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.



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 que esta tarea se más sencilla es útil la utilización de un script que recorre la columna deseada contando los valores que aparecen en ella.



De esta forma si no hemos contemplado uno de los patrones con las expresiones regulares lo veremos y por tanto podemos ajustar nuestro script de limpieza.

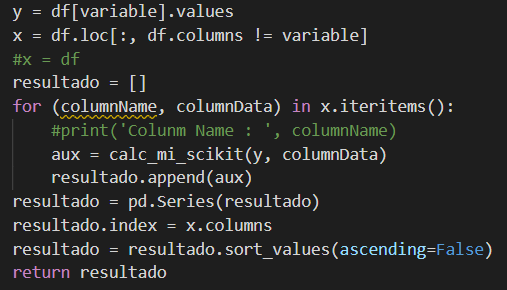
También se ha tenido que realizar una limpieza de la columna Primary Programming Language, pero en menor escala y por tanto mucho más sencilla.



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.

## Algoritmos de MI

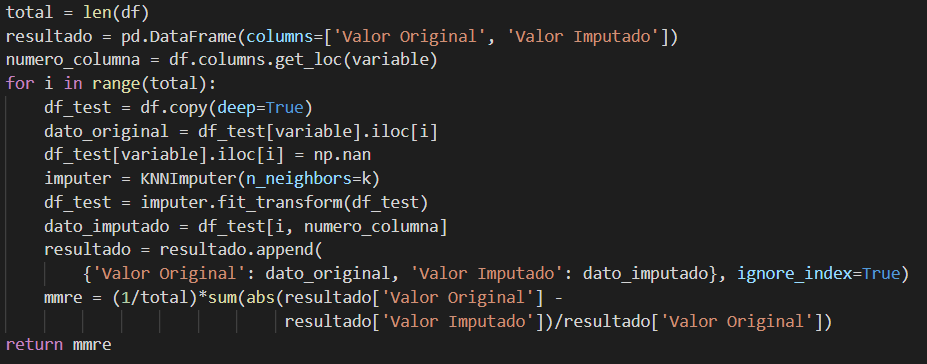
Para realizar el calculo de Mutual Information de las variables del dataframe. Se separan los valores de la variable objetivo del resto. Una vez lo tenemos separado recorremos cada una de las variables dependientes con la variable objetivo utilizando la función de MI.



En este caso la función devuelve un pandas.Series[[10]](#footnote-10) 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 A, y formas de calcular MI finalmente quedándonos con la que más nos ha gustado.

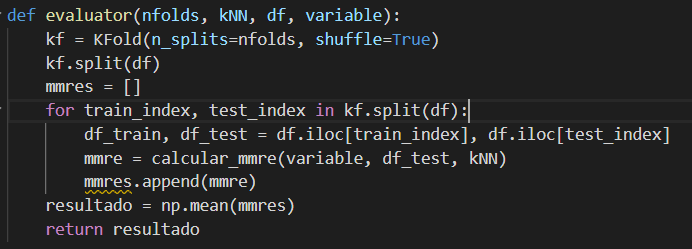
## Algoritmo MMRE

En este caso para el calculo del MMRE se ha utilizado KNNImputer[[11]](#footnote-11) 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. 

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[[12]](#footnote-12), 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.



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

## Greedy Forward Selection y Doquire Forward Selection

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.

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.

## Generación de las 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:



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

En un primer momento se utilizó matplot, como se puede ver en las graficas 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.

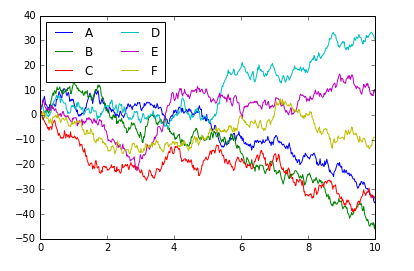
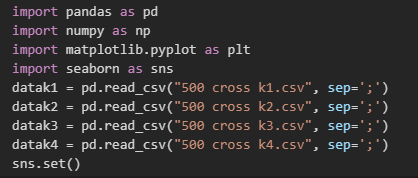


Ilustración 1 Ejemplo gráfica Matplot



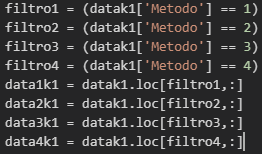
Ilustración 2 Ejemplo gráfica seaborn

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

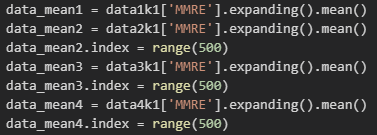


Sns.set() nos permite establecer seaborn para que trabaje directamente con los Dataframes de Pandas.

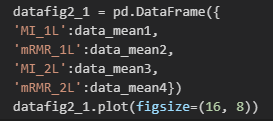
Con la estructura de datos diseñada como resultado de las ejecuciones del algoritmo, se genera 1 columna con todos los valores de MMRE para cada una de las iteraciones. Para acceder a ellas, separamos los datos en los distintos métodos aplicando un filtro, como se ha hecho anteriormente con los proyectos.



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()[[13]](#footnote-13) y mean() del pandas.Dataframe.



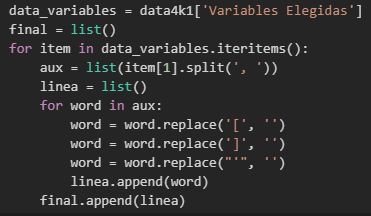
Para generar la gráfica la forma más sencilla es utilizar un DataFrame con los datos.



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.



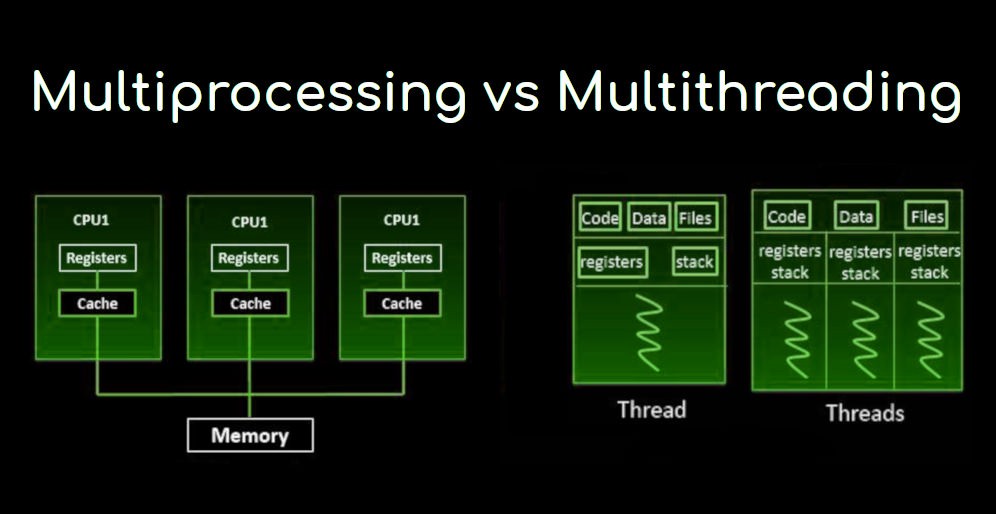
Con una correcta codificación de los resultados, al realizar la importación en pandas podemos definirlo como que son objetos y por tanto este proceso no sería necesario.

# Conclusiones y Trabajo Futuro

## Dimensiones de la solución

## Trabajo Futuro y mejoras

Durante la realización del trabajo se ha experimentado con realizar una solución multiproceso con el fin de aprovechar al máximo la maquina que genera los cálculos. Desgraciadamente la falta de tiempo ha impedido el generar una solución válida utilizando este método. No obstante, el script se modificó a mitad para poder ser lanzado en paralelo y de esta forma obtener los datos más rápidamente. Como los cálculos se han realizado para varios valores de k la división se ha realizado en los distintos valores de esta.



# Referencias

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/>

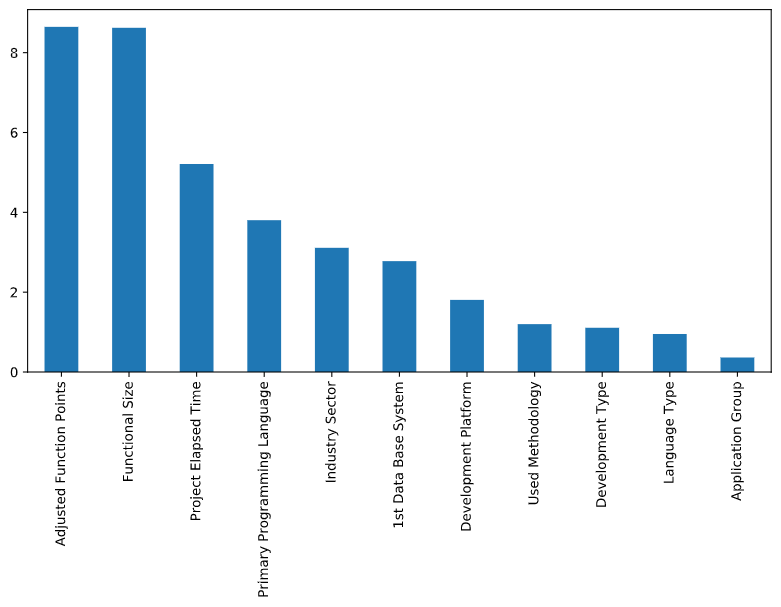
1. Documentación y tutoriales Seaborn

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

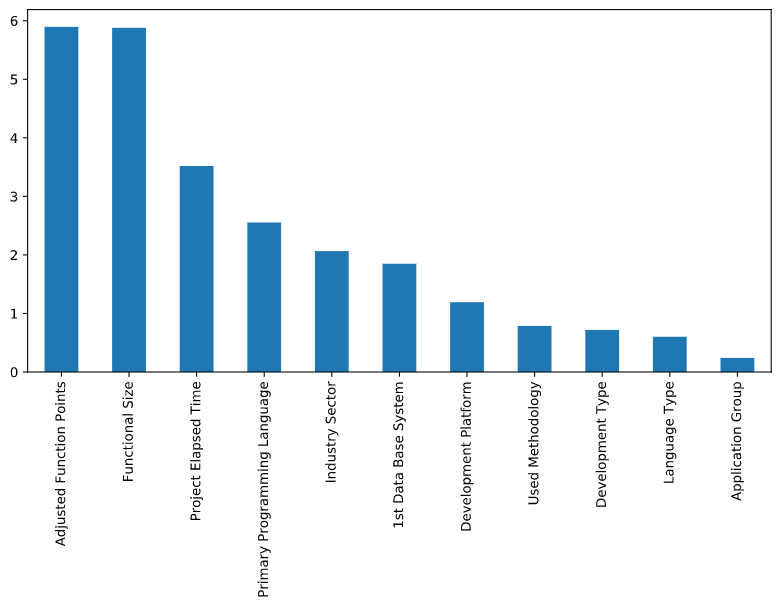
# Anexo

## Anexo A – Gráficas de ejecución de MI

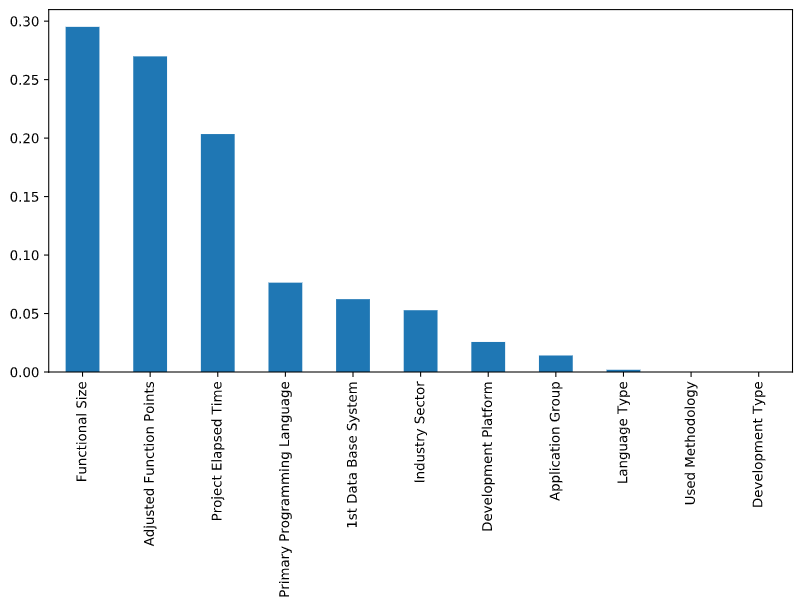
Esta primera gráfica se corresponde a la librería info\_gain.

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

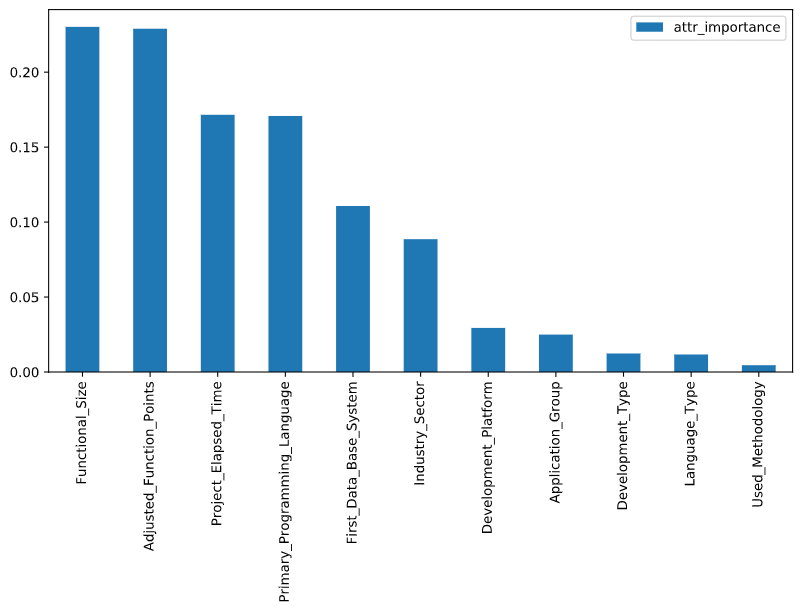
Esta grafica se hace aplicando manualmente al dataframe mutual\_info\_score de scikit

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

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

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

Gráfica de Mi con Fselector

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

Para la integración de Python con R se utiliza RPY2, se hace en detalle en el Anexo B.

## Anexo B – 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[[14]](#footnote-14).

Para realizar la instalación del modulo 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.

1. <https://www.python.org/> [↑](#footnote-ref-1)
2. <https://www.anaconda.com/> [↑](#footnote-ref-2)
3. <https://pandas.pydata.org/> [↑](#footnote-ref-3)
4. <https://scikit-learn.org/> [↑](#footnote-ref-4)
5. <https://jupyter.org/> [↑](#footnote-ref-5)
6. <https://matplotlib.org/> [↑](#footnote-ref-6)
7. <https://seaborn.pydata.org/> [↑](#footnote-ref-7)
8. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.loc.html> [↑](#footnote-ref-8)
9. <https://www.w3schools.com/python/python_regex.asp> [↑](#footnote-ref-9)
10. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.html> [↑](#footnote-ref-10)
11. <https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html> [↑](#footnote-ref-11)
12. <https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html> [↑](#footnote-ref-12)
13. <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.expanding.html> [↑](#footnote-ref-13)
14. <https://pypi.org/project/rpy2/> [↑](#footnote-ref-14)